Faster Algorithms for Convex and Combinatorial Optimization

by

Yin Tat Lee

Submitted to the Department of Mathematics
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Mathematics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2016

Author .....
Department of Mathematics
May 13, 2016

Certified by .....
Jonathan A. Kelner
Associate Professor
Thesis Supervisor

Accepted by .....
Jonathan A. Kelner
Chairman, Applied Mathematics Committee

Faster Algorithms for Convex and Combinatorial Optimization

by
Yin Tat Lee

Submitted to the Department of Mathematics
on May 13, 2016, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Mathematics

Abstract

In this thesis, we revisit three algorithmic techniques: sparsification, cutting and collapsing. We use them to obtain the following results on convex and combinatorial optimization:

- **Linear Programming**: We obtain the first improvement to the running time for linear programming in 25 years. The convergence rate of this randomized algorithm nearly matches the universal barrier for interior point methods. As a corollary, we obtain the first $\tilde{O}(m\sqrt{n})$ time randomized algorithm for solving the maximum flow problem on directed graphs with m edges and n vertices. This improves upon the previous fastest running time of $\tilde{O}(m \min(n^{2/3}, m^{1/2}))$, achieved over 15 years ago by Goldberg and Rao.

- **Maximum Flow Problem**: We obtain one of the first almost-linear time randomized algorithms for approximating the maximum flow in undirected graphs. As a corollary, we improve the running time of a wide range of algorithms that use the computation of maximum flows as a subroutine.

- **Non-Smooth Convex Optimization**: We obtain the first nearly-cubic time randomized algorithm for convex problems under the black box model. As a corollary, this implies a polynomially faster algorithm for three fundamental problems in computer science: submodular function minimization, matroid intersection, and semidefinite programming.

- **Graph Sparsification**: We obtain the first almost-linear time randomized algorithm for spectrally approximating any graph by one with just a linear number of edges. This sparse graph approximately preserves all cut values of the original graph and is useful for solving a wide range of combinatorial problems. This algorithm improves all previous linear-sized constructions, which required at least quadratic time.

- **Numerical Linear Algebra**: Multigrid is an efficient method for solving large-scale linear systems which arise from graphs in low dimensions. It has been used extensively for 30 years in scientific computing. Unlike the previous approaches that make assumptions on the graphs, we give the first generalization of the multigrid that provably solves Laplacian systems of any graphs in nearly-linear expected time.

Thesis Supervisor: Jonathan A. Kelner
Title: Associate Professor

Acknowledgments

First, I would like to thank my advisor Jonathan Kelner. Jon is always full of ideas and provided me a lot of high-level inspiration. I am particularly thankful to him for encouraging me to learn interior point methods, which led to many of key results in this thesis. From the very beginning, Jon has given me lots of support, freedom and advice. Because of him, I have achieved what I never imagined. I will be forever grateful to Jon.

Next, I would like to thank my colleague, Aaron Sidford. Over the past 4 years, one of my most exciting activities has been discussing with Aaron in how to solve Maximum Flow faster. During thousands of hours of discussions, we obtained many of the results presented in this thesis. Although we still haven't gotten the ultimate result we want, I believe, someday, we will get it.

I would also like to thank Lap Chi Lau. Originally, I was planning to study scientific computing, but he made me change my mind. Since my undergraduate, I have been asking him for all kinds of career advice and, up to now, all of them are proven to be useful.

Next, I would like to thank Pravin Vaidya for his beautiful work on his interior point methods, cutting plane methods and Laplacian systems. Vaidya's results were definitely far ahead of his time and many results in this thesis are inspired by and built on his results.

Also, I would like to thank my academic friends and collaborators: Sébastien Bubeck, Michael Cohen, Laurent Demanet, Michel Goemans, Michael Kapralov, Tsz Chiu Kwok, Rasmus Kyng, Aleksander Mądry, Gary Miller, Cameron Musco, Christopher Musco, Lorenzo Orecchia, Shayan Oveis Gharan, Jakub Pachocki, Richard Peng, Satish Rao, Sushant Sachdeva, Ludwig Schmidt, Mohit Singh, Daniel Spielman, Nikhil Srivastava, Robert Strichartz, He Sun, Luca Trevisan, Santosh Vempala, Nisheeth Vishnoi, Matt Weinberg, Sam Chiu-wai Wong, and Zeyuan Allen Zhu.

Last but not the least, I would like to thank my parents Man Cheung and Sau Yuen, my siblings Ah Chi and Ah Tat, and my wife Yan Kit.

Yin Tat Lee
Cambridge, Massachusetts
May 2016

My work was partially supported by NSF awards 0843915 and 1111109. Some of my work was done while I was visiting the Simons Institute for the Theory of Computing, UC Berkeley.

Contents

Chapter 1

Introduction

Convex optimization has been studied extensively and is a prominent tool used in various areas such as combinatorial optimization, data analysis, operations research, and scientific computing. Each field has developed specialized tools including data structures, sampling methods, and dimension reduction. In this thesis, we combine and improve the optimization techniques from different fields to design provably faster optimization algorithms for several important problems, including linear programming, general non-smooth convex minimization, and maximum flow problems.

1.1 Techniques

The techniques used in this thesis can be categorized into three types: sparsification, cutting and collapsing. Here, we explain these three techniques and highlight their applications to convex optimization and combinatorial optimization.

1.1.1 Sparsification

Sparsification techniques originated from graph theory, and it refers to the task of approximating dense graphs by sparse graphs. There are different types of sparsifiers specifically designed for different graph problems, such as spanners for distance-related problems [49], cut sparsifiers for cut-related problems [33], spectral sparsifiers for spectral related problems [243, 242]. For instance, if we want to compute a minimum s-t cut on a given dense graph, instead of running our favorite s-t cut algorithm on the original graph, we can first compute a cut sparsifier of the original graph, then run the s-t cut algorithm on the sparsifier. These sparsifiers allow us to speed up graph algorithms for many problems on dense graphs by paying some error introduced by the sparsifiers.

Beyond graph related problems, this notion of spectral sparsifier has been generalized and applied to solve different matrix-related problems [76, 170, 56]. In the linear regression problem, we are given a matrix $\boldsymbol{A}$ and a vector $\boldsymbol{b}$, and we want to minimize $\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|_2$ over all $\boldsymbol{x}\in\mathbb{R}^n$. The sparsification concept allows us to compress a tall matrix $\boldsymbol{A}$ into an almost square matrix. This is useful in various contexts such as statistics problems with much more samples than variables. However, how can we find those sparsifiers in the first place without solving a dense problem? How sparse of a matrix can we effectively attain? How general is the idea of sparsification? In Part I, we answer these questions. In particular, we show how to solve overdetermined systems faster, how to find linear-sized sparsifiers of graphs in almost-linear time, and how to find geometric medians of point clouds in nearly-linear time. In one of the key results of this part, we extend the idea of sparsification to linear programming and give the first improvement to the running time for linear programming in 25 years.

1.1.2 Cutting

Cutting refers to the task of iteratively refining a region that contains the solution. This technique is particularly useful for minimizing convex functions. In combinatorial optimization, Khachiyan's seminal result in 1980 [142] proved that the ellipsoid method, a cutting plane method[1], solves linear programs in polynomial time. Since then, cutting plane methods have been crucial to solving discrete problems in polynomial time [110]. In continuous optimization, cutting plane methods have long played a critical role in convex optimization, where they are fundamental to the theory of non-smooth optimization [102].

Despite the key role that cutting plane methods have played historically in both combinatorial and convex optimization, over the past two decades, the theoretical running time of cutting plane methods for convex optimization has not been improved since the breakthrough result by Vaidya in 1989 [253, 253]. Moreover, for many of the key combinatorial applications of cutting plane methods, such as submodular minimization, matroid intersection, and submodular flow, the running time improvements over the past two decades have been primarily combinatorial; that is they have been achieved by discrete algorithms that do not use numerical machinery such as cutting plane methods.

In Part II, we make progress towards cutting plane methods on multiple directions. Firstly, we show how to improve on the running time of cutting plane methods. Secondly, we provide several frameworks for applying the cutting plane methods and illustrate the efficacy of these frameworks by obtaining faster running times for semidefinite programming, matroid intersection, and submodular flow. Thirdly, we show how to couple our approach with the problem-specific structure and obtain faster weakly and strongly polynomial running times for submodular function minimization, a problem of tremendous importance in combinatorial optimization. In both cases, our algorithms are faster than the previous best running times by a factor of roughly $\Omega(n^2)$. Finally, we give some variants of cutting plane methods that is suitable for large-scale problems and demonstrate empirically that our new methods compare favorably with the state-of-the-art algorithms.

1.1.3 Collapsing

Collapsing refers to the task of removing variables from problems while (approximately) preserving solutions. There are different collapsings for different situations, such as mathematical induction for proofs, Gaussian elimination and Schur complement for matrices, and Fourier–Motzkin elimination for linear programs. Since collapsing often increases the density of the problems, it often does not lead to efficient algorithms.

In a seminal work of Spielman and Teng, they showed that this density issue can be avoided for graph Laplacians via sparsification. In Part III, we generalize this idea and show how to combine collapsing and sparsification for the maximum flow problem and various linear systems. As a result, we obtain the first almost-linear time algorithms for approximating the maximum flow in undirected graphs[2] and the first nearly-linear-time algorithms for solving systems of equations in connection Laplacians, a generalization of Laplacian matrices that arises in many problems in image and signal processing.

In general, we assume $T(m, n)$ to be the time needed to solve a certain graph problem on a graph with m edges and n vertices. Suppose that we know $\mathcal{T}(m, n) = O(m)+O(\mathcal{T}(O(n), n))$ by sparsification and $\mathcal{T}(m, n) = O(m) + O(T(O(m), n/2))$ by collapsing, then it is easy to show that $\mathcal{T}(m, n) = O(m)$

[1] Throughout this thesis, our focus is on algorithms for polynomial time solvable convex optimization problems given access to a linear separation oracle. Our usage of the term *cutting plane methods*, should not be confused with work on integer programming.

[2] We also note that independently, Jonah Sherman produced an almost-linear time algorithm for maximum flow.

by alternatively sparsifying and collapsing. Although we do not know how to do sparsifying and collapsing in general, we believe that the concept of combining sparsifying and collapsing is a powerful technique that it will provide a foundation of future works.

1.2 Results

By applying and improving the three techniques mentioned above, we have substantially advanced the state-of-the-art algorithms for solving many fundamental problems in computer science. Here, we describe three of the key results in this thesis.

1.2.1 Convex Programming

Many optimization problems in theoretical computer science are shown to be polynomial-time-solvable by reducing to one of the following two general problems:

- **The Feasibility Problem**: Given a set $K \subset \mathbb{R}^n$ and a separation oracle that, given a point $\boldsymbol{x}$, either outputs that $\boldsymbol{x}$ is in K or outputs a separating hyperplane. The problem is to find a point in K or prove that K is almost empty.

- **The Intersection Problem**: Given two sets $K_1, K_2 \subset \mathbb{R}^n$, any vector $\boldsymbol{d}$ and an optimization oracle that, given a vector $\boldsymbol{c}$, outputs an $\boldsymbol{x}_1$ that maximizes $\boldsymbol{c}^\top \boldsymbol{x}_1$ over K_1 and $\boldsymbol{x}_2$ that maximizes $\boldsymbol{c}^\top \boldsymbol{x}_2$ over K_2. The problem is to find a point $\boldsymbol{x}$ nearly inside $K_1 \cap K_2$ such that it approximately maximizes $\boldsymbol{d}^\top \boldsymbol{x}$.

For many problems, this reduction is either the best or the only known approach. Therefore, it is important to find an efficient algorithm for these two general problems. Since a breakthrough of Vaidya in 1989 [253], the feasibility problem can be solved in $\widetilde{O}(nT + n^{\omega+1})$ time (Table 1.1), where T is the cost of the oracle and $\omega \sim 2.373$. Furthermore, using the seminal works of Grötschel, Lovász, Schrijver and independently, Karp and Papadimitriou in 1981 and 1982 [110, 138], the intersection problem can be solved by solving n feasibility problems. In this thesis, we show that both problems can be solved in $\widetilde{O}(nT + n^3)$ expected time[3], the first improvement for both problems over 25 years.

As a corollary of these two theorems, we obtained polynomial improvements for many important problems, such as submodular function minimization (Table 1.2), matroid intersection (Table 1.3, 1.4), submodular flow (Table 1.5), and semidefinite programming (Table 1.6).

Year	Algorithm	Complexity
1979	Ellipsoid Method [236, 274, 142]	$\widetilde{O}(n^2T + n^4)$
1988	Inscribed Ellipsoid [144, 213]	$\widetilde{O}(nT + n^{4.5})$
1989	Volumetric Center [253]	$\widetilde{O}(nT + n^{\omega+1})$
1995	Analytic Center [20]	$\widetilde{O}(nT + n^{\omega+1})$
2004	Random Walk [36]	$\rightarrow \widetilde{O}(nT + n^7)$
-	Chapter 8	$\widetilde{O}(nT + n^3)$

Table 1.1: Algorithms for the Feasibility Problem. The arrow, $\rightarrow$, indicates that it solves a more general problem where only a membership oracle is given.

[3] In this thesis, nearly all algorithms are randomized and nearly all time are expected running time.

Year	Author	Complexity
1981,1988	Grötschel, Lovász, Schrijver [110, 109]	$\widetilde{O}(n^5 \cdot \mathrm{EO} + n^7)$ [189]
1985	Cunningham [61]	$O(Mn^6 \log(nM) \cdot \mathrm{EO})$
2000	Schrijver [231]	$O(n^8 \cdot \mathrm{EO} + n^9)$
2000	Iwata, Fleischer, Fujishige [121]	$O(n^5 \cdot \mathrm{EO} \log M)$ $O(n^7 \log n \cdot \mathrm{EO})$
2000	Iwata, Fleischer [87]	$O(n^7 \cdot \mathrm{EO} + n^8)$
2003	Iwata [118]	$O((n^4 \cdot \mathrm{EO} + n^5) \log M)$ $O((n^6 \cdot \mathrm{EO} + n^7) \log n)$
2003	Vygen [264]	$O(n^7 \cdot \mathrm{EO} + n^8)$
2007	Orlin [216]	$O(n^5 \cdot \mathrm{EO} + n^6)$
2009	Iwata, Orlin [125]	$O((n^4 \cdot \mathrm{EO} + n^5) \log(nM))$ $O((n^5 \cdot \mathrm{EO} + n^6) \log n)$
-	Chapter 9	$O(n^2 \log(nM) \cdot \mathrm{EO} + n^3 \log^{O(1)}(nM))$ $O(n^3 \log^2 n \cdot \mathrm{EO} + n^4 \log^{O(1)} n)$

Table 1.2: Algorithms for submodular function minimization. EO is the time for evaluation oracle of the submodular function and M is the maximum absolute function value.

Year	Author	Complexity
1968	Edmonds [81]	not stated
1971	Aigner, Dowling [6]	$O(n^3 \mathcal{T}_{\mathrm{ind}})$
1975	Lawler [157]	$O(n^3 \mathcal{T}_{\mathrm{ind}})$
1986	Cunningham [60]	$O(n^{2.5} \mathcal{T}_{\mathrm{ind}})$
-	Chapter 9	$\widetilde{O}(n^2 \mathcal{T}_{\mathrm{ind}} + n^3)$

Table 1.3: Algorithms for (unweighted) matroid intersection. n is the size of the ground set and $\mathcal{T}_{\mathrm{ind}}$ is the time to check if a set is independent.

Year	Author	Complexity
1978	Fujishige [92]	not stated
1981	Grotschel, Lovasz, Schrijver[110]	weakly polynomial
1987	Frank, Tardos [91]	strongly polynomial
1993	McCormick, Ervolina [190]	$O(n^7 h^* \log(nCU))$
1994	Wallacher, Zimmermann [266]	$O(n^8 h \log(nCU))$
1997	Iwata [117]	$O(n^7 h \log U)$
1998	Iwata, McCormick, Shigeno [123]	$O\left(n^4 h \min\left\{\log(nC), n^2 \log n\right\}\right)$
1999	Iwata, McCormick, Shigeno [124]	$O\left(n^6 h \min\left\{\log(nU), n^2 \log n\right\}\right)$
1999	Fleischer, Iwata, McCormick[89]	$O\left(n^4 h \min\left\{\log U, n^2 \log n\right\}\right)$
1999	Iwata, McCormick, Shigeno [122]	$O\left(n^4 h \min\left\{\log C, n^2 \log n\right\}\right)$
2000	Fleischer, Iwata [88]	$O(mn^5 \log(nU) \cdot \mathrm{EO})$
-	Chapter 9	$O(n^2 \log(nCU) \cdot \mathrm{EO} + n^3 \log^{O(1)}(nCU))$

Table 1.5: Algorithms for minimum cost submodular flow with n vertices, maximum cost C and maximum capacity U. The factor h is the time for an exchange capacity oracle, h^* is the time for a more complicated exchange capacity oracle, and EO is the time for evaluation oracle.

Year	Author	Complexity
1968	Edmonds [81]	not stated
1975	Lawler [157]	$O(n^3\mathcal{T}_{\text{ind}} + n^4)$
1981	Frank [90]	$O(n^3(\mathcal{T}_{\text{circuit}} + n))$
1986	Brezovec, Cornuéjols, Glover[39]	$O(n^2(\mathcal{T}_{\text{circuit}} + n))$
1995	Fujishige, Zhang [96]	$O(n^{2.5}\log(nM)\cdot\mathcal{T}_{\text{ind}})$
1995	Shigeno, Iwata [234]	$O((n+\mathcal{T}_{\text{circuit}})n^{1.5}\log(nM))$
-	Chapter 9	$O(n^2\log(nM)\mathcal{T}_{\text{ind}} + n^3\log^{O(1)}(nM))$

Table 1.4: Algorithms for weighted matroid intersection. In addition to the notation in Table 1.3, $\mathcal{T}_{\text{circuit}}$ is the time needed to find a fundamental circuit and M is the maximum weight.

Year	Author	Complexity
1992	Nesterov, Nemirovsky[210]	$\tilde{O}(\sqrt{m}(nm^\omega + n^{\omega-1}m^2))$
2000	Anstreicher [13]	$\tilde{O}((mn)^{1/4}(nm^\omega + n^{\omega-1}m^2))$
2003	Krishnan, Mitchell [152]	$\tilde{O}(n(n^\omega + m^\omega + S))$ (dual SDP)
-	Chapter 9	$\tilde{O}(n(n^2 + m^\omega + S))$

Table 1.6: Algorithms for solving a $m \times m$ SDP with n constraints and S non-zero entries

1.2.2 Linear Programming

For an arbitrary linear program $\min_{\boldsymbol{x} \text{ s.t } \boldsymbol{A}\boldsymbol{x}\geq\boldsymbol{b}} \boldsymbol{c}^\top\boldsymbol{x}$ with n variables and m constraints, the fastest algorithm required either solving $\widetilde{O}(\sqrt{m})$ linear systems or inverting $\widetilde{O}((mn)^{1/4})$ dense matrices [227, 253]. However, since a breakthrough of Nesterov and Nemirovski in 1994 [211], it has been known that $\widetilde{O}(\sqrt{n})$ iterations in fact suffice. Unfortunately, each iteration of their algorithm requires computing the center of gravity of a polytope, which is a problem that is harder than linear programming. In this thesis, we resolved this 25-year-old computational question and developed an efficient algorithm that requires solving only $\widetilde{O}(\sqrt{n})$ linear systems (Table 1.7).

1.2.3 Maximum Flow Problem

The maximum flow problem has been studied extensively for more than 60 years; it has a wide range of theoretical and practical applications and is widely used as a key subroutine in other algorithms. Until 5 years ago, work on the maximum flow and the more general problems of linear programming and first order convex optimization was typically performed independently. Nevertheless, all known

Year	Author	Number of Iterations	Nature of iterations
1984	Karmarkar [136]	$\widetilde{O}(m)$	Linear system solve
1986	Renegar [227]	$\widetilde{O}(\sqrt{m})$	Linear system solve
1989	Vaidya [253]	$\widetilde{O}((mn)^{1/4})$	Expensive linear algebra
1994	Nesterov and Nemirovskii [211]	$\widetilde{O}(\sqrt{n})$	Volume computation
-	Chapter 6	$\widetilde{O}(\sqrt{n})$	$\widetilde{O}(1)$ Linear system solves

Table 1.7: Algorithms for linear programs with n variables and m constraints

Year	Author	Complexity	Directed	Weighted
1998	Goldberg & Rao [104]	$\widetilde{O}\left(m \min\left(m^{1/2}, n^{2/3}\right)\right)$	Yes	Yes
1998	Karger [134]	$\widetilde{O}\left(m\sqrt{n}\varepsilon^{-1}\right)$	No	Yes
2002	Karger & Levine [135]	$\widetilde{O}\left(nF\right)$	No	Yes
2011	Christiano et al. [51]	$\widetilde{O}\left(mn^{1/3}\varepsilon^{-11/3}\right)$	No	Yes
2012	Lee, Rao & Srivastava [161]	$\widetilde{O}\left(mn^{1/3}\varepsilon^{-2/3}\right)$	No	No
2013	Chapter 12 Sherman [233]	$\widetilde{O}\left(m^{1+o(1)}\varepsilon^{-2}\right)$	No	Yes
2013	Mądry [183]	$\widetilde{O}\left(m^{10/7}\right)$	Yes	No
-	Chapter 6	$\widetilde{O}\left(m\sqrt{n}\right)$	Yes	Yes

Table 1.8: Algorithms for maximum flow problem with n vertices, m edges and maximum flow value F. Results before 1998 are omitted for brevity.

techniques, whether combinatorial or numerical, achieved essentially the same (or worse) running time, $\widetilde{O}(m^{1.5})$ or $\widetilde{O}(mn^{2/3})$ for solving the maximum flow on a weighted graph with n vertices and m edges. Consequently, the maximum flow problem poses a problem for testing the state-of-the-art methods in both convex and combinatorial optimization.

In this thesis, we obtain the first $\widetilde{O}(m\sqrt{n})$ time algorithm for solving the maximum flow problem on directed graphs with m edges and n vertices. This is an improvement over the previous fastest running time of $\widetilde{O}(m \min\{\sqrt{m}, n^{2/3}\})$ achieved over 15 years ago by Goldberg and Rao [104] and the previous fastest running times for solving dense directed unit capacity graphs of $O(m \min\{\sqrt{m}, n^{2/3}\})$ achieved by Even and Tarjan [85] over 35 years ago and of $\widetilde{O}(m^{10/7})$ achieved recently by Mądry [183]. For undirected graphs, we give one of the first algorithm with running time $O(m^{1+o(1)}/\varepsilon^2)$ for approximating maximum flow.

1.2.4 Philosophical Contributions

Besides the improvements on running time in many classical problems in computer science, the secondary goal of this thesis is to further promote the interaction between continuous optimization and discrete optimization.

For many decades, continuous optimization and discrete (combinatorial) optimization have been studied separately. In continuous optimization, researchers have developed optimal iterative methods under various black-box models, and in discrete optimization, researchers have developed various randomization techniques and data structures. Despite the different approaches, convex problems in both fields in fact share some common difficulties. In this thesis, we identify three important components from both fields and demonstrate the combination of both techniques is indeed useful for solving many fundamental problems in optimization.

Despite decades of research on convex optimization and combinatorial optimization, we still do not have nearly optimal time algorithm for the majority of non-trivial convex problems. We strongly believe that the combination of both fields will help advance future research in this area.

Chapter 2

Preliminaries and Thesis Structure

In this chapter, we first describe the problems we encounter throughout this thesis, then we discuss the structure of this thesis. At the end, we list some linear algebraic notations and some facts we commonly used throughout this thesis.

2.1 Problems

2.1.1 Black-Box Problems

We say that a set $K \subseteq \mathbb{R}^n$ is convex if for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ and all $t \in [0,1]$ it holds that $t\boldsymbol{x}+(1-t)\boldsymbol{y} \in K$. We say that a function $f : K \to \mathbb{R}$ is convex if K is convex and for all $\boldsymbol{x}, \boldsymbol{y} \in K$ and all $t \in [0,1]$ it holds that $f(t\boldsymbol{x}+(1-t)\boldsymbol{y}) \leq t \cdot f(\boldsymbol{x}) + (1-t) \cdot f(\boldsymbol{y})$. In this thesis, we are interested in the following minimization problem

$$\min_{\boldsymbol{x} \in K} f(\boldsymbol{x})$$

where both f and K are convex and we call this a convex problem. There are two types of convex problems, black-box and structural problems, depending on if we have full information about the problem.

In black-box problems, the function f and/or the set K are unknown and one can only access them through queries to an oracle. Here, we give a list of black-box problems we are interested in. All of these problems below are very general and cover a wide range of problems in computer science.

- **Convex Minimization:** Given a convex function f and a (sub)gradient oracle that computes $\nabla f(\boldsymbol{x})$ at any point $\boldsymbol{x}$. The problem is to find a point that minimizes f over $\boldsymbol{x} \in \mathbb{R}^n$.

- **The Feasibility Problem:** Given a convex set $K \subset \mathbb{R}^n$ and a separation oracle that, given a point $\boldsymbol{x}$, either outputs that $\boldsymbol{x}$ is in K or outputs a hyperplane separates $\boldsymbol{x}$ and K. The problem is to find a point in K or prove that K is almost empty.

- **The Intersection Problem**: Given two convex sets $K_1, K_2 \subset \mathbb{R}^n$ and an optimization oracle that, given a vector $\boldsymbol{c}$, outputs a vector $\boldsymbol{x}_1$ that maximizes $\boldsymbol{c}^\top \boldsymbol{x}_1$ over K_1 and $\boldsymbol{x}_2$ that maximizes $\boldsymbol{c}^\top \boldsymbol{x}_2$ over K_2. For any vector $\boldsymbol{d}$, the problem is to find a point $\boldsymbol{x}$ nearly inside $K_1 \cap K_2$ such that it approximately maximizes $\boldsymbol{d}^\top \boldsymbol{x}$.

- **Submodular Minimization:** Given an integer-valued function f defined on all subsets of n elements. f is submodular if it satisfies the diminishing returns, i.e.

  $$f(X \cup \{\boldsymbol{x}\}) - f(X) \geq f(Y \cup \{\boldsymbol{x}\}) - f(Y)$$

  for any X, Y such that $X \subset Y$ and any $\boldsymbol{x} \notin Y$. We are given an evaluation oracle that computes $f(S)$ for any subset S. The problem is to find a subset S that minimizes f.

2.1.2 Structural Problems

In structural problems, the function f and the set K are given directly by explicit formulas. In this and the next section, we list some structure problems we are interested in this thesis.

- ℓ_p **regression:** Given a matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and a vector $\boldsymbol{b} \in \mathbb{R}^m$, the ℓ_p regression problem requires us to find the minimizer of

$$\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_p.$$

  We are particularly interested in the case $p = 2$ because many optimization algorithms need to solve a linear system per iteration. As we will see, techniques for solving ℓ_2 regression problems can be sometimes used or generalized to general convex problems.

- **Linear Programming:** Given a matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and vectors $\boldsymbol{b} \in \mathbb{R}^m, \boldsymbol{c} \in \mathbb{R}^n$, linear programming requires us to find the minimizer of

$$\min_{\boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b}} \boldsymbol{c}^\top \boldsymbol{x}.$$

  This is one of the key problems in computer science. In both theory and practice, many problems can be reformulated as linear programs to take advantage of fast algorithms.

- **Semidefinite Programming:** Given a symmetric matrices $\boldsymbol{C} \in \mathbb{R}^{m \times m}$, $\boldsymbol{A}_i \in \mathbb{R}^{m \times m}$ for $i = 1, \cdots, n$, the semidefinite program requires us to find the minimizer of

$$\max_{\boldsymbol{X} \succeq \boldsymbol{0}} \boldsymbol{C} \bullet \boldsymbol{X} \text{ s.t. } \boldsymbol{A}_i \bullet \boldsymbol{X} = b_i.$$

  This is a generalization of linear programming and has a lot of applications in control theory, polynomial minimization, and approximation algorithms.

- **Geometric Median (Fermat-Weber problem):** Given n points $\{\boldsymbol{a}_i\}_{i=1}^m$ in $\mathbb{R}^n$, the geometric median problem requires us to find the minimizer of

$$\min_{\boldsymbol{x}} \sum_{i=1}^m \|\boldsymbol{a}_i - \boldsymbol{x}\|_2.$$

  This is one of the oldest problems in computation geometry and has numerous applications such as facility location, clustering and statistics.

2.1.3 Graph-Related Problems

Here, we define some graph-related problems that are used throughout in this thesis.

We let $G = (V, E)$ denote a graph with $n = |V|$ vertices, $m = |E|$ edges. We let $w_e \geq 0$ denote the weight of an edge. For all undirected graphs in this thesis, we assume an arbitrary orientation of the edges to clarify the meaning of vectors $\boldsymbol{f} \in \mathbb{R}^E$.

Let $\boldsymbol{W} \in \mathbb{R}^{E \times E}$ denote the diagonal matrices associated with the weights and $\boldsymbol{B} \in \mathbb{R}^{E \times V}$ denote the graphs incidence matrix where for all $e = (a, b) \in E$ we have $\boldsymbol{B}^\top \mathbf{1}_e = \mathbf{1}_a - \mathbf{1}_b$. Let $\mathcal{L} \in \mathbb{R}^{V \times V}$ denote the graph Laplacian, i.e. $\mathcal{L} \stackrel{\text{def}}{=} \boldsymbol{B}^\top \boldsymbol{W} \boldsymbol{B}$. It is easy to see that

$$\boldsymbol{x}^\top \mathcal{L} \boldsymbol{x} = \sum_{u \sim v} \boldsymbol{w}_{u,v} (\boldsymbol{x}_u - \boldsymbol{x}_v)^2 \tag{2.1}$$

for any $\boldsymbol{x} \in \mathbb{R}^n$. Here are three important graph related problems:

Shortest Path Problem: The shortest path problem requires us to find a path $\boldsymbol{f}$ between two vertices that minimizes the sum of the lengths $\boldsymbol{l}$ on its constituent edges. Let s be the starting point and t be the ending point. This problem can be written as

$$\min_{\boldsymbol{B}^\top \boldsymbol{f}=\boldsymbol{\chi}} \sum_e \boldsymbol{w}_e^{-1} |\boldsymbol{f}_e|$$

where $\boldsymbol{\chi} = \mathbf{1}_s - \mathbf{1}_t$ and $\boldsymbol{w}_e = 1/\boldsymbol{l}_e$. We say $\boldsymbol{f} \in \mathbb{R}^E$ meets the demands $\boldsymbol{\chi}$ if $\boldsymbol{B}^\top \boldsymbol{f} = \boldsymbol{\chi}$.

Laplacian Systems: Given a graph with weights $\boldsymbol{w}$ and a vector $\boldsymbol{\chi} \in \mathbb{R}^E$ with $\sum_e \boldsymbol{\chi}_e = 0$, our goal is to find $\boldsymbol{x}$ such that $\boldsymbol{\mathcal{L}}\boldsymbol{x} = \boldsymbol{\chi}$. Setting $\boldsymbol{f} = \boldsymbol{W}\boldsymbol{B}\boldsymbol{x}$, this problem can be written as

$$\min_{\boldsymbol{B}^\top \boldsymbol{f}=\boldsymbol{\chi}} \sum_e \boldsymbol{w}_e^{-1} \boldsymbol{f}_e^2.$$

Undirected Maximum Flow Problem: Given a graph with capacities $\boldsymbol{w}$, the maximum flow problem requires us to find a flow $\boldsymbol{f}$ that routes as much flow as possible from a source vertex s to a sink vertex t while sending at most $\boldsymbol{w}_e$ units of flow over each edge e. Rescaling $\boldsymbol{f}$, this problem can be written as

$$\min_{\boldsymbol{B}^\top \boldsymbol{f}=\boldsymbol{\chi}} \max_e \boldsymbol{w}_e^{-1} |\boldsymbol{f}_e|$$

where $\boldsymbol{\chi} = \mathbf{1}_s - \mathbf{1}_t$.

Interestingly, these three problems respectively require us to find a flow that satisfies a certain demand and minimizes ℓ_1, ℓ_2, or ℓ_∞ norm.

2.2 Thesis Organization

This thesis presents a thorough study of convex optimization and their applications to combinatorial optimization. In figure 2-1, we show the relationship between the chapters. Although the ideas used and the questions addressed in each chapter are all directly or indirectly related, we present the results in each chapter in a modular way so that they may be read in any order. We now discuss the results of each chapter in further details.

2.2.1 Part I: Sparsification

In this part, we study several structural convex problems where the convex functions involve some tall matrices $\boldsymbol{A}$. Our goal is to find general techniques to reduce problems involve tall-and-skinny matrices $\boldsymbol{A}$ to problems involve only square matrices.

2.2.1.1 Chapter 3: ℓ_2 Regression In Input Sparsity Time

In this chapter, we study the ℓ_2 regression problem $\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2$ for tall matrices $\boldsymbol{A} \in \mathbb{R}^{m \times n}$. In general, a small, manageable set of rows of $\boldsymbol{A}$ can be randomly selected to approximate a tall-and-skinny data matrix, improving processing time significantly. In particular, if we sample each row with probability proportional to its importance (called *leverage score*), $O(n \log n)$ rows suffices to approximate $\boldsymbol{A}^\top \boldsymbol{A}$ (Lemma 2.3.3). Unfortunately, the previous best approach to compute leverage scores involves solving some linear systems of the form $\boldsymbol{A}^\top \boldsymbol{A}$ (Lemma 2.3.4) and hence this is as difficult as the original problem. A simple alternative is to sample rows uniformly at random. While this often works, uniform sampling will eliminate critical row information for many natural instances.

We consider uniform sampling from a new persepective by examining what information it *does* preserve. Specifically, we show that uniform sampling yields a matrix that, in some sense, well ap-

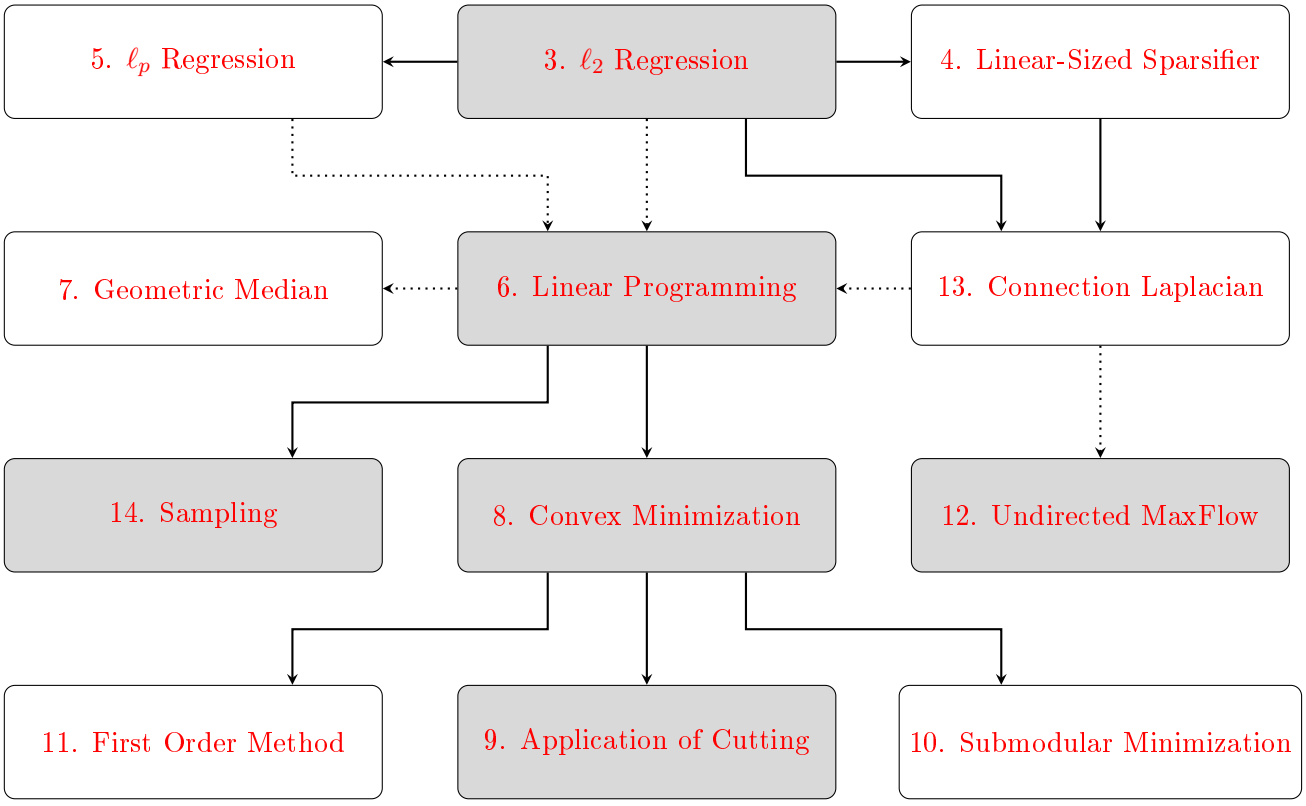


Figure 2-1: This flow chart shows how the chapters are related between each other. A solid arrow indicates a direct usage of the result and a dotted arrow indicates a vague relation. The gray box indicates the key results in this thesis.

proximates a large fraction of the original matrix. While this weak form of approximation is not enough to solve linear regression directly, it *is* enough to compute a better approximation. This observation leads to simple iterative row sampling algorithms for matrix approximation that preserves row structure and sparsity at all intermediate steps. If it takes $\mathcal{T}(m,n)$ time to solve a ℓ_2 regression problem with m rows and n columns, our result can be viewed as a reduction showing that

$$\mathcal{T}(m,n) = \widetilde{O}(\text{nnz}(\boldsymbol{A})) + \widetilde{O}\left(\mathcal{T}(O(n\log n), n)\right).$$

2.2.1.2 Chapter 4: Linear-Sized Sparsifier In Almost-Linear Time

Naturally, one may ask if $\Omega(n\log n)$ rows is necessary for spectral approximations. In a seminal work of Batson, Spielman, Srivastava, they showed that

Theorem 2.2.1 ([31]). *For any matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, there is a diagonal matrix $\boldsymbol{D}$ with $O(n/\varepsilon^2)$ non-zeros such that*

$$(1-\varepsilon)\boldsymbol{A}^\top\boldsymbol{A} \preceq \boldsymbol{A}^\top\boldsymbol{D}\boldsymbol{A} \preceq (1+\varepsilon)\boldsymbol{A}^\top\boldsymbol{A}.$$

However, all previous constructions of linear-sized spectral sparsification require solving $\widetilde{\Omega}(n)$ linear systems. In comparison, the sampling approaches use $O(n\log n/\varepsilon^2)$ rows, but the construction only requires solving $\widetilde{O}(1)$ linear systems. In this chapter, we give a much better trade-off for getting linear-sized spectral sparsifiers. For any integer ρ, we present an algorithm that uses $O(\rho n/\varepsilon^2)$ rows via solving $\tilde{O}(n^{1/\rho})$ linear systems. Hence, this gives a smooth trade off between sparsity and running time. A key component in our algorithm is a novel combination of two techniques used in literature for constructing spectral sparsification: random sampling by leverage scores and adaptive constructions based on barrier functions. Similar to the previous chapter, one can view this result as a reduction showing that

$$\mathcal{T}(m,n) = O(\text{nnz}(\boldsymbol{A})n^{o(1)}) + \mathcal{T}(O(n),n).$$

We believe the term $n^{o(1)}$ can be improved to $\log^{O(1)} n$ and we left it as an open problem.

2.2.1.3 Chapter 4: ℓ_p Regression and John Ellipsoid

Similar to the ℓ_2 regression, it is known that if we sample each row with certain probability (called ℓ_p Lewis weight), one can construct a matrix $\widetilde{\boldsymbol{A}}$ such that, for any $\boldsymbol{x}$, $\|\widetilde{\boldsymbol{A}}\boldsymbol{x}\|_p \approx \|\boldsymbol{A}\boldsymbol{x}\|_p$. This is a useful primitive for solving ℓ_p regression problem. For $1 \le p < 4$, Cohen and Peng showed how to compute ℓ_p Lewis weight by solving $\widetilde{O}(1)$ linear systems [56] and they used it to construct an efficient approximation algorithm for ℓ_p regression. However, for $p \ge 4$, the current best algorithm for computing ℓ_p Lewis weight involves applying a general algorithm for convex problems.

In this chapter, we give a simple algorithm for ℓ_p Lewis weight for $p \ge 2$. Similar to [56], the running time of our algorithm is dominated by solving $\widetilde{O}(1)$ linear systems. Then, we discuss the relation between ℓ_∞ Lewis weight and John ellipsoid, the largest volume ellipsoid inside a given convex set. Finally, we show how to approximate John ellipsoid of symmetric polytopes by solving $\widetilde{O}(1)$ linear systems again.

2.2.1.4 Chapter 6: Faster Linear Programming

In this chapter, we show how to use leverage score and John ellipsoid to solve linear programs using only $\widetilde{O}(\sqrt{n}L)$ iterations where $\boldsymbol{A}$ is the constraint matrix of a linear program with m constraints, n variables, and bit complexity L. Each iteration of our method consists of solving $\widetilde{O}(1)$ linear systems and additional nearly-linear time computation. Our method improves upon the previous best iteration

bounds by a factor of $\widetilde{\Omega}((m/n)^{1/4})$ for methods with polynomial time computable iterations [253] and by a factor of $\widetilde{\Omega}((m/n)^{1/2})$ for methods which solve at most $\widetilde{O}(1)$ linear systems in each iteration achieved over 20 years ago [227]. This result shows that one can "sparsify" the linear program via John ellipsoid and make sure that the complexity of solving a linear program mainly depends on the number of variables n instead of the number of constraints m.

Applying our techniques to the maximum flow yields an $\widetilde{O}(m\sqrt{n}\log^2 U)$ time algorithm for solving the maximum flow problem on directed graphs with m edges, n vertices, and capacity ratio U. This improves upon the previous fastest running time of $O(m\min\{m^{1/2}, n^{2/3}\}\log\left(n^2/m\right)\log(U))$ achieved over 15 years ago by Goldberg and Rao [104] and improves upon the previous best running times for solving dense directed unit capacity graphs of $O(m\min\{m^{1/2}, n^{2/3}\})$ achieved by Even and Tarjan [85] over 35 years ago and a running time of $\widetilde{O}(m^{10/7})$ achieved recently by Mądry [183].

2.2.1.5 Chapter 7: Geometric Median In Nearly-Linear Time

From the results above, we see that a wide range of problems can be sparsified and the running time of solving a convex optimization problem mainly depends on the number of variables. In this chapter, we would like to argue it sometimes depends on the number of true "dimensions", which can be much smaller. For example, linear programs can be solved efficiently when the constraint matrix $\boldsymbol{A}$ is an identity matrix. Then, it is natural to ask if we can solve a linear program faster if the constraint matrix of the linear program can be divided into different blocks and all blocks are identity matrices. Unfortunately, we do not have such result yet.

In this chapter, we illustrate this belief via the geometric median problem: given n points in $\mathbb{R}^d$ compute a point that minimizes the sum of Euclidean distances to the points. This is one of the oldest non-trivial problems in computational geometry yet despite an abundance of research the previous fastest algorithms for computing a $(1+\varepsilon)$-approximate geometric median were $O(d\cdot n^{4/3}\varepsilon^{-8/3})$ by Chin et. al [50], $\tilde{O}\left(d\exp\left(\varepsilon^{-4}\log\varepsilon^{-1}\right)\right)$ by Badoiu et. al [25], $O(nd+\mathrm{poly}(d,\varepsilon^{-1})$ by Feldman and Langberg [86], and $O((nd)^{O(1)}\log\frac{1}{\varepsilon})$ by Parrilo and Sturmfels [222] and Xue and Ye [271].

Different from all previous approaches, we rely on the fact that the Hessian of the objective function can be approximated by an identity matrix minus a rank 1 matrix, namely, the number of true "dimensions" of this problem is 2. This allows us to develop a non-standard interior point method that computes a $(1+\varepsilon)$-approximate geometric median in time $O(nd\log^3\frac{n}{\varepsilon})$.[1] Our result is one of very few cases we are aware of outperforming traditional interior point theory and the only we are aware of using interior point methods to obtain a nearly linear time algorithm for a canonical optimization problem that traditionally requires superlinear time.

2.2.2 Part II: Cutting

In this part, we give a faster cutting plane method and show how to apply this cutting plane method efficiently in various problems.

2.2.2.1 Chapter 8: Convex Minimization In Nearly-Cubic Time

The central problem we consider in this chapter is the feasibility problem. We are promised that a set K is contained a box of radius R and a separation oracle, that given a point $\boldsymbol{x}$ in time SO, either outputs that $\boldsymbol{x}$ is in K or outputs a separating hyperplane. We wish to either find a point in K or prove that K does not contain a ball of radius ε.

[1] We also show how to compute a $(1+\varepsilon)$-approximate geometric median in sub-linear time $O(d\varepsilon^{-2})$.

We show how to solve this problem in $O(n\mathrm{SO}\log(nR/\varepsilon) + n^3\log^{O(1)}(nR/\varepsilon))$ time. This is an improvement over the previous best running time of $O(n\mathrm{SO}\log(nR/\varepsilon)+n^{\omega+1}\log(nR/\varepsilon))$ for the current best known bound of $\omega \sim 2.37$ [99]. Our key idea for achieving this running time improvement is a new straightforward technique for providing low variance unbiased estimates for changes in leverage scores that we hope will be of independent interest (See Section 8.4.1). We show how to use this technique along with ideas from [20, 257] and Chapter 6 to decrease the $\widetilde{O}(n^{\omega+1}\log(nR/\varepsilon))$ overhead in the previous fastest algorithm [253].

2.2.2.2 Chapter 9: Effective Use of Cutting Plane Method

In this chapter, we provide two techniques for applying our cutting plane method (and cutting plane methods in general) to optimization problems and provide several applications of these techniques.

The first technique concerns reducing the number of dimensions through duality. For many problems, their dual is significantly simpler than itself (primal). We use semidefinite programming as a concrete example to show how to improve upon the running time for finding both primal and dual solution by using the cutting planes maintained by our cutting plane method.

The second technique concerns how to minimize a linear function over the intersection of convex sets using optimization oracle. We analyze a simple potential function which allows us to bypass the typical reduction between separation and optimization to achieve faster running times. This reduction provides an improvement over the reductions used previously in [110]. Moreover, we show how this technique allows us to achieve improved running times for matroid intersection and minimum cost submodular flow.

2.2.2.3 Chapter 10: Submodular Minimization In Nearly-Cubic Time

In this chapter, we consider the problem of submodular minimization – a fundamental problem in combinatorial optimization with many diverse applications in theoretical computer science, operations research, machine learning and economics. We show that by considering the interplay between the guarantees of our cutting plane algorithm and the primal-dual structure of submodular minimization, we can achieve improved running times in various settings.

First, we show that a direct application of our method yields an improved weakly polynomial time algorithm for submodular minimization. Then, we present a simple geometric argument that submodular function can be solved with $O(n^3\log n)$ oracle calls but with exponential running time. Finally, we show that by further studying the combinatorial structure of submodular minimization and a modification to our cutting plane algorithm we can obtain a fully improved strongly polynomial time algorithm for submodular minimization.

2.2.2.4 Chapter 11: First Order Method vs Cutting Plane Method

Cutting plane methods are often regarded as inefficient in practice because each iteration of these methods is costly and their convergence rate are sometimes independent of how simple the problem is. In this chapter, we obtain a cutting plane method enjoys the following properties

1. Each iteration takes only linear time for simple problems.

2. It converges faster for simpler problems.

3. It is a polynomial time algorithm in the worst case.

We first propose a new method, a combination of gradient descent and the ellipsoid method, that satisfies both properties 1 and 2. This method attains the optimal rate of convergence of Nesterov's accelerated gradient descent [203] and each iteration of this method takes exactly linear time. However, this method is not a polynomial time algorithm if the function is not smooth.

Then, we propose a new framework which leverages several concepts from convex optimization, from standard first-order methods to cutting plane methods. We show how to use our framework to derive a method that satisfies all three properties. We also demonstrate empirically that our new technique compares favorably with state of the art algorithms (such as accelerated gradient descent and BFGS). Therefore, this suggests that cutting plane methods can be as aggressive as first order methods if designed properly.

2.2.3 Part III: Collapsing

In this part, we show how to collapse vertices in the maximum flow problem and the connection Laplacian problem.

2.2.3.1 Chapter 12: Undirected MaxFlow In Almost-Linear Time

Flow problems on trees are simple because we often know how to collapse the leaves on trees while maintaining the solution. Using this fact and various routing and optimization techniques, we introduce a new framework for approximately solving flow problems in capacitated, undirected graphs and apply it to provide asymptotically faster algorithms for the maximum s-t flow and maximum concurrent multicommodity flow problems. For graphs with n vertices and m edges, it allows us to find an ε-approximate maximum s-t flow in time $O(m^{1+o(1)}\varepsilon^{-2})$[2], improving the previous best bound of $\widetilde{O}(mn^{1/3}\text{poly}(1/\varepsilon))$. Applying the same framework in the multicommodity setting solves a maximum concurrent multicommodity flow problem with k commodities in $O(m^{1+o(1)}\varepsilon^{-2}k^2)$ time, improving the existing bound of $\widetilde{O}(m^{4/3}\text{poly}(k,\varepsilon^{-1}))$.

Our algorithms utilize several new technical tools that we believe may be of independent interest:

- We show how to reduce approximate maximum flow and maximum concurrent flow to oblivious routing.
- We define and provide an efficient construction of a new type of *flow sparsifier* that allowing one to transfer flows from the sparse graph back to the original graph.
- We give the first almost-linear-time construction of an $O(m^{o(1)})$-competitive oblivious routing scheme. No previous such algorithms ran in time better than $\widetilde{\Omega}(mn)$. By reducing the running time to almost-linear, our work provides a powerful new primitive for constructing very fast graph algorithms.

2.2.3.2 Chapter 13: Connection Laplacian In Nearly-Linear Time

The idea of collapsing can be extended to general convex problems as follows: Given a convex function $f : \mathbb{R}^n \to \mathbb{R}$. For any $\boldsymbol{x} \in \mathbb{R}^n$, we write $\boldsymbol{x} = (\boldsymbol{y}, \boldsymbol{z})$ where $\boldsymbol{y}$ is the first $n/2$ variables and $\boldsymbol{z}$ is the last $n/2$ variables. We define the collapse of f by

$$g(\boldsymbol{y}) = \min_{\boldsymbol{z} \in \mathbb{R}^{n/2}} f(\boldsymbol{x}, \boldsymbol{z}).$$

[2]We also note that independently, Jonah Sherman produced an almost-linear time algorithm for maximum flow.

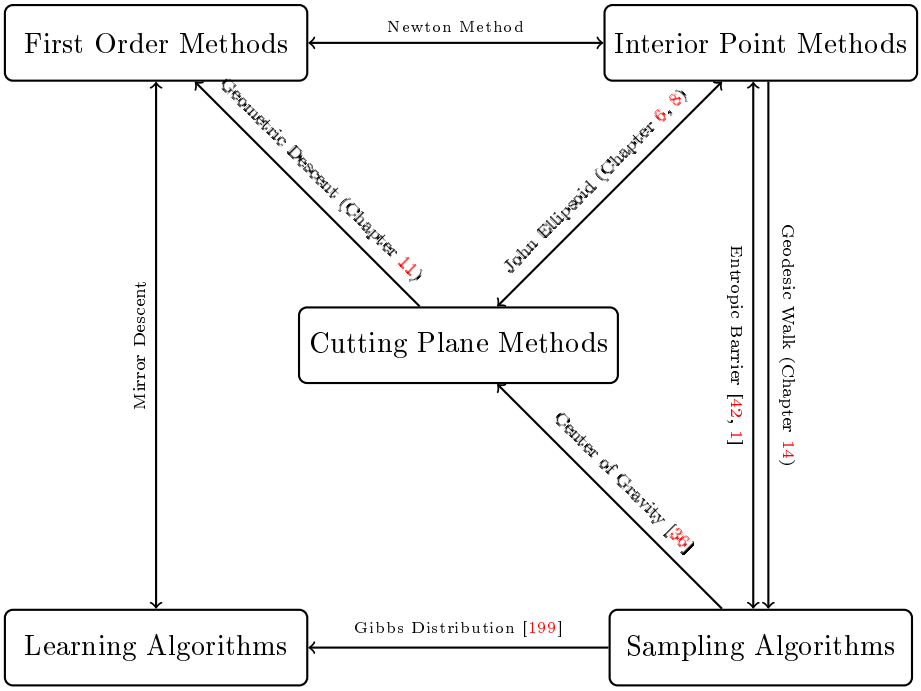


Figure 2-2: Connections between interior point methods, cutting plane methods, first-order methods, learning algorithms for bandit problems, and sampling algorithms for convex sets.

In general, a convex function is difficult to minimize if some variables are intimately related to each other. If not, we can simply solve the problem coordinate-wise. However, if some variables are related and if we can solve the half of the these coordinates, the minimization problem often becomes trivial. Therefore, if we pick half of the variables in random as $\boldsymbol{y}$, g should be easy to compute and this effectively reduce the dimension by half.

In this chapter, we show this is indeed true for connection Laplacian. As a result, we give the first nearly-linear time algorithms for solving systems of equations in connection Laplacians — a generalization of Laplacian matrices that arise in many problems in image and signal processing.

We also prove that every connection Laplacian has a linear sized approximate inverse. This is a LU factorization with a linear number of non-zero entries that is a spectral approximation of the original matrix. Using such a factorization, one can solve systems of equations in a connection Laplacian in linear time. Such a factorization was unknown even for ordinary graph Laplacians.

2.2.4 Part IV: Geometry

In the end of this thesis, we would like to point out a large set of connections between interior point methods, cutting plane methods, first-order methods, learning algorithms for bandit problems, and sampling algorithms for convex sets as they relate to important problems across computer science, operation research, convex optimization and machine learning; see Figure 2-2 for a diagram of work that suggests this relation. One key technique spans all these algorithms is the geometry of convex set. Since these algorithms are widely used across various disciplines, we believe these connections can lead to breakthroughs in different areas. As an example, we only give an application to the sampling problem.

2.2.4.1 Chapter 13: Sampling In Sub-Quadratic Steps

The problems of sampling random points in convex set and computing volume of convex sets is a central problem in Markov chain Monte Carlo and leads to developments in geometric random walks. This problem has been studied for twenty years and the best algorithms take $\tilde{O}(nm)$ iterations for explicit polytopes. We introduce the geodesic walk for sampling Riemannian manifolds and apply it to the problem of generating uniform random points from polytopes. The walk is a discrete-time simulation of a stochastic differential equation (SDE) on the Riemannian manifold equipped with the metric induced by the Hessian of a convex function used for solving linear programming; each step is the solution of an ordinary differential equation (ODE). The resulting sampling algorithm for polytopes mixes in $\tilde{O}(mn^{\frac{3}{4}})$ steps. This is the first walk that breaks the quadratic barrier for mixing in high dimension. We also show that each step of the geodesic walk (solving an ODE) can be implemented efficiently, thus obtaining the current fastest complexity for sampling polytopes.

Advances in algorithms often come from insights from different areas; I believe these new connections will make multifaceted impacts on computation.

2.3 Notation

Here, we define some notations that are used throughout in this thesis.

Variables: We use bold lowercase, e.g $\boldsymbol{x}$, to denote a vector and bold uppercase, e.g. $\boldsymbol{A}$, to denote a matrix. For integers $z \in \mathbb{Z}$ we use $[z] \subseteq \mathbb{Z}$ to denote the set of integers from 1 to z. We let $\mathbf{1}_i$ denote the vector that has value 1 in coordinate i and is 0 elsewhere. We let $\mathbf{1}$ denote the vector that has value 1 in all coordinates. We use $\text{nnz}(\boldsymbol{x})$ or $\text{nnz}(\boldsymbol{A})$ to denote the number of nonzero entries in a vector or a matrix respectively. For a vector, $\boldsymbol{x}$, we let $\|\boldsymbol{x}\|_{\boldsymbol{M}} \stackrel{\text{def}}{=} \sqrt{\boldsymbol{x}^\top \boldsymbol{M}\boldsymbol{x}}$. We use $\mathbb{R}^m_{>0}$ to denote the vectors in $\mathbb{R}^m$ where each coordinate is positive.

Vector Operations: We frequently apply scalar operations to vectors with the interpretation that these operations should be applied coordinate-wise. For example, for vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ we let $\boldsymbol{x}/\boldsymbol{y} \in \mathbb{R}^n$ with $[\boldsymbol{x}/\boldsymbol{y}]_i \stackrel{\text{def}}{=} (x_i/y_i)$ and $\log(\boldsymbol{x}) \in \mathbb{R}^n$ with $[\log(\boldsymbol{x})]_i = \log(x_i)$ for all $i \in [n]$.

Spectral Approximations: We call a symmetric matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ positive semidefinite (PSD) if $\boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x} \geq 0$ for all $\boldsymbol{x} \in \mathbb{R}^n$ and we call $\boldsymbol{A}$ positive definite (PD) if $\boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x} > 0$ for all $\boldsymbol{x} \in \mathbb{R}^n$. For symmetric matrices $\boldsymbol{N}, \boldsymbol{M} \in \mathbb{R}^{n\times n}$, we write $\boldsymbol{N} \preceq \boldsymbol{M}$ to denote that $\boldsymbol{x}^\top \boldsymbol{N}\boldsymbol{x} \leq \boldsymbol{x}^\top \boldsymbol{M}\boldsymbol{x}$ for all $\boldsymbol{x} \in \mathbb{R}^n$, and we define $\boldsymbol{N} \succeq \boldsymbol{M}$, $\boldsymbol{N} \prec \boldsymbol{M}$ and $\boldsymbol{N} \succ \boldsymbol{M}$ analogously. For symmetric matrices, we call $\boldsymbol{B}$ is a ε-spectral approximation of $\boldsymbol{A}$ if $(1-\varepsilon)\boldsymbol{A} \preceq \boldsymbol{B} \preceq (1+\varepsilon)\boldsymbol{A}$. We denote this by $\boldsymbol{A} \approx_\varepsilon \boldsymbol{B}$. In chapter 13, we instead define it as $e^{-\varepsilon}\boldsymbol{A} \preceq \boldsymbol{B} \preceq e^{\varepsilon}\boldsymbol{A}$ for computational simplicity. For nonsymmetric matrices, we call $\boldsymbol{B}$ is a ε-spectral approximation of $\boldsymbol{A}$ if $\boldsymbol{B}^\top \boldsymbol{B}$ is a ε-spectral approximation of $\boldsymbol{A}^\top \boldsymbol{A}$.

Matrix Operations: For $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{n\times m}$, we let $\boldsymbol{A} \circ \boldsymbol{B}$ denote the Schur product, i.e. $[\boldsymbol{A} \circ \boldsymbol{B}]_{ij} \stackrel{\text{def}}{=} \boldsymbol{A}_{ij} \cdot \boldsymbol{B}_{ij}$ for all $i \in [n]$ and $j \in [m]$, and we let $\boldsymbol{A}^{(2)} \stackrel{\text{def}}{=} \boldsymbol{A} \circ \boldsymbol{A}$. For any norm $\|\cdot\|$ and matrix $\boldsymbol{M}$, the operator norm of $\boldsymbol{M}$ is defined by $\|\boldsymbol{M}\| = \sup_{\|\boldsymbol{x}\|=1} \|\boldsymbol{M}\boldsymbol{x}\|$. For any two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ of equal dimensions, let the dot product of $\boldsymbol{A}$ and $\boldsymbol{B}$ defined by $\boldsymbol{A} \bullet \boldsymbol{B} = \text{Tr}\left(\boldsymbol{A}^\top \boldsymbol{B}\right)$. For any matrix $\boldsymbol{M}$, the Frobenius norm of $\boldsymbol{M}$ is defined by $\|\boldsymbol{M}\|_F = \sqrt{\sum_{i,j} \boldsymbol{M}_{ij}^2} = \sqrt{\boldsymbol{M} \bullet \boldsymbol{M}}$.

Spectral Theory: For $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ with rank r, the singular value decomposition $\boldsymbol{A}$ is given by $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ where $\boldsymbol{U} \in \mathbb{R}^{m\times r}$ and $\boldsymbol{V} \in \mathbb{R}^{n\times r}$ have orthonormal columns and $\boldsymbol{\Sigma} \in \mathbb{R}^{r\times r}$ is diagonal. We let $(\boldsymbol{A}^\top \boldsymbol{A})^+$ denote the Moore-Penrose pseudoinverse of $\boldsymbol{A}^\top \boldsymbol{A}$, namely, $(\boldsymbol{A}^\top \boldsymbol{A})^+ = \boldsymbol{V}(\boldsymbol{\Sigma}^{-1})^2\boldsymbol{V}^\top$. For symmetric $\boldsymbol{A}$, the singular value decomposition $\boldsymbol{A}$ becomes $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{U}^\top$ where the diagonal entries of $\boldsymbol{\Sigma}$

are the eigenvalues of $\boldsymbol{A}$. We let $\lambda_{\max}(\boldsymbol{A})$ and $\lambda_{\min}(\boldsymbol{A})$ be the maximum and minimum eigenvalues of $\boldsymbol{A}$. The condition number of matrix $\boldsymbol{A}$ is defined by $\lambda_{\max}(\boldsymbol{A})/\lambda_{\min}(\boldsymbol{A})$.

Diagonal Matrices: For $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ we let $\boldsymbol{diag}(\boldsymbol{A}) \in \mathbb{R}^n$ denote the vector such that $\boldsymbol{diag}(\boldsymbol{A})_i = \boldsymbol{A}_{ii}$ for all $i \in [n]$. For $\boldsymbol{x} \in \mathbb{R}^n$ we let $\boldsymbol{Diag}(\boldsymbol{x}) \in \mathbb{R}^{n\times n}$ be the diagonal matrix such that $\boldsymbol{diag}(\boldsymbol{Diag}(\boldsymbol{x})) = \boldsymbol{x}$. For $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ we let $\boldsymbol{Diag}(\boldsymbol{A})$ be the diagonal matrix such that $\boldsymbol{diag}(\boldsymbol{Diag}(\boldsymbol{A})) = \boldsymbol{diag}(\boldsymbol{A})$. For $\boldsymbol{x} \in \mathbb{R}^n$ when the meaning is clear from context we let $\boldsymbol{X} \in \mathbb{R}^{n\times n}$ denote $\boldsymbol{X} \stackrel{\text{def}}{=} \boldsymbol{Diag}(\boldsymbol{x})$.

Calculus: For $f : \mathbb{R}^n \to \mathbb{R}$ differentiable at $\boldsymbol{x} \in \mathbb{R}^n$, we let $\nabla f(\boldsymbol{x}) \in \mathbb{R}^n$ denote the gradient of f at $\boldsymbol{x}$, i.e. $[\nabla f(\boldsymbol{x})]_i = \frac{\partial}{\partial x_i} f(\boldsymbol{x})$ for all $i \in [n]$. For $f \in \mathbb{R}^n \to \mathbb{R}$ twice differentiable at $\boldsymbol{x} \in \mathbb{R}^n$, we let $\nabla^2 f(\boldsymbol{x})$ denote the Hessian of f at $\boldsymbol{x}$, i.e. $[\nabla f(\boldsymbol{x})]_{ij} = \frac{\partial^2}{\partial x_i \partial x_j} f(\boldsymbol{x})$ for all $i, j \in [n]$. Sometimes, we will consider functions of two vectors, $g : \mathbb{R}^{n_1\times n_2} \to \mathbb{R}$, and wish to compute the gradient and Hessian of g restricted to one of the two vectors. For $\boldsymbol{x} \in \mathbb{R}^{n_1}$ and $\boldsymbol{y} \in \mathbb{R}^{n_2}$ we let $\nabla_{\boldsymbol{x}} \boldsymbol{g}(\boldsymbol{a}, \boldsymbol{b}) \in \mathbb{R}^{n_1}$ denote the gradient of $\boldsymbol{g}$ for fixed $\boldsymbol{y}$ at point $(\boldsymbol{a}, \boldsymbol{b}) \in \mathbb{R}^{n_1\times n_2}$. We define $\nabla_{\boldsymbol{y}}$, $\nabla^2_{\boldsymbol{xx}}$, and $\nabla^2_{\boldsymbol{yy}}$ similarly. For $h : \mathbb{R}^n \to \mathbb{R}^m$ differentiable at $\boldsymbol{x} \in \mathbb{R}^n$ we let $\boldsymbol{J}(\boldsymbol{h}(\boldsymbol{x})) \in \mathbb{R}^{m\times n}$ denote the Jacobian of $\boldsymbol{h}$ at $\boldsymbol{x}$ where for all $i \in [m]$ and $j \in [n]$ we let $[\boldsymbol{J}(\boldsymbol{h}(\boldsymbol{x}))]_{ij} \stackrel{\text{def}}{=} \frac{\partial}{\partial x_j} h(\boldsymbol{x})_i$. For functions of multiple vectors we use subscripts, e.g. $\boldsymbol{J}_{\boldsymbol{x}}$, to denote the Jacobian of the function restricted to the $\boldsymbol{x}$ variable.

Sets: We call a set U *is symmetric* if $\boldsymbol{x} \in \mathbb{R}^n \Leftrightarrow -\boldsymbol{x} \in \mathbb{R}^n$. For any $\alpha > 0$ and set $U \subseteq \mathbb{R}^n$ we let $\alpha U \stackrel{\text{def}}{=} \{\boldsymbol{x} \in \mathbb{R}^n | \alpha^{-1}\boldsymbol{x} \in U\}$. For any $p \in [1, \infty]$ and $r > 0$ we refer to the set $B_p(r) \stackrel{\text{def}}{=} \{\boldsymbol{x} \in \mathbb{R}^n | \|\boldsymbol{x}\|_p \leq r\}$ as *the ℓ_p ball of radius r*. For brevity, we refer to $B_2(r)$ as a *ball of radius r* and $B_\infty(r)$ as a *box of radius r*.

Matrix Multiplication Constant: We let $\omega < 2.3728639$ [100] denote the matrix multiplication constant. It is known that the product a $m \times n$ matrix with a $n \times r$ matrix can be computed in $mnr \min(m, n, r)^{\omega-3+o(1)}$ time and the inverse of a $n \times n$ matrix can be computed in $n^{\omega+o(1)}$ time. For simplicity, we always write $\omega + o(1)$ as ω.

Running Time: Unless mentioned specifically, we will disregard the issue of rounding error if the algorithm is obviously stable. Most of the algorithms we proposed are randomized algorithms that succeeds with high probability and the running time is often the expected running time. For any function f, we write $\widetilde{O}(f) \triangleq O(f \cdot \log^{O(1)} f)$. Suppose m is the length of the input and n is the number of variables, we ssay that an algorithm runs in nearly-linear time if its running time is $\widetilde{O}(m)$, runs in almost linear time if its running time is $m^{1+o(1)}$, and runs in input sparsity time if its running time is $\widetilde{O}(m + n^{O(1)})$.

2.3.1 Sparsification and Leverage Score

In this section, we define spectral approximation and leverage score. They are two key concepts in this thesis, especially in Part I. First, we define leverage score which is a measure of importance. Given a matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$. The leverage score of the i^{th} row $\boldsymbol{a}_i$ of $\boldsymbol{A}$ is:

$$\boldsymbol{\sigma}_i(\boldsymbol{A}) \stackrel{\text{def}}{=} \boldsymbol{a}_i^\top (\boldsymbol{A}^\top \boldsymbol{A})^+ \boldsymbol{a}_i.$$

We also define the related *cross leverage score* as $\boldsymbol{\sigma}_{ij}(\boldsymbol{A}) \stackrel{\text{def}}{=} \boldsymbol{a}_i^\top (\boldsymbol{A}^\top \boldsymbol{A})^+ \boldsymbol{a}_j$. Let $\boldsymbol{\sigma}(\boldsymbol{A})$ be a vector containing $\boldsymbol{A}$'s m leverage scores. $\boldsymbol{\sigma}(\boldsymbol{A})$ is the diagonal of $\boldsymbol{A}(\boldsymbol{A}^\top \boldsymbol{A})^+ \boldsymbol{A}^\top$, which is a projection matrix. Thus, $\boldsymbol{\sigma}_i(\boldsymbol{A}) = \boldsymbol{1}_i^\top \boldsymbol{A}(\boldsymbol{A}^\top \boldsymbol{A})^+ \boldsymbol{A}^\top \boldsymbol{1}_i \leq 1$. Furthermore, since $\boldsymbol{A}(\boldsymbol{A}^\top \boldsymbol{A})^+ \boldsymbol{A}^\top$ is a projection matrix, the sum of $\boldsymbol{A}$'s leverage scores is equal to the matrix's rank:

$$\sum_{i=1}^m \boldsymbol{\sigma}_i(\boldsymbol{A}) = \text{Tr}(\boldsymbol{A}(\boldsymbol{A}^\top \boldsymbol{A})^+ \boldsymbol{A}^\top) = \text{rank}(\boldsymbol{A}(\boldsymbol{A}^\top \boldsymbol{A})^+ \boldsymbol{A}^\top) = \text{rank}(\boldsymbol{A}) \leq n. \tag{2.2}$$

A row's leverage score measures how important it is in composing the row space of $\boldsymbol{A}$. If a row has a component orthogonal to all other rows, its leverage score is 1. Removing it would decrease the rank of $\boldsymbol{A}$, completely changing its row space. The *coherence* of $\boldsymbol{A}$ is $\|\boldsymbol{\sigma}(\boldsymbol{A})\|_\infty$. If $\boldsymbol{A}$ has low coherence, no particular row is especially important. If $\boldsymbol{A}$ has high coherence, it contains at least one row whose removal would significantly affect the composition of $\boldsymbol{A}$'s row space. Two characterizations that helps with this intuition follows:

Lemma 2.3.1. *For all $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ and $i \in [m]$ we have that*

$$\boldsymbol{\sigma}_i(\boldsymbol{A}) = \min_{\boldsymbol{A}^\top \boldsymbol{x} = \boldsymbol{a}_i} \|\boldsymbol{x}\|_2^2$$

where $\boldsymbol{a}_i$ is the i^{th} row of $\boldsymbol{A}$. Let $\boldsymbol{x}_i$ denote the optimal $\boldsymbol{x}$ for $\boldsymbol{a}_i$. The j^{th} entry of $\boldsymbol{x}_i$ is given by $\boldsymbol{\sigma}_{ij}(\boldsymbol{A})$.

Lemma 2.3.2. *For all $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ and $i \in [m]$, we have that $\boldsymbol{\sigma}_i(\boldsymbol{A})$ is the smallest t such that*

$$\boldsymbol{a}_i \boldsymbol{a}_i^\top \preceq t \cdot \boldsymbol{A}^\top \boldsymbol{A}. \tag{2.3}$$

Sampling rows from $\boldsymbol{A}$ according to their exact leverage scores gives a spectral approximation for $\boldsymbol{A}$ with high probability. Sampling by leverage score overestimates also suffices. The following lemma is a corollary of matrix concentration result [251, Cor 5.2] and the inequality (2.3).

Lemma 2.3.3. *Given an error parameter $0 < \varepsilon < 1$ and a vector $\boldsymbol{u}$ of leverage score overestimates, i.e., $\boldsymbol{\sigma}_i(\boldsymbol{A}) \leq \boldsymbol{u}_i$ for all i. For any large enough universal constant c, we define a probability distribution $\boldsymbol{p}_i = \min\{1, c\varepsilon^{-2}\boldsymbol{u}_i \log n\}$. Let $\boldsymbol{S}$ be the diagonal matrix with independently chosen entries. $\boldsymbol{S}_{ii} = \frac{1}{\sqrt{\boldsymbol{p}_i}}$ with probability $\boldsymbol{p}_i$ and 0 otherwise. Then, with probability at least $1 - n^{-\Theta(c)}$, we have*

$$(1-\varepsilon)\boldsymbol{A}^\top \boldsymbol{A} \preceq \boldsymbol{A}^\top \boldsymbol{S}^2 \boldsymbol{A} \preceq (1+\varepsilon)\boldsymbol{A}^\top \boldsymbol{A}.$$

Although (2.2) shows that $\|\boldsymbol{\sigma}\|_1$ is small, computing $\boldsymbol{\sigma}$ exactly is too expensive for many purposes. In [242], they showed that we can compute leverage scores, $\boldsymbol{\sigma}$, approximately by solving only polylogarithmically many regression problems. This result uses the fact that

$$\boldsymbol{\sigma}_i(\boldsymbol{A}) = \|\boldsymbol{A}(\boldsymbol{A}^\top \boldsymbol{A})^+ \boldsymbol{A}^\top \boldsymbol{1}_i\|_2$$

and that by the Johnson-Lindenstrauss Lemma these lengths are persevered up to multiplicative error if we project these vectors onto certain random low dimensional subspaces.

Lemma 2.3.4 ([242])**.** *Let $\mathcal{T}$ be the time to apply $(\boldsymbol{A}^\top \boldsymbol{A})^+$ to an arbitrary vector. Then, for any $0 < \theta < 1$, it is possible to compute an estimate of $\sigma(\boldsymbol{A})$, $\boldsymbol{\sigma}^{(apx)}$, in time $O(\theta^{-1}(\mathcal{T} + \text{nnz}(\boldsymbol{A})))$ such that w.h.p. in n, for all i ,*

$$d^{-\max(\sqrt{\theta/\log n}, \theta)} \boldsymbol{\sigma}_i(\boldsymbol{A}) \leq \boldsymbol{\sigma}_i^{(apx)} \leq d^{\max(\sqrt{\theta/\log n}, \theta)} \boldsymbol{\sigma}_i(\boldsymbol{A}).$$

Setting $\theta = \frac{\varepsilon^2}{\log n}$ gives $(1+\varepsilon)$ factor leverage score approximations in $\widetilde{O}((\mathcal{T} + \text{nnz}(\boldsymbol{A}))/\varepsilon^2)$ time. Therefore, approximating leverage scores is as simple as solving $\widetilde{O}(1)$ linear systems.

2.3.2 Separation Oracle

In this section, we define separation oracles. They are the key concepts in Part II. Our definitions are possibly non-standard and chosen to handle the different settings that occur in this part.

Definition 2.3.5 (Separation Oracle for a Set)**.** Given a set $K \subset \mathbb{R}^n$ and $\delta \geq 0$, a *δ-separation oracle* for K is a function on $\mathbb{R}^n$ such that for any input $\boldsymbol{x} \in \mathbb{R}^n$, it either outputs "successful" or a half space of the form $H = \{\boldsymbol{z} : \boldsymbol{c}^T\boldsymbol{z} \leq \boldsymbol{c}^T\boldsymbol{x} + b\} \supseteq K$ with $b \leq \delta\|\boldsymbol{c}\|_2$ and $\boldsymbol{c} \neq \boldsymbol{0}$. We let $\mathrm{SO}_\delta(K)$ be the time complexity of this oracle.

The parameter δ indicates the accuracy of the oracle. For brevity we refer to a 0-separation oracle for a set as just a *separation oracle*.

Note that in Definition 2.3.5 we do not assume that K is convex. However, we remark that there is a separation oracle for a set if and only if it is convex and that there is a δ separation oracle if and only if the set is close to convex in some sense.

Definition 2.3.6 (Separation Oracle for a Function)**.** For any function f, $\eta \geq 0$ and $\delta \geq 0$, a (η, δ)-separation oracle on a set Γ for f is a function on $\mathbb{R}^n$ such that for any input $\boldsymbol{x} \in \Gamma$, it either asserts $f(\boldsymbol{x}) \leq \min_{\boldsymbol{y}\in\Gamma} f(\boldsymbol{y}) + \eta$ or outputs a half space H such that

$$\{\boldsymbol{z} \in \Gamma : f(\boldsymbol{z}) \leq f(\boldsymbol{x})\} \subset H \stackrel{\text{def}}{=} \{\boldsymbol{z} : \boldsymbol{c}^T\boldsymbol{z} \leq \boldsymbol{c}^T\boldsymbol{x} + b\} \tag{2.4}$$

with $b \leq \delta\|\boldsymbol{c}\|$ and $\boldsymbol{c} \neq \boldsymbol{0}$. We let $\mathrm{SO}_{\eta,\delta}(f)$ be the time complexity of this oracle.

2.3.3 Other Basic Facts

Here, we state some lemmas that are used throughout this thesis.

Lemma 2.3.7 (Sherman-Morrison Formula, [193, Thm 3])**.** *Let $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, and $\boldsymbol{u}, \boldsymbol{v} \perp \ker(\boldsymbol{A})$. Suppose that $1 + \boldsymbol{v}^\top\boldsymbol{A}^+\boldsymbol{u} \neq 0$. Then, it holds that*

$$(\boldsymbol{A} + \boldsymbol{u}\boldsymbol{v}^\top)^+ = \boldsymbol{A}^+ - \frac{\boldsymbol{A}^+\boldsymbol{u}\boldsymbol{v}^T\boldsymbol{A}^+}{1 + \boldsymbol{v}^T\boldsymbol{A}^+\boldsymbol{u}}.$$

Lemma 2.3.8. *Let $\boldsymbol{B}_t$ be a differentiable family of invertible matrix. Then, we have*

1. $\frac{d}{dt}\ln\det(\boldsymbol{B}_t) = \mathrm{Tr}\left(\boldsymbol{B}_t^{-1}\frac{d}{dt}\boldsymbol{B}_t\right)$.
2. $\frac{d}{dt}\boldsymbol{B}_t^{-1} = -\boldsymbol{B}_t^{-1}\left(\frac{d}{dt}\boldsymbol{B}_t\right)\boldsymbol{B}_t^{-1}$.

Lemma 2.3.9. *Let $\boldsymbol{f} : \mathbb{R}^{n+m} \to \mathbb{R}^m$ be a continuously differentiable function. Suppose that $f(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{0}$ for some $\boldsymbol{x} \in \mathbb{R}^m$ and $\boldsymbol{y} \in \mathbb{R}^n$ and that $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}(\boldsymbol{x}, \boldsymbol{y})$ is invertible. Then, there is an open set U containing $\boldsymbol{x}$, an open set V containing $\boldsymbol{y}$, and a unique continuously differentiable function $\boldsymbol{g} : U \to V$ such that $f(\boldsymbol{x}, \boldsymbol{g}(\boldsymbol{x})) = \boldsymbol{0}$. Also, we have that*

$$\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}}(\boldsymbol{x}) = -\left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}(\boldsymbol{x}, \boldsymbol{g}(\boldsymbol{x}))\right)^{-1}\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}(\boldsymbol{x}, \boldsymbol{g}(\boldsymbol{x})).$$

Lemma 2.3.10 (Matrix Chernoff Bound, [251, Cor 5.2])**.** *Let $\{\boldsymbol{X}_k\}$ be a finite sequence of independent, random, and self-adjoint matrices with dimension n. Assume that each random matrix satisfies $\boldsymbol{X}_k \succeq \boldsymbol{0}$, and $\lambda_{\max}(\boldsymbol{X}_k) \leq R$. Let $\mu_{\max} \geq \lambda_{\max}\left(\sum_k \mathbf{E}\boldsymbol{X}_k\right)$ and $\mu_{\min} \leq \lambda_{\min}\left(\sum_k \mathbf{E}\boldsymbol{X}_k\right)$. Then, it holds that*

$$\Pr\left[\lambda_{\max}\left(\sum_k \boldsymbol{X}_k\right) \geq (1+\delta)\mu_{\max}\right] \leq n \cdot \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^{\mu_{\max}/R} \quad \text{for any } \delta \geq 0, \text{ and}$$

$$\Pr\left[\lambda_{\min}\left(\sum_k \boldsymbol{X}_k\right) \leq (1-\delta)\mu_{\min}\right] \leq n \cdot \left(\frac{e^{-\delta}}{(1-\delta)^{1-\delta}}\right)^{\mu_{\min}/R} \quad \text{for any } \delta \in [0, 1]$$

Lemma 2.3.11 (Lieb Thirring Inequality, [171]). *Let $\boldsymbol{A}$ and $\boldsymbol{B}$ be positive definite matrices and $q \geq 1$. Then it holds that*

$$\mathrm{Tr}(\boldsymbol{BAB})^q \leq \mathrm{Tr}(\boldsymbol{B}^q\boldsymbol{A}^q\boldsymbol{B}^q).$$

Theorem 2.3.12 (Simple Constrained Minimization for Twice Differentiable Function [206]). *Let $\boldsymbol{H}$ be a positive definite matrix, K be a convex set and $L \geq \mu \geq 0$. Let $f(\boldsymbol{x}) : K \to \mathbb{R}$ be a twice differentiable function such that $\mu\boldsymbol{H} \preceq \nabla^2 f(\boldsymbol{x}) \preceq L\boldsymbol{H}$ for all $\boldsymbol{x} \in K$. If for some $\boldsymbol{x}^{(0)} \in K$ and all $k \geq 0$, we apply the update rule*

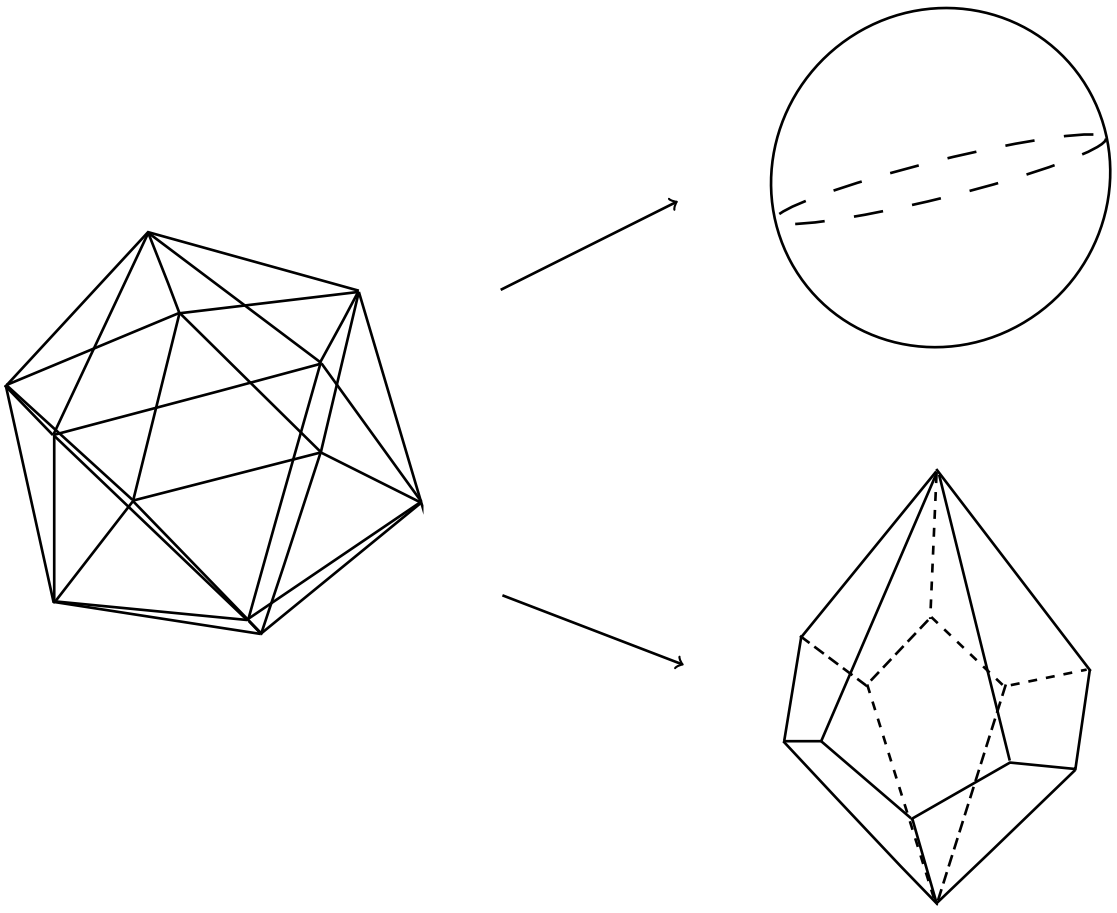
$$\boldsymbol{x}^{(k+1)} = \arg\min_{\boldsymbol{x}\in K} \left\langle \nabla f(\boldsymbol{x}^{(k)}), \boldsymbol{x} - \boldsymbol{x}^{(k)} \right\rangle + \frac{L}{2}\|\boldsymbol{x} - \boldsymbol{x}^{(k)}\|_{\boldsymbol{H}}^2$$

then for all $k \geq 0$ we have

$$\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^*\|_{\boldsymbol{H}}^2 \leq \left(1 - \frac{\mu}{L}\right)^k \|\boldsymbol{x}^{(0)} - \boldsymbol{x}^*\|_{\boldsymbol{H}}^2.$$

Part I

Sparsification



Chapter 3

ℓ_2 Regression In Input Sparsity Time

3.1 Introduction

Many convex optimization problems can be written in the form of

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \sum_{i=1}^{m} f_i(\boldsymbol{a}_i^\top \boldsymbol{x})$$

where f_i are 1 dimension convex functions and $\boldsymbol{a}_i \in \mathbb{R}^n$. The goal of this part is to show that the running time for solving this type of problems, in general, should depends on m linearly.

In this chapter, we first study the simplest case, f_i being quadratic functions. In this case, the problem becomes the ℓ_2 regression problem $\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|$. Currently, there are two types of methods on this problem for the case $m \gg n$.

The first type of methods rely on *spectral approximation*. For a tall, narrow data matrix $\boldsymbol{A}$, these methods find a shorter approximate data matrix, $\widetilde{\boldsymbol{A}}$, such that, for all vectors $\boldsymbol{x}$, $\|\widetilde{\boldsymbol{A}}\boldsymbol{x}\|_2 \approx \|\boldsymbol{A}\boldsymbol{x}\|_2$. All of which rely on variations of Johnson-Lindenstrauss random projections for constructing $\widetilde{\boldsymbol{A}}$ [53, 185, 200, 170]. Such random projections do not maintain sparsity and row structure of $\boldsymbol{A}$. The second type of methods rely on uniform (or non-adaptive) sampling, such as stochastic descent. These methods works only for *low coherence* data and hence are less useful for theory development.

By re-examining uniform sampling, we give spectral approximation algorithms that avoid random projection entirely. Our methods are the first to match state-of-the-art runtimes while preserving row structure and sparsity in all matrix operations. Since it preserves row structure, this allows us to take advantage of the problem structure. Let $\mathcal{T}(m,n)$ be the time needed to solve a ℓ_2 regression problem $\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|$ where $\boldsymbol{A}$ is a $m\times n$ matrix and each rows of $\boldsymbol{A}$ comes from a certain row dictionary. Roughly speaking, our method shows that

$$\mathcal{T}(m,n) = \widetilde{O}(\mathrm{nnz}(\boldsymbol{A})) + \widetilde{O}(\mathcal{T}(\widetilde{O}(n),n)).$$

This powerful reduction allows us to develop the first nearly linear time algorithm for connection Laplacian (Chapter 13). In comparison, the random projection methods only shows that $\mathcal{T}(m,n) = \widetilde{O}(\mathrm{nnz}(\boldsymbol{A}) + n^\omega)$.

As we explained in Section 2.3.1, it is known that for a data matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, a spectral approximation can be obtained by sampling $O(n\log n)$ rows, each with probability proportional to its *statistical leverage score*. Recall that the leverage score of $\boldsymbol{A}$'s i^{th} row, $\boldsymbol{a}_i$, is $\boldsymbol{\sigma}_i = \boldsymbol{a}_i^\top(\boldsymbol{A}^\top\boldsymbol{A})^+\boldsymbol{a}_i$, where $\boldsymbol{M}^+$ denotes the Moore-Penrose pseudoinverse of $\boldsymbol{M}$. A higher leverage score indicates that $\boldsymbol{a}_i$ is more important in composing the spectrum of $\boldsymbol{A}$.

Unfortunately, leverage scores are difficult to calculate – finding them involves computing the pseudoinverse $(\boldsymbol{A}^\top\boldsymbol{A})^+$, which is as slow as solving our regression problem in the first place. In practice, data is often assumed to have *low coherence* [195], in which case simply selecting rows uniformly at random works [22, 156]. However, uniform sampling could be disastrous — if $\boldsymbol{A}$ contains

a row with some component orthogonal to all other rows, removing it will reduce the rank of $\boldsymbol{A}$ and thus we cannot possibly preserve all vector products ($\|\widetilde{\boldsymbol{A}}\boldsymbol{x}\|_2$ will start sending some vectors to 0). Any uniform sampling scheme is likely to drop any such single row. On the other hand, when leverage score sampling, such a row would have the highest possible leverage score of 1.

Possible fixes include randomly "mixing" data points to avoid degeneracies [22]. However, this approach sacrifices sparsity and structure in our data matrix, increasing storage and runtime costs. Is there a more elegant fix? First note that sampling $\boldsymbol{A}$ by *approximate* leverage scores is fine, but we may need to select more than the optimal $O(n \log n)$ rows. With that in mind, consider the following straightforward algorithm for iterative sampling, inspired by [170]:

1. **Reduce $\boldsymbol{A}$ significantly by sampling rows uniformly.**
2. **Use the smaller matrix to approximate $(\boldsymbol{A}^\top \boldsymbol{A})^+$ and leverage scores for $\boldsymbol{A}$.**
3. **Resample rows of $\boldsymbol{A}$ using these estimates to obtain a spectral approximation $\widetilde{\boldsymbol{A}}$.**

While intuitive, this scheme was not previously known to work. Our main result is proving that it does. This process will quickly converge on a small spectral approximation to $\boldsymbol{A}$, i.e. with $O(n \log n)$ rows.

A few results come close to an analysis of such a routine – in particular, two iterative sampling schemes are analyzed in [170]. However, the first ultimately requires Johnson-Lindenstrauss projections that mix rows, something we were hoping to avoid. The second almost maintains sparsity and row structure (except for possibly including rows of the identity in $\widetilde{\boldsymbol{A}}$), but its convergence rate depends on the condition number of $\boldsymbol{A}$.

More importantly, both of these results are similar in that they rely on the primitive that a (possibly poor) spectral approximation to $\boldsymbol{A}$ is sufficient for approximately computing leverage scores, which are in turn good enough for obtaining an even better spectral approximation. As mentioned, uniform sampling will not in general give a spectral approximation – it does not preserve information about all singular values. Our key contribution is a better understanding of what information uniform sampling *does* preserve. It turns out that, although weaker than a spectral approximation, the matrix obtained from uniform sampling can nonetheless give leverage score estimates that are good enough to obtain increasingly better approximations to $\boldsymbol{A}$.

3.1.1 Our Approach

Suppose we compute a set of leverage score estimates, $\{\tilde{\boldsymbol{\sigma}}_i\}$, using $(\widetilde{\boldsymbol{A}}^\top \widetilde{\boldsymbol{A}})^+$ in place of $(\boldsymbol{A}^\top \boldsymbol{A})^+$ for some already obtained matrix approximation $\widetilde{\boldsymbol{A}}$. As long as our leverage score approximations are *upper bounds* on the true scores ($\tilde{\boldsymbol{\sigma}}_i \geq \boldsymbol{\sigma}_i$) we can use them for sampling and still obtain a spectral approximation to $\boldsymbol{A}$. The number of samples we take will be

$$c \cdot \log n \cdot \sum_{i=1}^{m} \tilde{\boldsymbol{\sigma}}_i$$

where c is some fixed constant. When sampling by exact leverage scores, it can be shown that $\sum_i \boldsymbol{\sigma}_i \leq n$ so we take $c \cdot n \log n$ rows.

Thus, to prove that our proposed iterative algorithm works, we need to show that, if we uniformly sample a relatively small number of rows from $\boldsymbol{A}$ (Step 1) and estimate leverage scores using these rows (Step 2), then the sum of our estimates will be small. Then, when we sample by these estimated leverage scores in Step 3, we can sufficiently reduce the size of $\boldsymbol{A}$.

In prior work, the sum of overestimates was bounded by estimating *each* leverage score to within a multiplicative factor. This requires a spectral approximation, which is why previous iterative sampling schemes could only boost poor spectral approximations to better spectral approximations. Of course, a "for each" statement is not required, and we will not get one through uniform sampling. Thus, our core result avoids this technique. Specifically, we show,

Theorem 3.1.1. *For any k, we can select k rows uniformly at random from $\boldsymbol{A}$ to obtain $\widetilde{\boldsymbol{A}}$. Let $\{\tilde{\boldsymbol{\sigma}}_i\}$ be a set of leverage score estimates for $\boldsymbol{A}$ computed using $\widetilde{\boldsymbol{A}}$[1]. Then, we have that $\tilde{\boldsymbol{\sigma}}_i \geq \boldsymbol{\sigma}_i$ for all i and*

$$\mathbf{E}\left[\sum_{i=1}^{m} \tilde{\boldsymbol{\sigma}}_i\right] \leq \frac{mn}{k}.$$

The validity of our proposed iterative sampling scheme immediately follows from Theorem 3.1.1. For example, if we uniformly sample $k = m/2$ rows then $c \log n \sum \tilde{\boldsymbol{\sigma}}_i \leq O(n \log n)$, so we can cut our matrix down to $O(n \log n)$ rows. There is a convenient tradeoff – the more rows uniformly sampled in Step 1, the more we can cut $\boldsymbol{A}$ down by in Step 3. This tradeoff leads to natural recursive and iterative algorithms for row sampling.

We give a proof of Theorem 3.1.1 using a simple expectation argument that bounds $\mathbf{E}\sum \tilde{\boldsymbol{\sigma}}_i$. We also prove alternative versions of Theorem 3.1.1 with slightly different guarantees (Theorems 3.6.1) using a technique that we believe is of independent interest. It is well known that, if $\boldsymbol{A}$ has low coherence – that is, has a low maximum leverage score – then uniform sampling from the matrix is actually sufficient for obtaining a full spectral approximation. The uniform rate will upper bound the leverage score rate for every row. With this in mind, we show a powerful fact: while many matrices do not have low coherence, for any $\boldsymbol{A}$, we can decrease the weight on a small subset of rows to make the matrix have low coherence. Specifically,

Lemma 3.1.2 (Coherence Reducing Reweighting)**.** *For any $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and any coherence upper bound $\alpha > 0$, there exists a diagonal reweighting matrix $\boldsymbol{W} \in \mathbb{R}^{m \times m}$ with all entries in $[0, 1]$ and just (n/α) entries not equal to 1, such that:*

$$\forall i,\ \boldsymbol{\sigma}_i(\boldsymbol{W}\boldsymbol{A}) \leq \alpha.$$

Intuitively, this lemma shows that uniform sampling gives a matrix that spectrally approximates a large sub-matrix of the original data. It follows from our more general Theorem 3.5.1, which describes exactly how leverage scores of $\boldsymbol{A}$ can be manipulated through row reweighting.

We never actually find $\boldsymbol{W}$ explicitly – simply its existence implies our uniform sampling theorems. As explained, since $\boldsymbol{W}\boldsymbol{A}$ has low coherence, uniform sampling would give a spectral approximation to the reweighted matrix and thus a multiplicatively good approximation to *each* leverage score. Thus, the sum of estimated leverage scores for $\boldsymbol{W}\boldsymbol{A}$ will be low, i.e. $< O(n)$. It can be shown that, for any row that is not reweighted, the leverage score in $\boldsymbol{A}$ computed using a uniformly sampled $\widetilde{\boldsymbol{A}}$, is never greater than the corresponding leverage score in $\boldsymbol{W}\boldsymbol{A}$ computed using a uniformly sampled $\widetilde{\boldsymbol{W}\boldsymbol{A}}$. Thus, the sum of approximate leverage scores for rows in $\boldsymbol{A}$ that are not reweighted is small by comparison to their corresponding leverage scores in $\boldsymbol{W}\boldsymbol{A}$. How about the rows that *are* reweighted in $\boldsymbol{W}\boldsymbol{A}$? Lemma 3.1.2 claims there are not too many of these – we can trivially bound their leverage score estimates by 1 and even then the total sum of estimated leverage scores will be small.

This argument gives the result we need: even if a uniformly sampled $\widetilde{\boldsymbol{A}}$ cannot be used to obtain good per row leverage score upper bounds, it is sufficient for ensuring that the sum of all leverage score estimates is not too high.

[1] We decribe exactly how each $\tilde{\sigma}_i$ is computed when we prove Theorem 3.1.1 in Section 3.4.

3.2 Previous Work

3.2.1 Randomized Numerical Linear Algebra

In the past decade, fast randomized algorithms for matrix problems have risen to prominence. Numerous results give improved running times for matrix multiplication, linear regression, and low rank approximation – helpful surveys of this work include [184] and [112]. In addition to asymptotic runtime gains, randomized alternatives to standard linear algebra tools tend to offer significant gains in terms of data access patterns and required working memory.

Algorithms for randomized linear algebra often work by generically reducing problem size – large matrices are compressed (using randomness) to smaller approximations which are processed deterministically via standard linear algebraic methods. Methods for matrix reduction divide roughly into two categories – random projection methods [54, 53, 185, 200, 229] and sampling methods [70, 71, 72, 74, 73, 170].

Random projection methods recombine rows or columns from a large matrix to form a much smaller problem that approximates the original. Descending from the Johnson-Lindenstrauss Lemma [130] and related results, these algorithms are impressive for their simplicity and speed – reducing a large matrix simply requires multiplication by an appropriately chosen random matrix.

Sampling methods, on the other hand, seek to approximate large matrices by judiciously selecting (and reweighting) few rows or columns. Sampling itself is even simpler and faster than random projection – the challenge becomes efficiently computing the correct measure of "importance" for rows or columns. More important rows or columns are selected with higher probability.

3.2.2 Approximate Linear Regression

We focus on linear regression, i.e. solving overdetermined systems, which requires our matrix reduction step to produce a spectral approximation $\widetilde{\boldsymbol{A}}$ to the data matrix $\boldsymbol{A}$. One possibility is to obtain a ε-approximation with $O(n \log n/\varepsilon^2)$ rows and to solve regression on the smaller problem to give an approximate solution. To improve stability and achieve $\log(1/\varepsilon)$ dependence, randomized schemes can be combined with known iterative regression algorithms. These methods only require a constant factor spectral approximation with $O(n \log n)$ rows [22, 55, 53, 192, 228].

When random projections are used, $\widetilde{\boldsymbol{A}} = \boldsymbol{\Pi}\boldsymbol{A}$ for some randomly generated matrix $\boldsymbol{\Pi}$ which is known as a *subspace embedding*. Recent progress has significantly sped up the process of computing $\boldsymbol{\Pi}\boldsymbol{A}$, leading to the first *input-sparsity time* algorithms for linear regression (or nearly input-sparsity time if iterative methods are employed) [53, 185, 200].

3.2.3 Row Sampling

An alternative route to spectral approximation is importance sampling. Specifically, $O(n \log n/\varepsilon^2)$ rows can be sampled with probability proportional to their leverage scores [75, 76, 242]. In [242], Spielman and Srivastava specifically focus on spectral approximations for the edge-vertex incidence matrix of a graph. This is more commonly referred to as *spectral graph sparsification*, a primitive that has become important in research on graph algorithms. Each row in a graph's (potentially tall) edge-vertex incident matrix corresponds to an edge and the row's leverage score is exactly the edge's *weighted effective resistance*, which is used as the sampling probability in [242].

This application illustrates an important point: for spectral sparsification, it is critical that $\boldsymbol{A}$ is compressed via sampling instead of random projection. Sampling ensures that $\widetilde{\boldsymbol{A}}$ contains only reweighted rows from $\boldsymbol{A}$, so it remains an edge-vertex incidence matrix. In general, sampling is

interesting because it preserves row structure. Even if that structure is just a certain level of sparsity, it can reduce memory requirements and accelerate matrix operations.

While leverage scores for the edge-vertex incidence matrix of a graph can be computed quickly [147, 243], in general, computing leverage scores requires evaluating $(\boldsymbol{A}^\top \boldsymbol{A})^+$, which is as difficult as solving regression in the first place. Li, Miller, and Peng address this issue with methods for iteratively computing good row samples [170]. Their algorithms achieve input-sparsity time regression, but are fairly involved and rely on intermediate operations that ultimately require Johnson-Lindenstrauss projections, mixing rows and necessitating dense matrix operations. An alternative approach from [170] does preserve row structure (except for possible additions of rows from the identity to intermediate matrices) but converges in a number of steps that depends on $\boldsymbol{A}$'s condition number.

3.3 Generalized Leverage Score

We often approximate the leverage scores of $\boldsymbol{A}$ by computing them with respect to some other matrix $\boldsymbol{B} \in \mathbb{R}^{m' \times n}$. We define the *generalized leverage score*:

$$\sigma_i^{\boldsymbol{B}}(\boldsymbol{A}) \stackrel{\text{def}}{=} \begin{cases} \boldsymbol{A}_i^\top (\boldsymbol{B}^\top \boldsymbol{B})^+ \boldsymbol{A}_i & \text{if } \boldsymbol{A}_i \perp \ker(\boldsymbol{B}), \\ \infty & \text{otherwise.} \end{cases} \tag{3.1}$$

If $\boldsymbol{A}_i$ has an component in $\ker(\boldsymbol{B})$, we set its generalized leverage score to ∞, since it might be the only row in $\boldsymbol{A}$ pointing in this direction. Thus, when sampling rows, we cannot remove it. We could set the generalized leverage score to 1, but using ∞ simplifies notation in some of our proofs. If $\boldsymbol{B}$ is a spectral approximation for $\boldsymbol{A}$, then every generalized leverage score is a good multiplicative approximation to its corresponding true leverage score:

Lemma 3.3.1 ([170, Lem 4.3]). *If* $\frac{1}{\lambda}\boldsymbol{A}^\top \boldsymbol{A} \preceq \boldsymbol{B}^\top \boldsymbol{B} \preceq \boldsymbol{A}^\top \boldsymbol{A}$, *then* $\sigma_i(\boldsymbol{A}) \leq \sigma_i^{\boldsymbol{B}}(\boldsymbol{A}) \leq \lambda \cdot \sigma_i(\boldsymbol{A})$.

3.4 Expectation Bound on Leverage Score Estimation

In this section, we use a simple expectation argument to prove Theorem 3.1.1, which is restated in full below:

Theorem 3.1.1 (Full Statement). Given any $\boldsymbol{A} \in \mathbb{R}^{m \times n}$. Let S denote a uniformly random sample of k rows from $\boldsymbol{A}$ and let $\boldsymbol{S} \in \mathbb{R}^{m \times m}$ be its diagonal indicator matrix (i.e. $\boldsymbol{S}_{ii} = 1$ for $i \in S$, $\boldsymbol{S}_{ii} = 0$ otherwise). Define

$$\tilde{\boldsymbol{\sigma}}_i \stackrel{\text{def}}{=} \begin{cases} \boldsymbol{\sigma}_i^{\boldsymbol{SA}}(\boldsymbol{A}) & \text{if } i \in S, \\ \frac{1}{1 + \frac{1}{\boldsymbol{\sigma}_i^{\boldsymbol{SA}}(\boldsymbol{A})}} & \text{otherwise.} \end{cases}$$

Then, $\tilde{\boldsymbol{\sigma}}_i \geq \boldsymbol{\sigma}_i(\boldsymbol{A})$ for all i and

$$\mathbf{E}\left[\sum_{i=1}^{m} \tilde{\boldsymbol{\sigma}}_i\right] \leq \frac{mn}{k}.$$

Proof. First we show that our estimates are valid leverage score upper bounds, i.e. $\tilde{\boldsymbol{\sigma}}_i \geq \boldsymbol{\sigma}_i(\boldsymbol{A})$. Let $\boldsymbol{S}^{(i)}$ be the diagonal indicator matrix for $S \cup \{i\}$. We claim that, for all i,

$$\tilde{\boldsymbol{\sigma}}_i = \boldsymbol{\sigma}_i^{\boldsymbol{S}^{(i)}\boldsymbol{A}}(\boldsymbol{A}). \tag{3.2}$$

This is proved case-by-case:

When $i \in S$, $\boldsymbol{S} = \boldsymbol{S}^{(i)}$ so it holds trivially.
When $i \notin S$ and $\boldsymbol{a}_i \not\perp \ker(\boldsymbol{SA})$, then by definition, $\boldsymbol{\sigma}_i^{\boldsymbol{SA}}(\boldsymbol{A}) = \infty$ and $\tilde{\boldsymbol{\sigma}}_i = \frac{1}{1+\frac{1}{\infty}} = 1 = \boldsymbol{\sigma}_i^{\boldsymbol{S}^{(i)}\boldsymbol{A}}(\boldsymbol{A})$.
When $i \notin S$ and $\boldsymbol{a}_i \perp \ker(\boldsymbol{SA})$ then by the Sherman-Morrison formula (Lemma 2.3.7),

$$\begin{aligned}
\boldsymbol{\sigma}_i^{\boldsymbol{S}^{(i)}\boldsymbol{A}}(\boldsymbol{A}) &= \boldsymbol{a}_i^\top \left(\boldsymbol{A}^\top \boldsymbol{S}^2 \boldsymbol{A} + \boldsymbol{a}_i \boldsymbol{a}_i^\top\right)^+ \boldsymbol{a}_i \\
&= \boldsymbol{a}_i^\top \left(\left(\boldsymbol{A}^\top \boldsymbol{S}^2 \boldsymbol{A}\right)^+ - \frac{\left(\boldsymbol{A}^\top \boldsymbol{S}^2 \boldsymbol{A}\right)^+ \boldsymbol{a}_i \boldsymbol{a}_i^\top \left(\boldsymbol{A}^\top \boldsymbol{S}^2 \boldsymbol{A}\right)^+}{1 + \boldsymbol{a}_i^\top \left(\boldsymbol{A}^\top \boldsymbol{S}^2 \boldsymbol{A}\right)^+ \boldsymbol{a}_i}\right) \boldsymbol{a}_i \\
&= \boldsymbol{\sigma}_i^{\boldsymbol{SA}}(\boldsymbol{A}) - \frac{\boldsymbol{\sigma}_i^{\boldsymbol{SA}}(\boldsymbol{A})^2}{1 + \boldsymbol{\sigma}_i^{\boldsymbol{SA}}(\boldsymbol{A})} = \frac{1}{1 + \frac{1}{\boldsymbol{\sigma}_i^{\boldsymbol{SA}}(\boldsymbol{A})}} = \tilde{\boldsymbol{\sigma}}_i.
\end{aligned}$$

By (3.2) and the fact that $\boldsymbol{A}^\top {\boldsymbol{S}^{(i)}}^2 \boldsymbol{A} \preceq \boldsymbol{A}^\top \boldsymbol{A}$ (see Lemma 3.3.1), we have $\tilde{\boldsymbol{\sigma}}_i = \boldsymbol{\sigma}_i^{\boldsymbol{S}^{(i)}\boldsymbol{A}}(\boldsymbol{A}) \geq \boldsymbol{\sigma}_i(\boldsymbol{A})$, so our estimates are upper bounds as desired. It remains to upper bound the expected sum of $\tilde{\boldsymbol{\sigma}}_i$. We can break down the sum as:

$$\sum_{i=1}^m \tilde{\boldsymbol{\sigma}}_i = \sum_{i \in S} \tilde{\boldsymbol{\sigma}}_i + \sum_{i \notin S} \tilde{\boldsymbol{\sigma}}_i.$$

The first term is simply the sum of $\boldsymbol{SA}$'s leverage scores, so it is equal to $\operatorname{rank}(\boldsymbol{SA}) \leq n$ by (2.2). To bound the second term, consider a random process that first selects $\boldsymbol{S}$, then selects a random row $i \notin S$ and returns $\tilde{\boldsymbol{\sigma}}_i$. There are always exactly $m - k$ rows $\notin S$, so the value returned by this random process is, in expectation, exactly equal to $\frac{1}{m-k} \cdot \mathbf{E} \sum_{i \notin S} \tilde{\boldsymbol{\sigma}}_i$.

This random process is *also* equivalent to randomly selecting a set S' of $k+1$ rows, then randomly choosing a row $i \in S'$ and returning its leverage score! In expectation it is therefore equal to the average leverage score in $\boldsymbol{S'A}$. $\boldsymbol{S'A}$ has $k+1$ rows and its leverage scores sum to its rank, so we can bound its average leverage score by $\frac{n}{k+1}$. Overall:

$$\mathbf{E}\left[\sum_{i=1}^m \tilde{\boldsymbol{\sigma}}_i\right] \leq n + (m-k) \cdot \frac{n}{k+1} \leq \frac{n(m+1)}{k+1} \leq \frac{mn}{k}.$$

□

3.5 Coherence-Reducing Weighting

In this section, we prove Theorem 3.5.1, which shows that we can reweight a small number of rows in any matrix $\boldsymbol{A}$ to make it have low coherence. This structural result may be of independent interest. It is also fundamental in proving Theorem 3.6.1, a slightly stronger version of Theorem 3.1.1 that we will prove in Section 3.6.

Actually, for Theorem 3.5.1 we prove a more general statement, studying how to select a diagonal row reweighting matrix $\boldsymbol{W}$ to arbitrarily control the leverage scores of $\boldsymbol{WA}$. One simple conjecture would be that, given a vector $\boldsymbol{u}$, there always exists a $\boldsymbol{W}$ such that $\boldsymbol{\sigma}_i(\boldsymbol{WA}) = \boldsymbol{u}_i$. This conjecture is unfortunately not true - if $\boldsymbol{A}$ is the identity matrix, then $\boldsymbol{\sigma}_i(\boldsymbol{WA}) = 0$ if $\boldsymbol{W}_{ii} = 0$ and $\boldsymbol{\sigma}_i(\boldsymbol{WA}) = 1$ otherwise. Instead, we show the following:

Theorem 3.5.1. *For any $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and any vector $\boldsymbol{u} \in \mathbb{R}^m$ with $\boldsymbol{u}_i > 0$ for all i, there exists a diagonal matrix $\boldsymbol{W} \in \mathbb{R}^{m \times m}$ with $\mathbf{0} \preceq \boldsymbol{W} \preceq \boldsymbol{I}$ such that:*

$$\forall i,\ \boldsymbol{\sigma}_i(\boldsymbol{WA}) \leq \boldsymbol{u}_i, \tag{3.3}$$

and

$$\sum_{i:\boldsymbol{W}_{ii}\neq 1} \boldsymbol{u}_i \leq n. \tag{3.4}$$

Note that (3.3) is easy to satisfy – it holds if we set $\boldsymbol{W} = \boldsymbol{0}$. Hence, the main result is the second claim . Not only does a $\boldsymbol{W}$ exist that gives the desired leverage score bounds, but it is only necessary to reweight rows in $\boldsymbol{A}$ with a low total weight in terms of $\boldsymbol{u}$.

For any incoherence parameter α, if we set $\boldsymbol{u}_i = \alpha$ for all i, then this theorem shows the existence of a reweighting that reduces coherence to α. Such a reweighting has $\sum_{i:\boldsymbol{W}_{ii}\neq 1} \alpha \leq n$ and therefore $|\{i : \boldsymbol{W}_{ii} \neq 1\}| \leq \frac{n}{\alpha}$. So, we see that Lemma 3.1.2 follows as a special case of Theorem 3.5.1.

In order to prove Theorem 3.5.1, we first give two technical lemmas which are proved Section 3.8. Lemma 3.5.2 describes how the leverage scores of $\boldsymbol{A}$ evolve when a single row of $\boldsymbol{A}$ is reweighted. We show that, when we decrease the weight of a row, that row's leverage score decreases and the leverage score of all other rows increases.

Lemma 3.5.2 (Leverage Score Changes Under Rank 1 Updates)**.** *Given any $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, $\gamma \in (0,1)$, and $i \in [m]$, let $\boldsymbol{W}$ be a diagonal matrix such that $\boldsymbol{W}_{ii} = \sqrt{1-\gamma}$ and $\boldsymbol{W}_{jj} = 1$ for all $j \neq i$. Then,*

$$\boldsymbol{\sigma}_i(\boldsymbol{W}\boldsymbol{A}) = \frac{(1-\gamma)\,\boldsymbol{\sigma}_i(\boldsymbol{A})}{1-\gamma\boldsymbol{\sigma}_i(\boldsymbol{A})} \leq \boldsymbol{\sigma}_i(\boldsymbol{A}),$$

and for all $j \neq i$,

$$\boldsymbol{\sigma}_j(\boldsymbol{W}\boldsymbol{A}) = \boldsymbol{\sigma}_j(\boldsymbol{A}) + \frac{\gamma\boldsymbol{\sigma}_{ij}(\boldsymbol{A})^2}{1-\gamma\boldsymbol{\sigma}_i(\boldsymbol{A})} \geq \boldsymbol{\sigma}_j(\boldsymbol{A}).$$

Next we claim that, with respect to weightings of $\boldsymbol{A}$'s rows, leverage scores are lower semi-continuous.

Lemma 3.5.3 (Leverage Scores are Lower Semi-continuous)**.** *$\boldsymbol{\sigma}(\boldsymbol{W}\boldsymbol{A})$ is lower semi-continuous in the diagonal matrix $\boldsymbol{W}$, i.e. for any sequence $\boldsymbol{W}^{(k)} \to \overline{\boldsymbol{W}}$ with $\boldsymbol{W}^{(k)}_{ii} \geq 0$ for all k and i, we have*

$$\boldsymbol{\sigma}_i\left(\overline{\boldsymbol{W}}\boldsymbol{A}\right) \leq \liminf_{k\to\infty} \boldsymbol{\sigma}_i\left(\boldsymbol{W}^{(k)}\boldsymbol{A}\right).$$

With Lemmas 3.5.2 and 3.5.3 in place, we are ready to prove the main reweighting theorem.

Proof of Theorem 3.5.1. We prove the existence of the required $\boldsymbol{W}$ by considering the limit of the Algorithm 1 for computing a reweighting matrix.

For all $k \geq 0$, let $\boldsymbol{W}^{(k)}$ be the value of $\boldsymbol{W}$ after the k^{th} update to the weight. We show that $\overline{\boldsymbol{W}} = \lim_{k\to\infty} \boldsymbol{W}^{(k)}$ meets the conditions of Theorem 3.5.1. First note that Algorithm 1 is well defined and that all entries of $\boldsymbol{W}^{(k)}$ are non-negative for all $k \geq 0$. To see this, suppose we need to decrease $\boldsymbol{W}^{(k)}_{ii}$ so that $\boldsymbol{\sigma}_i(\boldsymbol{W}^{(k+1)}\boldsymbol{A}) = \boldsymbol{u}_i$. Note that the condition $\boldsymbol{\sigma}_i(\boldsymbol{W}^{(k)}\boldsymbol{A}) < 1$ gives

$$\lim_{\gamma\to 1} \frac{(1-\gamma)\,\boldsymbol{\sigma}_i\left(\boldsymbol{W}^{(k)}\boldsymbol{A}\right)}{1-\gamma\boldsymbol{\sigma}_i\left(\boldsymbol{W}^{(k)}\boldsymbol{A}\right)} = 0.$$

Therefore, Lemma 3.5.2 shows that we can make $\boldsymbol{\sigma}_i(\boldsymbol{W}^{(k+1)}\boldsymbol{A})$ arbitrary small by setting γ close enough to 1. Since the leverage score for row i is continuous, this implies that $\boldsymbol{W}^{(k+1)}$ exists as desired.

Algorithm 1: `ComputeReweighting` (a.k.a the whack-a-mole algorithm)

Initialize $\boldsymbol{W} = \boldsymbol{I}$.
while ***true*** **do**
  for $i = 1$ ***to*** n **do**
    if $\boldsymbol{\sigma}_i(\boldsymbol{WA}) \geq \boldsymbol{u}_i$ **then**
      if $\boldsymbol{\sigma}_i(\boldsymbol{WA}) < 1$ **then**
        Decrease $\boldsymbol{W}_{ii}$ so that $\boldsymbol{\sigma}_i(\boldsymbol{WA}) = \boldsymbol{u}_i$.
      else
        Set $\boldsymbol{W}_{ii} = 0$.
      end
    end
  end
end
Output: $\boldsymbol{W}$.

Since, the entries of $\boldsymbol{W}^{(k)}$ are non-negative and decrease monotonically by construction, clearly $\overline{\boldsymbol{W}}$ exists. Furthermore, since setting $\boldsymbol{W}_{ii} = 0$ makes $\boldsymbol{\sigma}_i(\boldsymbol{WA}) = 0$, we see that, by construction,

$$\liminf_{k\to\infty} \boldsymbol{\sigma}_i\left(\boldsymbol{W}^{(k)}\boldsymbol{A}\right) \leq \boldsymbol{u}_i \text{ for all } i \in [n].$$

Therefore, by Lemma 3.5.3 we have that $\boldsymbol{\sigma}_i(\overline{\boldsymbol{W}}\boldsymbol{A}) \leq u_i$.

It only remains to show that $\sum_{i:\overline{\boldsymbol{W}}_{ii}\neq 1} \boldsymbol{u}_i \leq n$. Let k be the first iteration such that $\boldsymbol{W}_{ii}^{(k)} \neq 1$ for any i such that $\overline{\boldsymbol{W}}_{ii} \neq 1$. Let $S \subseteq [m]$ be the set of rows such that $\boldsymbol{W}_{ii}^{(k)} = 0$ and let $T = \{i : \overline{\boldsymbol{W}}_{ii} \neq 1\} - S$. Since decreasing the weight of one row increases the leverage scores of all other rows, we have

$$\begin{aligned}\sum_{i\in T\cup S} \boldsymbol{u}_i &\leq \sum_{i\in T} \boldsymbol{\sigma}_i\left(\boldsymbol{W}^{(k)}\boldsymbol{A}\right) + \sum_{i\in S} 1 \\ &\leq \operatorname{rank}\left(\boldsymbol{W}^{(k)}\boldsymbol{A}\right) + |S|.\end{aligned}$$

When we set $\boldsymbol{W}_{ii} = 0$, it must be the case that $\boldsymbol{\sigma}_i(\boldsymbol{WA}) = 1$. In this case, removing the i^{th} row decreases the rank of $\boldsymbol{WA}$ by 1 and hence $\operatorname{rank}(\boldsymbol{W}^{(k)}\boldsymbol{A}) \leq n - |S|$. Therefore,

$$\sum_{i:\overline{\boldsymbol{W}}_{ii}\neq 1} \boldsymbol{u}_i = \sum_{i\in T\cup S} \boldsymbol{u}_i \leq n.$$

□

3.6 High Probability Bound on Leverage Score Estimation

Theorem 3.1.1 alone is enough to prove that a variety of iterative methods for spectral approximation work. In particular, we can use Lemma 2.3.3 to show that sampling each row with probability $\Theta\left(\min\{1, \tilde{\boldsymbol{\sigma}}_i \log n\}\right)$ independently with suitable reweighting gives a spectral approximation of the original matrix. Since $\tilde{\boldsymbol{\sigma}}_i$ is a constant approximation of $\min\{1, \boldsymbol{\sigma}_i^{\boldsymbol{SA}}(\boldsymbol{A})\}$, the total number of rows it takes is $\Theta\left(\sum_i \min\{1, \boldsymbol{\sigma}_i^{\boldsymbol{SA}}(\boldsymbol{A}) \log n\}\right)$ which can be bounded by the $\sum_i \min\{1, \boldsymbol{\sigma}_i^{\boldsymbol{SA}}(\boldsymbol{A})\}$ bound proved in Theorem 3.1.1.

In this section, we bound $\sum_i \min\{1, \boldsymbol{\sigma}_i^{\boldsymbol{SA}}(\boldsymbol{A}) \log n\}$ directly and it allows us to bound the quantity with a high probability instead of merely in expectation. The proof relies on Theorem 3.5.1, which

intuitively shows that a large fraction of our matrix $\boldsymbol{A}$ has low coherence. Sampling rows uniformly will give a spectral approximation for this portion of our matrix. Then, since few rows are reweighted in $\boldsymbol{WA}$, even loose upper bounds on the leverage scores for those rows will allow us to bound the total sum of estimated leverage scores when we sample uniformly.

Theorem 3.6.1. *Given any $\boldsymbol{A} \in \mathbb{R}^{m\times n}$. Let S denote a uniformly random sample of k rows from $\boldsymbol{A}$ and let $\boldsymbol{S} \in \mathbb{R}^{m\times m}$ be its diagonal indicator matrix (i.e. $\boldsymbol{S}_{ii} = 1$ for $i \in S$, $\boldsymbol{S}_{ii} = 0$ otherwise). With high probability in n, we have that*

$$\sum_{i=1}^{n} \min\left\{1, \boldsymbol{\sigma}_i^{\boldsymbol{SA}}(\boldsymbol{A}) \log n\right\} \leq O\left(\frac{mn \log n}{k}\right).$$

Resampling from $\boldsymbol{A}$ by these estimates will, with high probability, return a ε-spectral approximation to $\boldsymbol{A}$ with at most $O\left(\frac{mn\log n}{k\varepsilon^2}\right)$ rows.

Proof. Let c be the constant c in Lemma 2.3.3. By Theorem 3.5.1, there is a diagonal matrix $\boldsymbol{W} \preceq \boldsymbol{I}$ such that $\sigma_i(\boldsymbol{WA}) \leq \frac{k}{cm\log n}$ for all i and $\sum_{i:\boldsymbol{W}_{ii}\neq 1} \frac{k}{cm\log n} \leq n$. For this $\boldsymbol{W}$, we have

$$\begin{aligned}
\sum_{i=1}^{n} \min\left\{1, \boldsymbol{\sigma}_i^{\boldsymbol{SA}}(\boldsymbol{A}) \log n\right\} &\leq \sum_{i:\boldsymbol{W}_{ii}\neq 1} 1 + \log n \sum_{i:\boldsymbol{W}_{ii}=1} \boldsymbol{\sigma}_i^{\boldsymbol{SA}}(\boldsymbol{A}) \\
&\leq \frac{cmn\log n}{k} + \log n \sum_{i:\boldsymbol{W}_{ii}=1} \boldsymbol{\sigma}_i^{\boldsymbol{SA}}(\boldsymbol{A}) \\
&= \frac{cmn\log n}{k} + \log n \sum_{i:\boldsymbol{W}_{ii}=1} \boldsymbol{\sigma}_i^{\boldsymbol{SA}}(\boldsymbol{WA}). \qquad (3.5)
\end{aligned}$$

Since $\boldsymbol{\sigma}_i(\boldsymbol{WA}) \leq \frac{k}{cm\log n}$ and $\boldsymbol{S}$ is a uniformly random sample of k rows, Lemma 2.3.3 shows that

$$\frac{1}{2}\boldsymbol{A}^\top \boldsymbol{W}^2 \boldsymbol{A} \preceq \frac{m}{k} \boldsymbol{A}^\top \boldsymbol{W}\boldsymbol{S}^2\boldsymbol{W}\boldsymbol{A}. \qquad (3.6)$$

Hence (3.6) along with Lemma 3.3.1 shows that

$$\boldsymbol{\sigma}_i^{\boldsymbol{SA}}(\boldsymbol{WA}) \leq \frac{2m}{k}\boldsymbol{\sigma}_i(\boldsymbol{WA}).$$

Combining with (3.5), we have

$$\begin{aligned}
\sum_{i=1}^{n} \min\left\{1, \boldsymbol{\sigma}_i^{\boldsymbol{SA}}(\boldsymbol{A}) \log n\right\} &\leq \frac{cmn\log n}{k} + \frac{2m\log n}{k} \sum_{i:\boldsymbol{W}_{ii}=1} \boldsymbol{\sigma}_i(\boldsymbol{WA}) \\
&\leq \frac{cmn\log n}{k} + \frac{2mn\log n}{k} = O\left(\frac{mn\log n}{k}\right).
\end{aligned}$$

The last statement follows from Lemma 3.3.1 with sampling probabilities $\min\left\{1, \frac{c\boldsymbol{\sigma}_i^{\boldsymbol{SA}}(\boldsymbol{A})\log n}{\varepsilon^2}\right\}$. □

3.7 Main Result

As discussed in the introduction, Theorems 3.1.1 and 3.6.1 immediately yield new, extremely simple algorithms for spectral matrix approximation. For clarity, we initially present versions running in *nearly* input-sparsity time. However, we later explain how our first algorithm can be modified with

standard techniques to remove log factors, achieving input-sparsity time and thus matching state-of-the-art results [53, 185, 200]. Our algorithms rely solely on row sampling, which preserves matrix sparsity and structure, possibly improving space usage and runtime for intermediate system solves.

3.7.1 Repeated Halving

The algorithm we present, `RepeatedHalving`, is a simple recursive procedure. We uniformly sample $\frac{m}{2}$ rows from $\boldsymbol{A}$ to obtain $\boldsymbol{A}'$. By Theorems 3.1.1 and 3.6.1, estimating leverage scores of $\boldsymbol{A}$ with respect to this sample allows us to immediately find a spectral approximation to $\boldsymbol{A}$ with $O(n \log n)$ rows. Of course, $\boldsymbol{A}'$ is still large, so computing these estimates would be slow. Thus, we *recursively* find a spectral approximation of $\boldsymbol{A}'$ and use this to compute the estimated leverage scores.

Algorithm 2: `RepeatedHalving`

Input: $m \times n$ matrix $\boldsymbol{A}$
Output: spectral approximation $\widetilde{\boldsymbol{A}}$ consisting of $O(n \log n)$ rescaled rows of $\boldsymbol{A}$
Uniformly sample $\frac{m}{2}$ rows of $\boldsymbol{A}$ to form $\boldsymbol{A}'$.
If $\boldsymbol{A}'$ has $> O(n \log n)$ rows, **recursively** compute a spectral approximation $\widetilde{\boldsymbol{A}}'$ of $\boldsymbol{A}'$.
Compute approximate generalized leverage scores of $\boldsymbol{A}$ w.r.t. $\widetilde{\boldsymbol{A}}'$
Use these estimates to sample rows of $\boldsymbol{A}$ to form $\widetilde{\boldsymbol{A}}$.
Output $\widetilde{\boldsymbol{A}}$.

In analyzing the runtimes of these algorithms, we assume $m = O(\text{poly}(n))$, which is a reasonable assumption for any practical regression problem.[2] Furthermore, we use the fact that a $n \times n$ system can be solved in time n^ω, where ω is the matrix multiplication exponent. However, we emphasize that, depending on the structure and sparsity of $\boldsymbol{A}$, alternative system solving methods may yield faster results or runtimes with different trade offs. For example, if the rows of $\boldsymbol{A}$ are sparse, solving a system in $\widetilde{\boldsymbol{A}}$, where $\widetilde{\boldsymbol{A}}$ consists of $O(n \log n)$ rescaled rows from $\boldsymbol{A}$ may be accelerated by using iterative conjugate gradient, or other Krylov subspace methods (which can also avoid explicitly computing $\widetilde{\boldsymbol{A}}^\top \widetilde{\boldsymbol{A}}$). It is best to think of n^ω as the runtime of the fastest available system solver in your domain, and the quoted runtimes as general guidelines that will change somewhat depending on exactly how the above algorithms are implemented.

First, we give an important primitive showing that estimates of generalized leverage scores can be computed efficiently. Computing exact generalized leverage scores is slow and we only need constant factor approximations, which will only increase our sampling rates and hence number of rows sampled by a constant factor.

Lemma 3.7.1. *Given $\boldsymbol{B}$ containing $O(n \log n)$ rescaled rows of $\boldsymbol{A}$, for any $1 > \theta > 0$, it is possible to compute an estimate of $\boldsymbol{\sigma}^{\boldsymbol{B}}(\boldsymbol{A})$, $\tilde{\boldsymbol{\sigma}}$, in $O(n^\omega \theta^{-1} + \text{nnz}(\boldsymbol{A})\theta^{-1})$ time such that, w.h.p. in n, for all i, $\tilde{\boldsymbol{\sigma}}_i \geq \boldsymbol{\sigma}_i^{\boldsymbol{B}}(\boldsymbol{A})$ and $\tilde{\boldsymbol{\sigma}}_i \leq O(n^\theta)\boldsymbol{\sigma}_i^{\boldsymbol{B}}(\boldsymbol{A})$.*

Setting $\theta = O(\frac{1}{\log n})$ gives a constant factor approximation to generalized leverage score in $O(n^\omega \log n + \text{nnz}(\boldsymbol{A}) \log n)$ time.

Proof. The proof is basically same as Lemma 2.3.4 except that we use fast matrix multiplication to solve the corresponding linear systems and that we need to handle the case $\boldsymbol{a}_i \not\perp \ker(\boldsymbol{B})$. When $\boldsymbol{a}_i \not\perp \ker(\boldsymbol{B})$, its generalized leverage score should be ∞ – see (3.1). So, we need to check whether each $\boldsymbol{a}_i$ has a component in the null-space of $\boldsymbol{B}$. This can be done in a variety of ways. For example, we can choose a random gaussian vector $\boldsymbol{g}$ and compute $\boldsymbol{g} - \boldsymbol{B}(\boldsymbol{B}^\top \boldsymbol{B})^+ \boldsymbol{B}^\top \boldsymbol{g}$, which is the same as

[2] A simple method for handling even larger values of n is outlined in [170].

$\boldsymbol{g} - (\boldsymbol{B}^\top \boldsymbol{B})^+ \boldsymbol{B}^\top \boldsymbol{B}\boldsymbol{g}$. This gives a random vector in the null space of $\boldsymbol{B}$, so computing its dot product with any row $\boldsymbol{a}_i$ will tell us (with probability 1) whether $\boldsymbol{a}_i$ is orthogonal to the null space or not. □

With Lemma 3.7.1 in place, we can analyze the runtime of our algorithms. For simplicity, we give runtimes for computing a constant factor spectral approximation to $\boldsymbol{A}$, which can be used as a preconditioner in iterative regression algorithms [22, 53, 228] or used to compute leverage scores of $\boldsymbol{A}$ up to a constant factor. We could sample $O(n \log n \varepsilon^{-2})$ rows to directly obtain a ε-approximation. By Lemma 3.7.1 the runtime of this final refinement is just $O(\text{nnz}(\boldsymbol{A}) \log n + n^\omega \log n)$.

Lemma 3.7.2. `RepeatedHalving` *(Algorithm 2) runs in time* $O(\text{nnz}(\boldsymbol{A}) \log n + n^\omega \log(m/n) \log n)$, *outputting a matrix with* $\widetilde{\boldsymbol{A}}$ *with* $O(n \log n)$ *rows such that* $\widetilde{\boldsymbol{A}}^\top \widetilde{\boldsymbol{A}} \approx_2 \boldsymbol{A}^\top \boldsymbol{A}$.

Proof. The proof is by induction – it suffices to show that the work at the top level is $O(\text{nnz}(\boldsymbol{A}) \log n + n^\omega \log n)$. At each of the $O(\log(m/n))$ levels of recursion, we cut our matrix in half uniformly so $\text{nnz}(\boldsymbol{A})$ will also be cut approximately in half with high probability.

By Theorem 3.6.1, sampling by $\boldsymbol{\sigma}_i^{\boldsymbol{A}'}(\boldsymbol{A})$ allows us to obtain $\widetilde{\boldsymbol{A}}$ with $O(n \log n)$ rows. If we instead use $\widetilde{\boldsymbol{A}}'$, our estimated leverage scores increase by at most a constant factor (since $\widetilde{\boldsymbol{A}}'$ is a constant factor spectral approximation to $\boldsymbol{A}'$). Furthermore, using Lemma 3.7.1 to approximate generalized leverage scores increases our estimates by another constant factor at most. Overall, $\widetilde{\boldsymbol{A}}$ will have $O(n \log n)$ rows as desired and our runtime at the top level is just the runtime of estimating leverage scores from Lemma 3.7.1 – $O(\text{nnz}(\boldsymbol{A}) \log n + n^\omega \log n)$. □

3.7.2 Achieving Input Sparsity Time

We briefly note that, using techniques from [170], it is possible to remove the $\log n$ factor on the $\text{nnz}(\boldsymbol{A})$ term to achieve true input-sparsity time with `RepeatedHalving`. Instead of using Lemma 3.7.1 to estimate generalized leverage scores from up to a constant factor using $\boldsymbol{A}'$, we only estimate them up to a $O(n^\theta)$ factor for some constant $0 < \theta < \omega - 2$. Using these rough estimates, we obtain $\widetilde{\boldsymbol{A}}$ with $O(n^{1+\theta} \log n)$ rows. Then, for the rows in $\widetilde{\boldsymbol{A}}$, we can again compute generalized leverage scores with respect to $\boldsymbol{A}'$, now up to constant factors, and reduce down to just $O(n \log n)$ rows. In total, each iteration will take time $O(\theta^{-1}(\text{nnz}(\boldsymbol{A}) + n^\omega \log n))$, so obtaining a constant factor approximation to $\boldsymbol{A}$ takes time $O(\theta^{-1}\text{nnz}(\boldsymbol{A}) + \theta^{-1} n^\omega \log^2 n)$. Recall that we assume $m = \text{poly}(n)$, so we have $\log(m/n) = O(\log n)$ iterations.

In order to obtain a ε-spectral approximation with only $O(n \log n \varepsilon^{-2})$ rows, we first obtain a constant factor approximation, $\widetilde{\boldsymbol{A}}$, with $O(n \log n)$ rows. We then use leverage scores estimated with $\widetilde{\boldsymbol{A}}$ to compute a $\varepsilon/2$-approximation to $\boldsymbol{A}$ with $O(n^{1+\theta} \log n \varepsilon^{-2})$ rows. Finally, we again use leverage scores estimated with $\widetilde{\boldsymbol{A}}$ and Lemma 3.7.1 with $\theta = O(\varepsilon^2 / \log n)$ to a compute a $\varepsilon/2$-approximation to this smaller matrix with only $O(n \log n \varepsilon^{-2})$ rows. This takes total time $O(\theta^{-1}\text{nnz}(\boldsymbol{A}) + \theta^{-1} n^\omega \log^2 n + n^{2+\theta} \log^2 n \varepsilon^{-2})$. The $n^{2+\theta} \log^3 n \varepsilon^{-2}$ comes from applying Lemma 3.7.1 to refine our second approximation, which has $O(n^{1+\theta} \log n \varepsilon^{-2})$ rows and thus at most $O(n^{2+\theta} \log n \varepsilon^{-2})$ nonzero entries. Overall, the technique yields:

Lemma 3.7.3. *Given any constant* $0 < \theta \leq \omega - 2$, *and any error* $0 \leq \varepsilon < 1$, *w.h.p. in* n *we can compute a matrix* $\widetilde{\boldsymbol{A}}$ *with* $O(n \log n \varepsilon^{-2})$ *rows such that* $\widetilde{\boldsymbol{A}}^\top \widetilde{\boldsymbol{A}} \approx_{1+\varepsilon} \boldsymbol{A}^\top \boldsymbol{A}$ *in* $O(\theta^{-1}\text{nnz}(\boldsymbol{A}) + \theta^{-1} n^\omega \log^2 n + n^{2+\theta} \varepsilon^{-2})$ *time.*

Proof. As is standard, $\log n$ factors on the $n^{2+\theta}$ term are 'hidden' as we can just slightly increase the value of θ to subsume them. □

3.8 Appendix

3.8.1 Rank 1 Updates

Here we prove Lemma 3.5.2, making critical use of the Sherman-Morrison formula for the Moore-Penrose pseudoinverse (Lemma 2.3.7).

Lemma 3.5.2 (Leverage Score Changes Under Rank 1 Updates). Given any $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, $\gamma \in (0,1)$, and $i \in [m]$, let $\boldsymbol{W}$ be a diagonal matrix such that $\boldsymbol{W}_{ii} = \sqrt{1-\gamma}$ and $\boldsymbol{W}_{jj} = 1$ for all $j \neq i$. Then, we have

$$\boldsymbol{\sigma}_i(\boldsymbol{W}\boldsymbol{A}) = \frac{(1-\gamma)\boldsymbol{\sigma}_i(\boldsymbol{A})}{1-\gamma\boldsymbol{\sigma}_i(\boldsymbol{A})} \leq \boldsymbol{\sigma}_i(\boldsymbol{A}),$$

and for all $j \neq i$,

$$\boldsymbol{\sigma}_j(\boldsymbol{W}\boldsymbol{A}) = \boldsymbol{\sigma}_j(\boldsymbol{A}) + \frac{\gamma\boldsymbol{\sigma}_{ij}(\boldsymbol{A})^2}{1-\gamma\boldsymbol{\sigma}_i(\boldsymbol{A})} \geq \boldsymbol{\sigma}_j(\boldsymbol{A}).$$

Proof.

$$\begin{aligned}
\boldsymbol{\sigma}_i(\boldsymbol{W}\boldsymbol{A}) &= \mathbf{1}_i\boldsymbol{W}\boldsymbol{A}\left(\boldsymbol{A}^\top\boldsymbol{W}^2\boldsymbol{A}\right)^+\boldsymbol{A}^\top\boldsymbol{W}^\top\mathbf{1}_i^\top && \text{(definition of leverage scores)}\\
&= (1-\gamma)\boldsymbol{a}_i^\top\left(\boldsymbol{A}^\top\boldsymbol{A} - \gamma\boldsymbol{a}_i\boldsymbol{a}_i^\top\right)^+\boldsymbol{a}_i && \text{(definition of } \boldsymbol{W}\text{)}\\
&= (1-\gamma)\boldsymbol{a}_i^\top\left((\boldsymbol{A}^\top\boldsymbol{A})^+ + \gamma\frac{(\boldsymbol{A}^\top\boldsymbol{A})^+\boldsymbol{a}_i\boldsymbol{a}_i^\top(\boldsymbol{A}^\top\boldsymbol{A})^+}{1-\gamma\boldsymbol{a}_i^\top(\boldsymbol{A}^\top\boldsymbol{A})^+\boldsymbol{a}_i}\right)\boldsymbol{a}_i && \text{(Sherman-Morrison formula)}\\
&= (1-\gamma)\left(\boldsymbol{\sigma}_i(\boldsymbol{A}) + \frac{\gamma\boldsymbol{\sigma}_i(\boldsymbol{A})^2}{1-\gamma\boldsymbol{\sigma}_i(\boldsymbol{A})}\right)\\
&= \frac{(1-\gamma)\boldsymbol{\sigma}_i(\boldsymbol{A})}{1-\gamma\boldsymbol{\sigma}_i(\boldsymbol{A})} \leq \boldsymbol{\sigma}_i(\boldsymbol{A}).
\end{aligned}$$

Similarly,

$$\begin{aligned}
\boldsymbol{\sigma}_j(\boldsymbol{W}\boldsymbol{A}) &= \boldsymbol{a}_j^\top\left(\boldsymbol{A}^\top\boldsymbol{A} - \gamma\boldsymbol{a}_i\boldsymbol{a}_i^\top\right)^+\boldsymbol{a}_j\\
&= \boldsymbol{a}_j^\top\left((\boldsymbol{A}^\top\boldsymbol{A})^+ + \gamma\frac{(\boldsymbol{A}^\top\boldsymbol{A})^+\boldsymbol{a}_i\boldsymbol{a}_i^\top(\boldsymbol{A}^\top\boldsymbol{A})^+}{1-\gamma\boldsymbol{a}_i^\top(\boldsymbol{A}^\top\boldsymbol{A})^+\boldsymbol{a}_i}\right)\boldsymbol{a}_j\\
&= \boldsymbol{\sigma}_j(\boldsymbol{A}) + \frac{\gamma\boldsymbol{\sigma}_{ij}(\boldsymbol{A})^2}{1-\gamma\boldsymbol{\sigma}_i(\boldsymbol{A})} \geq \boldsymbol{\sigma}_j(\boldsymbol{A}).
\end{aligned}$$

□

3.8.2 Lower Semi-continuity of Leverage Scores

Here we prove Lemma 3.5.3 by providing a fairly general inequality, Lemma 3.8.1, for relating leverage scores under one set of weights to leverage scores under another.

Lemma 3.5.3 (Leverage Scores are Lower Semi-continuous). $\boldsymbol{\sigma}(\boldsymbol{W}\boldsymbol{A})$ is lower semi-continuous in the diagonal matrix $\boldsymbol{W}$, i.e. for any sequence $\boldsymbol{W}^{(k)} \to \overline{\boldsymbol{W}}$ with $\boldsymbol{W}_{ii}^{(k)} \geq 0$ for all k and i, we have

$$\boldsymbol{\sigma}_i(\overline{\boldsymbol{W}}\boldsymbol{A}) \leq \liminf_{k\to\infty}\boldsymbol{\sigma}_i(\boldsymbol{W}^{(k)}\boldsymbol{A}). \tag{3.7}$$

Lemma 3.8.1 (Comparing Leverage Scores). *Let $\boldsymbol{W}, \overline{\boldsymbol{W}} \in \mathbb{R}^{m\times m}$ be non-negative diagonal matrices and suppose that $\boldsymbol{W}_{ii} > 0$ and $\overline{\boldsymbol{W}}_{ii} > 0$ for some $i \in [m]$. Then*

$$\boldsymbol{\sigma}_i(\overline{\boldsymbol{W}}\boldsymbol{A}) \leq \frac{\overline{\boldsymbol{W}}_{ii}^2}{\boldsymbol{W}_{ii}^2}\left(1 + \sqrt{\lambda_{\max}\left(\boldsymbol{A}\left(\boldsymbol{A}^\top\overline{\boldsymbol{W}}^2\boldsymbol{A}\right)^+\boldsymbol{A}^\top\right)}\|\boldsymbol{W} - \overline{\boldsymbol{W}}\|_\infty\right)^2 \boldsymbol{\sigma}_i(\boldsymbol{W}\boldsymbol{A}). \tag{3.8}$$

Proof. Scaling the variables in Lemma 2.3.1 we have that there exists $\boldsymbol{x} \in \mathbb{R}^m$ such that

$$\boldsymbol{A}^\top\boldsymbol{W}\boldsymbol{x} = \boldsymbol{a}_i \quad \text{and} \quad \|\boldsymbol{x}\|_2^2 = \frac{\boldsymbol{\sigma}_i(\boldsymbol{W}\boldsymbol{A})}{\boldsymbol{W}_{ii}^2}. \tag{3.9}$$

Note that $\boldsymbol{A}^\top\left(\boldsymbol{W} - \overline{\boldsymbol{W}}\right)\boldsymbol{x}$ is in the image of $\boldsymbol{A}^\top\overline{\boldsymbol{W}}$ as $\boldsymbol{A}^\top\boldsymbol{W}\boldsymbol{x} = \boldsymbol{a}_i$ and $\overline{\boldsymbol{W}}_{ii} \neq 0$. Consequently,

$$\boldsymbol{A}^\top\overline{\boldsymbol{W}}\boldsymbol{y} = \boldsymbol{A}^\top\left(\boldsymbol{W} - \overline{\boldsymbol{W}}\right)\boldsymbol{x}^{(k)} \quad \text{for} \quad \boldsymbol{y} \stackrel{\text{def}}{=} \overline{\boldsymbol{W}}\boldsymbol{A}\left(\boldsymbol{A}^\top\overline{\boldsymbol{W}}^2\boldsymbol{A}\right)^+\boldsymbol{A}^\top\left(\boldsymbol{W} - \overline{\boldsymbol{W}}\right)\boldsymbol{x}.$$

Since $\boldsymbol{A}^\top\overline{\boldsymbol{W}}(\boldsymbol{x} + \boldsymbol{y}) = \boldsymbol{a}_i$, Lemma 2.3.1 implies

$$\boldsymbol{\sigma}_i(\overline{\boldsymbol{W}}\boldsymbol{A}) \leq \overline{\boldsymbol{W}}_{ii}^2\|\boldsymbol{x} + \boldsymbol{y}\|_2^2. \tag{3.10}$$

We can bound the contribution of $\boldsymbol{y}$ by

$$\begin{aligned}\|\boldsymbol{y}\|_2^2 &\leq \left\|\overline{\boldsymbol{W}}\boldsymbol{A}\left(\boldsymbol{A}^\top\overline{\boldsymbol{W}}^2\boldsymbol{A}\right)^+\boldsymbol{A}^\top\left(\boldsymbol{W} - \overline{\boldsymbol{W}}\right)\boldsymbol{x}\right\|_2^2 \\ &\leq \lambda_{\max}\left(\boldsymbol{A}\left(\boldsymbol{A}^\top\overline{\boldsymbol{W}}^2\boldsymbol{A}\right)^+\boldsymbol{A}^\top\right)\left\|\boldsymbol{W} - \overline{\boldsymbol{W}}\right\|_\infty^2\|\boldsymbol{x}\|_2^2.\end{aligned} \tag{3.11}$$

Applying triangle inequality to (3.9), (3.10), and (3.11) yields the result. □

Proof of Lemma 3.5.3. For any $i \in [n]$ such that $\overline{\boldsymbol{W}}_{ii} = 0$ (3.7) follows trivially from the fact that leverage scores are non-negative. For any $i \in [m]$ such that $\overline{\boldsymbol{W}}_{ii} > 0$, since $\boldsymbol{W}^{(k)} \to \overline{\boldsymbol{W}}$ we know that, for all sufficiently large $k \geq N$ for some fixed value N, it is the case that $\boldsymbol{W}_{ii}^{(k)} > 0$. Furthermore, this implies that as $k \to \infty$ we have $\overline{\boldsymbol{W}}_{ii}^2/(\boldsymbol{W}_{ii}^{(k)})^2 \to 1$ and $\left\|\boldsymbol{W}^{(k)} - \overline{\boldsymbol{W}}\right\|_\infty \to 0$. Applying Lemma 3.8.1 with $\boldsymbol{W} = \boldsymbol{W}^{(k)}$ and taking $\liminf_{k\to\infty}$ on both sides of (3.8) gives the result. □

Chapter 4

Linear-Sized Sparsifier In Almost-Linear Time

4.1 Introduction

In the last chapter, we showed how to compute $O(n \log n)$ sized spectral sparsifier in input sparsity time. In this chapter, we show how to construct a spectral sparsifier with even smaller size. In particular, we present an almost-linear time algorithm for constructing linear-sized spectral sparsifiers for graphs, which improves all previous constructions that either require $\Omega\left(n^{2+\varepsilon}\right)$ time in order to produce linear-sized sparsifiers [9, 31, 278], or $O(m \log^{O(1)} n/\varepsilon^2)$ time but the number of edges in the sparsifiers is sub-optimal [242]. Our algorithm is conceptually simple, and is based on a novel combination of two techniques used in literature for constructing spectral sparsifiers: random sampling by leverage scores we used in the last chapter, and adaptive construction based on barrier functions [9, 31]. Our result is summarized as follows:

Theorem 4.1.1. *Let $q \geq 1$ and $0 < \varepsilon \leq 1$ be constants, and $\boldsymbol{I} = \sum_{i=1}^{m} \boldsymbol{v}_i \boldsymbol{v}_i^T$ be the sum of m rank-1 PSD matrices. Then, there is an algorithm that outputs scalars $\{s_i\}_{i=1}^{m}$ with $|\{s_i : s_i \neq 0\}| = O\left(qn/\varepsilon^2\right)$ such that*

$$(1-\varepsilon) \cdot \boldsymbol{I} \preceq \sum_{i=1}^{m} s_i \boldsymbol{v}_i \boldsymbol{v}_i^T \preceq (1+\varepsilon) \cdot I.$$

The algorithm runs in $\tilde{O}\left(Z + n^{\omega+1/q} \varepsilon^{-4}\right)$ time, where Z is the total number of non-zeros in $\boldsymbol{v}_i$.

Theorem 4.1.2. *Let $q \geq 1$ and $0 < \varepsilon \leq 1$ be constants, and $G = (V, E, \boldsymbol{w})$ be an undirected and weighted graph with n vertices and m edges. Then, there is an algorithm that outputs a ε-spectral sparsifier of G with $O\left(qn/\varepsilon^2\right)$ edges. The algorithm runs in $\tilde{O}\left(m + n^{1+1/q} \varepsilon^{-(6+1/q)}\right)$ time.*

4.2 Algorithm

In this section we study the algorithm of sparsifying the sum of rank-1 PSD matrices. Remember that our goal is to, for any vectors $\boldsymbol{v}_1, \cdots \boldsymbol{v}_m$ with $\sum_{i=1}^{m} \boldsymbol{v}_i \boldsymbol{v}_i^T = I$, find scalars $\{s_i\}_{i=1}^{m}$ satisfying

$$|\{s_i : s_i \neq 0\}| = O\left(\frac{qn}{\varepsilon^2}\right),$$

such that

$$(1-\varepsilon) \cdot \boldsymbol{I} \preceq \sum_{i=1}^{m} s_i \boldsymbol{v}_i \boldsymbol{v}_i^T \preceq (1+\varepsilon) \cdot \boldsymbol{I}.$$

4.2.1 Overview of our approach

At a high level, our algorithm can be viewed as an improved and randomized version of the algorithm presented in Batson et al. [31]. We refer their algorithm BSS for short, and first give a brief overview of the BSS algorithm.

The BSS algorithm proceeds by iterations, and the matrix $\boldsymbol{A}_j$ in iteration $j \geq 1$ is defined as the sum of $\boldsymbol{A}_{j-1}$ and a proper chosen rank-1 matrix. To control the spectral properties of matrix $\boldsymbol{A}_j$, the algorithm introduces two barrier values u_j and ℓ_j, where $u_0 > 0, \ell_0 < 0$ initially. It was proven that one can always find a vector in $\{\boldsymbol{v}_i\}_{i=1}^m$ and update u_j, ℓ_j in a proper manner in each iteration, such that the invariant

$$\ell_j \boldsymbol{I} \prec \boldsymbol{A}_j \prec u_j \boldsymbol{I} \tag{4.1}$$

always holds [31]. To quantitatively measure "how close the eigenvalues of $\boldsymbol{A}$ are to the barriers u and ℓ", Batson et al. [31] analyzed the potential function defined by

$$\Phi_{u,\ell}(\boldsymbol{A}) \triangleq \operatorname{Tr}(u\boldsymbol{I} - \boldsymbol{A})^{-1} + \operatorname{Tr}(\boldsymbol{A} - \ell\boldsymbol{I})^{-1}. \tag{4.2}$$

Notice that the value of $\Phi_{u,\ell}(\boldsymbol{A})$ is large if and only if some eigenvalue of $\boldsymbol{A}$ is close to u or ℓ. With the help of this potential function, it was shown that, when updating $\boldsymbol{A}_j$ and barrier values u_j, ℓ_j properly, it holds after $k = \Theta\left(n/\varepsilon^2\right)$ iterations that $\ell_k \geq cu_k$ for some constant c. This implies that the resulting matrix $\boldsymbol{A}_k$ is a linear-sized and $\boldsymbol{A}_k \approx_{O(\varepsilon)} \boldsymbol{I}$.

However, the BSS algorithm is deterministic, and its time complexity for finding a desired rank-1 matrix in each iteration is high. To improve the runtime and further discuss our approach, let us study the following randomized variant of the BSS algorithm: In each iteration, we choose a vector $\boldsymbol{v}_i$ with probability p_i, and add a rank-1 matrix

$$\boldsymbol{\Delta}_{\boldsymbol{A}} \triangleq \frac{\varepsilon}{\operatorname{Tr}(u\boldsymbol{I} - \boldsymbol{A})^{-1} + \operatorname{Tr}(\boldsymbol{A} - \ell\boldsymbol{I})^{-1}} \cdot \frac{1}{p_i} \cdot \boldsymbol{v}_i \boldsymbol{v}_i^T$$

to the current matrix A. See Algorithm 3 for formal description.

Algorithm 3: Randomized BSS algorithm

Set $j = 0$, $\ell_0 = -8n/\varepsilon$, $u_0 = 8n/\varepsilon$, $\boldsymbol{A}_0 = \boldsymbol{0}$.
while $u_j - \ell_j < 8n/\varepsilon$ **do**
  Let $t = \operatorname{Tr}\left(u_j\boldsymbol{I} - \boldsymbol{A}_j\right)^{-1} + \operatorname{Tr}\left(\boldsymbol{A}_j - \ell_j\boldsymbol{I}\right)^{-1}$.
  Sample a vector $\boldsymbol{v}_i$ with probability

$$p_i \triangleq \left(\boldsymbol{v}_i^T \left(u_j\boldsymbol{I} - \boldsymbol{A}_j\right)^{-1} \boldsymbol{v}_i + \boldsymbol{v}_i^T \left(\boldsymbol{A}_j - \ell_j\boldsymbol{I}\right)^{-1} \boldsymbol{v}_i\right)/t.$$

  $\boldsymbol{A}_{j+1} = \boldsymbol{A}_j + \frac{\varepsilon}{t} \cdot \frac{1}{p_i} \cdot \boldsymbol{v}_i \boldsymbol{v}_i^T$;
  $u_{j+1} = u_j + \frac{\varepsilon}{t\cdot(1-\varepsilon)}$ and $\ell_{j+1} = \ell_j + \frac{\varepsilon}{t\cdot(1+\varepsilon)}$;
  $j \leftarrow j + 1$;
end
Output: $\boldsymbol{A}_j$.

Let us look at a fixed iteration j, and analyze how the added $\boldsymbol{\Delta}_{\boldsymbol{A}}$ impacts the potential function. We drop the subscript representing the iteration j for simplicity. After adding $\boldsymbol{\Delta}_{\boldsymbol{A}}$, the first-order approximation of $\Phi_{u,\ell}(\boldsymbol{A})$ gives that

$$\Phi_{u,\ell}(\boldsymbol{A} + \boldsymbol{\Delta}_{\boldsymbol{A}}) \sim \Phi_{u,\ell}(\boldsymbol{A}) + (u\boldsymbol{I} - \boldsymbol{A})^{-2} \bullet \boldsymbol{\Delta}_{\boldsymbol{A}} - (\boldsymbol{A} - \ell\boldsymbol{I})^{-2} \bullet \boldsymbol{\Delta}_{\boldsymbol{A}}. \tag{4.3}$$

Since

$$\mathbf{E}\left[\boldsymbol{\Delta}_{\boldsymbol{A}}\right] = \sum_{i=1}^m p_i \cdot \left(\frac{\varepsilon}{t} \cdot \frac{1}{p_i} \cdot \boldsymbol{v}_i \boldsymbol{v}_i^T\right) = \frac{\varepsilon}{t} \cdot \sum_{i=1}^m \boldsymbol{v}_i \boldsymbol{v}_i^T = \frac{\varepsilon}{t} \cdot I,$$

we have that

$$\begin{aligned}\mathbf{E}\left[\Phi_{u,\ell}(\boldsymbol{A}+\boldsymbol{\Delta_A})\right] &\sim \Phi_{u,\ell}(\boldsymbol{A})+\frac{\varepsilon}{t}\cdot(u\boldsymbol{I}-\boldsymbol{A})^{-2}\bullet I-\frac{\varepsilon}{t}\cdot(\boldsymbol{A}-\ell\boldsymbol{I})^{-2}\bullet I\\ &= \Phi_{u,\ell}(\boldsymbol{A})+\frac{\varepsilon}{t}\cdot\mathrm{Tr}\,(u\boldsymbol{I}-\boldsymbol{A})^{-2}-\frac{\varepsilon}{t}\cdot\mathrm{Tr}\,(\boldsymbol{A}-\ell\boldsymbol{I})^{-2}\\ &= \Phi_{u,\ell}(\boldsymbol{A})-\frac{\varepsilon}{t}\cdot\frac{\mathrm{d}}{\mathrm{d}u}\Phi_{u,\ell}(\boldsymbol{A})-\frac{\varepsilon}{t}\cdot\frac{\mathrm{d}}{\mathrm{d}\ell}\Phi_{u,\ell}(\boldsymbol{A}).\end{aligned}$$

Notice that if we increase u by ε/t and ℓ by ε/t, $\Phi_{u,\ell}$ approximately increases by

$$\frac{\varepsilon}{t}\cdot\frac{\mathrm{d}}{\mathrm{d}u}\Phi_{u,\ell}(\boldsymbol{A})+\frac{\varepsilon}{t}\cdot\frac{\mathrm{d}}{\mathrm{d}\ell}\Phi_{u,\ell}(\boldsymbol{A}).$$

Hence, comparing $\Phi_{u+\varepsilon/t,\ell+\varepsilon/t}(\boldsymbol{A}+\boldsymbol{\Delta_A})$ with $\Phi_{u,\ell}(\boldsymbol{A})$, the increase of the potential function due to the change of barrier values is approximately compensated by the decrease of the potential function by the effect of $\boldsymbol{\Delta_A}$. For a more rigorous analysis, we need to analyze the higher-order terms and increase u slightly more than ℓ. Batson et al. [31] gives the following estimate:

Lemma 4.2.1 ([31], proof of Lemma 3.3 and 3.4)**.** *Let $\boldsymbol{A}\in\mathbb{R}^{n\times n}$, and u,ℓ be parameters satisfying $\ell\boldsymbol{I}\prec\boldsymbol{A}\prec u\boldsymbol{I}$. Suppose that $\boldsymbol{w}\in\mathbb{R}^n$ satisfies $\boldsymbol{w}\boldsymbol{w}^T\preceq\delta(u\boldsymbol{I}-\boldsymbol{A})$ and $\boldsymbol{w}\boldsymbol{w}^T\preceq\delta(\boldsymbol{A}-\ell\boldsymbol{I})$ for some $0<\delta<1$. Then, it holds that*

$$\Phi_{u,\ell}(\boldsymbol{A}+\boldsymbol{w}\boldsymbol{w}^T)\le\Phi_{u,\ell}(\boldsymbol{A})+\frac{\boldsymbol{w}^T(u\boldsymbol{I}-\boldsymbol{A})^{-2}\boldsymbol{w}}{1-\delta}-\frac{\boldsymbol{w}^T(\boldsymbol{A}-\ell\boldsymbol{I})^{-2}\boldsymbol{w}}{1+\delta}.$$

The estimate above shows that the first-order approximation (4.3) is good as long as $\boldsymbol{w}\boldsymbol{w}^T\preceq\delta(u\boldsymbol{I}-\boldsymbol{A})$ and $\boldsymbol{w}\boldsymbol{w}^T\preceq\delta(\boldsymbol{A}-\ell\boldsymbol{I})$ for small δ. By setting $\delta=\varepsilon$, it is easy to see that the added matrix $\boldsymbol{\Delta_A}$ satisfies the preconditions in Lemma 4.2.1, since

$$\frac{\varepsilon}{t}\cdot\frac{1}{p_i}\cdot\boldsymbol{v}_i\boldsymbol{v}_i^T = \frac{\varepsilon\cdot\boldsymbol{v}_i\boldsymbol{v}_i^T}{v_i^T(u\boldsymbol{I}-\boldsymbol{A})^{-1}v_i+\boldsymbol{v}_i^T(\boldsymbol{A}-\ell\boldsymbol{I})^{-1}\boldsymbol{v}_i}\preceq\frac{\varepsilon\cdot v_iv_i^T}{v_i^T(u\boldsymbol{I}-\boldsymbol{A})^{-1}v_i}\preceq\varepsilon\left(u\boldsymbol{I}-\boldsymbol{A}\right).$$

Here we used the fact that $vv^T\preceq(v^TB^{-1}v)B$ for any vector v and PSD matrix B. Similarly, we have that

$$\frac{\varepsilon}{t}\cdot\frac{1}{p_i}\cdot v_iv_i^T\preceq\varepsilon(A-\ell I).$$

Hence, if the initial value of the potential function is small, $\Phi_{u,\ell}(A)$ is small throughout the executions of the whole algorithm. Up to a constant factor, this gives the same result as [31], and Algorithm 3 constructs an $\Theta(n/\varepsilon^2)$-sized $O(\varepsilon)$-spectral sparsifier.

However, Algorithm 3 runs for $\Theta(n/\varepsilon^2)$ iterations, and in each iteration the algorithm re-computes the probability distribution for sampling vectors. To improve the runtime of Algorithm 3, we need to overcome two bottlenecks: (1) we need a fast algorithm to approximate the sampling probability $\{p_i\}_{i=1}^m$ of vectors; (2) we need to reduce the number of iterations required by the algorithm, i.e. re-using the sampling probability $\{p_i\}_{i=1}^m$ for few iterations.

For fast approximation of the sampling probabilities, we adopt the idea proposed in [9]: instead of defining the potential function by (4.2), we define the potential function by

$$\Phi_{u,\ell}(\boldsymbol{A})\triangleq\mathrm{Tr}(u\boldsymbol{I}-\boldsymbol{A})^{-q}+\mathrm{Tr}(\boldsymbol{A}-\ell\boldsymbol{I})^{-q}. \tag{4.4}$$

Since q is a large constant, the value of the potential function becomes larger when some eigenvalue of A is closer to u or ℓ. Hence, a bounded value of $\Phi_{u,\ell}(A)$ insures that the eigenvalues of A never get too close to u or ℓ, which allows us to compute the sampling probabilities $\{p_i\}_{i=1}^m$ efficiently simply by Taylor expansion. Moreover, with our new potential function (4.4) one can prove a similar result

as Lemma 4.2.1. This gives an alternative analysis of the algorithm presented in [9], which is the first almost-quadratic time algorithm for constructing linear-sized spectral sparsifiers.

To overcome the second bottleneck, we re-compute $\{p_i\}_{i=1}^m$ after every $\Theta\left(n^{1-1/q}\right)$ iterations: we show that as long as the sampling probability satisfies

$$p_i \geq C \cdot \frac{\boldsymbol{v}_i^T \left(u\boldsymbol{I} - \boldsymbol{A}\right)^{-1} \boldsymbol{v}_i + \boldsymbol{v}_i^T \left(\boldsymbol{A} - \ell\boldsymbol{I}\right)^{-1} \boldsymbol{v}_i}{\sum_{i=1}^m \left(\boldsymbol{v}_i^T \left(u\boldsymbol{I} - \boldsymbol{A}\right)^{-1} \boldsymbol{v}_i + \boldsymbol{v}_i^T \left(\boldsymbol{A} - \ell\boldsymbol{I}\right)^{-1} \boldsymbol{v}_i\right)}$$

for some constant $C > 0$, we can still sample v_i with probability p_i and get the same guarantee on the potential function. The reason is as follows: assume that $\boldsymbol{\Delta}_{\boldsymbol{A}} = \sum_{i=1}^T \boldsymbol{\Delta}_{\boldsymbol{A},i}$ is the sum of the sampled matrices within $T = O\left(n^{1-1/q}\right)$ iterations. If a randomly chosen matrix $\boldsymbol{\Delta}_{\boldsymbol{A},i}$ satisfies $\boldsymbol{\Delta}_{\boldsymbol{A},i} \preceq \frac{1}{Cq}\left(u\boldsymbol{I} - \boldsymbol{A}\right)$, then by the matrix Chernoff bound $\boldsymbol{\Delta}_{\boldsymbol{A}} \preceq \frac{1}{2}\left(u\boldsymbol{I} - \boldsymbol{A}\right)$ holds with high probability. By scaling every sampled rank-1 matrix q times smaller, the sampling probability only changes by a constant factor within T iterations. Since we choose $\Theta(n/\varepsilon^2)$ vectors in total, our algorithm only recomputes the sampling probabilities $\Theta\left(n^{1/q}/\varepsilon^2\right)$ times.

4.2.2 Algorithm description

Our actual algorithm follows the same framework as Algorithm 3, and proceeds by iterations. Initially, the algorithm sets

$$u_0 \triangleq (2n)^{1/q}, \qquad \ell_0 \triangleq -(2n)^{1/q}, \qquad A_0 \triangleq \mathbf{0}.$$

After iteration j the algorithm updates u_j, ℓ_j by $\Delta_{u,j}, \Delta_{\ell,j}$ respectively, i.e.,

$$u_{j+1} \triangleq u_j + \Delta_{u,j}, \qquad \ell_{j+1} \triangleq \ell_j + \Delta_{\ell,j},$$

and updates A_j with respect to the chosen matrix in iteration j. The choice of $\Delta_{u,j}$ and $\Delta_{\ell,j}$ insures that the invariant

$$\ell_j I \prec \boldsymbol{A}_j \prec u_j I$$

always holds for any iteration j. In iteration j, the algorithm computes the *relative leverage score* of vectors $\{v_i\}_{i=1}^m$ defined by

$$R_i\left(\boldsymbol{A}_j, u_j, \ell_j\right) \triangleq \boldsymbol{v}_i^T \left(u_j\boldsymbol{I} - \boldsymbol{A}_j\right)^{-1} \boldsymbol{v}_i + \boldsymbol{v}_i^T \left(\boldsymbol{A}_j - \ell_j\boldsymbol{I}\right)^{-1} \boldsymbol{v}_i,$$

and samples N_j vectors independently with replacement, where vector v_i is chosen with probability proportional to $R_i(A_j, u_j, \ell_j)$, and

$$N_j \triangleq \frac{1}{n^{2/q}} \left(\sum_{i=1}^m R_i(\boldsymbol{A}_j, u_j, \ell_j)\right) \min\left\{\lambda_{\min}(u_j\boldsymbol{I} - \boldsymbol{A}_j), \lambda_{\min}(\boldsymbol{A}_j - \ell_j\boldsymbol{I})\right\}.$$

The algorithm sets $\boldsymbol{A}_{j+1}$ to be the sum of $\boldsymbol{A}_j$ and sampled $\boldsymbol{v}_i\boldsymbol{v}_i^T$ with proper reweighting. For technical reasons, we define $\Delta_{u,j}$ and $\Delta_{\ell,j}$ by

$$\Delta_{u,j} \triangleq (1 + 2\varepsilon) \cdot \frac{\varepsilon \cdot N_j}{q \cdot \sum_{i=1}^m R_i(\boldsymbol{A}_j, u_j, \ell_j)}, \qquad \Delta_{\ell,j} \triangleq (1 - 2\varepsilon) \cdot \frac{\varepsilon \cdot N_j}{q \cdot \sum_{i=1}^m R_i(\boldsymbol{A}_j, u_j, \ell_j)}.$$

See Algorithm 4 for formal description.

We remark that, although exact values of N_j and relative leverage scores are difficult to compute in almost-linear time, we can use approximated values of $\{R_i\}_{i=1}^m$ and N_j instead. It is easy to see that in each iteration an over estimate of the relative leverage score for every vector $\boldsymbol{v}_i$, and an under estimate of N_j with constant-factor approximation suffice for our purpose.

Algorithm 4: Algorithm for constructing spectral sparsifiers

Assume: $0 < \varepsilon \leq 1/120, q \geq 10$.
Set $j = 0, \ell_0 = -(2n)^{1/q}, u_0 = (2n)^{1/q}, \boldsymbol{A}_0 = \boldsymbol{0}$.
while $u_j - \ell_j < 4 \cdot (2n)^{1/q}$ **do**
  Let $\boldsymbol{W}_j = \boldsymbol{0}$.
  Compute $R_i(\boldsymbol{A}_j, u_j, \ell_j)$ for all vectors v_i.
  Sample N_j vectors independently with replacement, where every v_i is chosen with probability proportional to $R_i(\boldsymbol{A}_j, u_j, \ell_j)$.
  For every sampled v, add $\varepsilon/q \cdot (R_i(\boldsymbol{A}_j, u_j, \ell_j))^{-1} \cdot vv^T$ to $\boldsymbol{W}_j$.
  $\boldsymbol{A}_{j+1} = \boldsymbol{A}_j + \boldsymbol{W}_j$.
  $u_{j+1} = u_j + \Delta_{u,j}$, $\ell_{j+1} = \ell_j + \Delta_{\ell,j}$.
  $j = j + 1$.
end
Output: $\boldsymbol{A}_j$.

4.3 Analysis

In this section we prove that the output matrix returned by Algorithm 4 is a ε-spectral sparsifier, and analyze the algorithm's runtime. To simplify our analysis, we always assume in the rest of the chapter that ε and q are constants such that $0 < \varepsilon \leq 1/120$, and $q \geq 10$.

This section is organized as follows: we will first show how the potential function (4.4) evolves after each iteration in Section 4.3.1. Combing this with the ending condition of the algorithm, we will prove in Section 4.3.2 that the algorithm outputs a linear-sized spectral sparsifier. Finally, we will prove Theorem 4.1.2 and Theorem 4.1.1 in Section 4.3.3. In appendix, we show almost-linear time algorithms for approximately computing all required quantities of Algorithm 4 within each iteration when constructing graph sparsifiers.

4.3.1 Analysis of a single iteration

We analyze the sampling scheme within a single iteration, and drop the subscript representing the iteration j for simplicity. Recall that in each iteration the algorithm samples N vectors independently from $\{\boldsymbol{v}_i\}_{i=1}^m$ satisfying $\sum_{i=1}^m \boldsymbol{v}_i \boldsymbol{v}_i^T = \boldsymbol{I}$, where every vector $\boldsymbol{v}_i$ is sampled with probability

$$\frac{R_i(\boldsymbol{A}, u, \ell)}{\sum_{j=1}^m R_j(\boldsymbol{A}, u, \ell)}.$$

We use $\boldsymbol{v}_1, \cdots, \boldsymbol{v}_N$ to denote these N sampled vectors, and define the reweighted vectors by

$$\boldsymbol{w}_i \triangleq \sqrt{\frac{\varepsilon}{q \cdot R_i(\boldsymbol{A}, u, \ell)}} \cdot \boldsymbol{v}_i,$$

for any $1 \leq i \leq N$. Let

$$\boldsymbol{W} \triangleq \sum_{i=1}^N \boldsymbol{w}_i \boldsymbol{w}_i^T,$$

and we use $\boldsymbol{W} \sim \mathcal{D}(\boldsymbol{A}, u, \ell)$ to represent that $\boldsymbol{W}$ is sampled in this way with parameters $\boldsymbol{A}, u$ and ℓ. We will show that with high probability matrix $\boldsymbol{W}$ satisfies $\boldsymbol{0} \preceq \boldsymbol{W} \preceq \frac{1}{2}(u\boldsymbol{I} - \boldsymbol{A})$. This is basically follows from Matrix Chernoff Bound (Lemma 2.3.10).

Lemma 4.3.1. *Assume that the number of sampled vectors satisfies*

$$N < \frac{2}{n^{2/q}} \left(\sum_{i=1}^{m} R_i(\boldsymbol{A}, u, \ell) \right) \cdot \lambda_{\min}(u\boldsymbol{I} - \boldsymbol{A}).$$

Then, it holds that $\mathbf{E}[\boldsymbol{W}] = \frac{\varepsilon}{q} \cdot \frac{N}{\sum_{i=1}^{m} R_i(\boldsymbol{A},u,\ell)} \cdot \boldsymbol{I}$ *and*

$$Pr\left[\mathbf{0} \preceq \boldsymbol{W} \preceq \frac{1}{2} \cdot (u\boldsymbol{I} - \boldsymbol{A}) \right] \geq 1 - \frac{\varepsilon^2}{100qn}.$$

Proof. By the description of the sampling procedure, the first statement follows from the calculation

$$\mathbf{E}\left[\boldsymbol{w}_i \boldsymbol{w}_i^T\right] = \sum_{j=1}^{m} \frac{R_j(\boldsymbol{A}, u, \ell)}{\sum_{t=1}^{m} R_t(\boldsymbol{A}, u, \ell)} \cdot \frac{\varepsilon}{q} \cdot \frac{\boldsymbol{v}_j \boldsymbol{v}_j^T}{R_j(\boldsymbol{A}, u, \ell)} = \frac{\varepsilon}{q} \cdot \frac{1}{\sum_{t=1}^{m} R_t(\boldsymbol{A}, u, \ell)} \cdot \boldsymbol{I},$$

and

$$\mathbf{E}[\boldsymbol{W}] = \mathbf{E}\left[\sum_{i=1}^{N} \boldsymbol{w}_i \boldsymbol{w}_i^T \right] = \frac{\varepsilon}{q} \cdot \frac{N}{\sum_{i=1}^{m} R_i(\boldsymbol{A}, u, \ell)} \cdot \boldsymbol{I}.$$

Now for the second statement. Let $\boldsymbol{z}_i = (u\boldsymbol{I} - \boldsymbol{A})^{-1/2} \boldsymbol{w}_i$. It holds that

$$\begin{aligned} \mathrm{Tr}\left(\boldsymbol{z}_i \boldsymbol{z}_i^T\right) &= \frac{\varepsilon}{q} \cdot \frac{\mathrm{Tr}\left((u\boldsymbol{I} - \boldsymbol{A})^{-1/2} \boldsymbol{v}_i \boldsymbol{v}_i^T (u\boldsymbol{I} - \boldsymbol{A})^{-1/2}\right)}{R_i(A, u, \ell)} \\ &\leq \frac{\varepsilon}{q} \cdot \frac{\boldsymbol{v}_i^T (u\boldsymbol{I} - \boldsymbol{A})^{-1} \boldsymbol{v}_i}{\boldsymbol{v}_i^T (u\boldsymbol{I} - \boldsymbol{A})^{-1} \boldsymbol{v}_i + \boldsymbol{v}_i^T (\boldsymbol{A} - \ell\boldsymbol{I})^{-1} \boldsymbol{v}_i} \leq \frac{\varepsilon}{q}, \end{aligned}$$

and $\lambda_{\max}(\boldsymbol{z}_i \boldsymbol{z}_i^T) \leq \frac{\varepsilon}{q}$. Moreover, it holds that

$$\mathbf{E}\left[\sum_{i=1}^{N} \boldsymbol{z}_i \boldsymbol{z}_i^T \right] \preceq \frac{\varepsilon}{q} \cdot \frac{N}{\sum_{t=1}^{m} R_t(\boldsymbol{A}, u, \ell)} \cdot \lambda_{\max}\left(\frac{1}{u\boldsymbol{I} - \boldsymbol{A}} \right) \cdot \boldsymbol{I}. \tag{4.5}$$

This implies that

$$\lambda_{\max}\left(\mathbf{E}\left[\sum_{i=1}^{N} \boldsymbol{z}_i \boldsymbol{z}_i^T \right] \right) \leq \frac{\varepsilon}{q} \cdot \frac{N}{\sum_{t=1}^{m} R_t(\boldsymbol{A}, u, \ell)} \cdot \lambda_{\max}\left(\frac{1}{u\boldsymbol{I} - \boldsymbol{A}} \right).$$

Setting $\mu = \frac{\varepsilon}{q} \cdot \frac{N}{\sum_{i=1}^{m} R_i(\boldsymbol{A},u,\ell)} \cdot \lambda_{\max}\left(\frac{1}{u\boldsymbol{I}-\boldsymbol{A}}\right)$, the Matrix Chernoff Bound (Lemma 2.3.10) shows that

$$\Pr\left[\lambda_{\max}\left(\sum_{i=1}^{N} \boldsymbol{z}_i \boldsymbol{z}_i^T \right) \geq (1+\delta)\mu \right] \leq n \cdot \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}} \right)^{\mu \cdot q/\varepsilon}.$$

Set the value of $1 + \delta$ to be

$$\begin{aligned} 1 + \delta = \frac{1}{2\mu} &= \frac{q}{2\varepsilon N} \cdot \left(\sum_{j=1}^{m} R_j(\boldsymbol{A}, u, \ell) \right) \cdot \frac{1}{\lambda_{\max}\left(\frac{1}{u\boldsymbol{I}-\boldsymbol{A}}\right)} \\ &= \frac{q}{2\varepsilon N} \cdot \left(\sum_{j=1}^{m} R_j(\boldsymbol{A}, u, \ell) \right) \cdot \lambda_{\min}(u\boldsymbol{I} - \boldsymbol{A}) \geq \frac{q}{4\varepsilon} \cdot n^{2/q}, \end{aligned}$$

where the last inequality follows from the condition on N. Hence, with probability at least

$$1 - n \cdot \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\mu \cdot q/\varepsilon} \geq 1 - n \cdot \left(\frac{e}{1+\delta}\right)^{(1+\delta)\cdot\mu\cdot q/\varepsilon} \geq 1 - n \left(\frac{e}{1+\delta}\right)^{\frac{q}{2\varepsilon}} \geq 1 - \frac{\varepsilon^2}{100qn},$$

we have that

$$\lambda_{\max}\left(\sum_{i=1}^{N} z_i z_i^T\right) \leq (1+\delta)\cdot\mu = \frac{1}{2},$$

which implies that $\mathbf{0} \preceq \sum_{i=1}^{N} \boldsymbol{z}_i \boldsymbol{z}_i^T \preceq \frac{1}{2}\cdot I$ and $\mathbf{0} \preceq W \preceq \frac{1}{2}\cdot(uI - A)$. □

Now we analyze the change of the potential function after each iteration, and show that the expected value of the potential function decreases over time. By Lemma 4.3.1, with probability at least $1 - \frac{\varepsilon^2}{100qn}$, it holds that

$$\mathbf{0} \preceq \boldsymbol{W} \preceq \frac{1}{2}(u\boldsymbol{I} - \boldsymbol{A}).$$

We define

$$\widetilde{\mathbf{E}}\left[f(\boldsymbol{W})\right] \triangleq \sum_{\boldsymbol{W}\sim\mathcal{D}(\boldsymbol{A},u,\ell)} \Pr\left[\boldsymbol{W} \text{ is chosen and } \boldsymbol{W} \preceq \frac{1}{2}(u\boldsymbol{I} - \boldsymbol{A})\right] \cdot f\left(\boldsymbol{W}\right).$$

Lemma 4.3.2 below shows how the potential function changes after each iteration, and plays a key role in our analysis. This lemma was first proved in [31] for the case of $q = 1$, and similar lemma was proved in [9].

Lemma 4.3.2 ([9]). *Let $q \geq 10$ and $\varepsilon \leq 1/10$. Suppose that $\boldsymbol{w}^T(u\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{w} \leq \varepsilon/q$ and $\boldsymbol{w}^T(\boldsymbol{A} - \ell\boldsymbol{I})^{-1}\boldsymbol{w} \leq \varepsilon/q$. Then, it holds that*

$$\mathrm{Tr}(\boldsymbol{A} + \boldsymbol{w}\boldsymbol{w}^T - \ell\boldsymbol{I})^{-q} \leq \mathrm{Tr}(\boldsymbol{A} - \ell\boldsymbol{I})^{-q} - q(1-\varepsilon)\ \boldsymbol{w}^T(\boldsymbol{A} - \ell\boldsymbol{I})^{-(q+1)}\boldsymbol{w},$$

and

$$\mathrm{Tr}(u\boldsymbol{I} - \boldsymbol{A} - \boldsymbol{w}\boldsymbol{w}^T)^{-q} \leq \mathrm{Tr}(u\boldsymbol{I} - \boldsymbol{A})^{-q} + q(1+\varepsilon)\ \boldsymbol{w}^T(u\boldsymbol{I} - \boldsymbol{A})^{-(q+1)}\boldsymbol{w}.$$

Proof. Let $\boldsymbol{Y} = \boldsymbol{A} - \ell\boldsymbol{I}$. By the Sherman-Morrison Formula (Lemma 2.3.7), it holds that

$$\mathrm{Tr}(\boldsymbol{Y} + \boldsymbol{w}\boldsymbol{w}^T)^{-q} = \mathrm{Tr}\left(\boldsymbol{Y}^{-1} - \frac{\boldsymbol{Y}^{-1}ww^T\boldsymbol{Y}^{-1}}{1 + w^T\boldsymbol{Y}^{-1}w}\right)^{q}. \tag{4.6}$$

By the assumption of $w^T\boldsymbol{Y}^{-1}w \leq \varepsilon/q$, we have that

$$\begin{aligned}
\mathrm{Tr}(\boldsymbol{Y} + ww^T)^{-q} &\leq \mathrm{Tr}\left(\boldsymbol{Y}^{-1} - \frac{\boldsymbol{Y}^{-1}ww^T\boldsymbol{Y}^{-1}}{1+\varepsilon/q}\right)^{q} && (4.7)\\
&= \mathrm{Tr}\left(\boldsymbol{Y}^{-1/2}\left(\boldsymbol{I} - \frac{\boldsymbol{Y}^{-1/2}ww^T\boldsymbol{Y}^{-1/2}}{1+\varepsilon/q}\right)\boldsymbol{Y}^{-1/2}\right)^{q} \\
&\leq \mathrm{Tr}\left(\boldsymbol{Y}^{-q/2}\left(\boldsymbol{I} - \frac{\boldsymbol{Y}^{-1/2}ww^T\boldsymbol{Y}^{-1/2}}{1+\varepsilon/q}\right)^{q}\boldsymbol{Y}^{-q/2}\right) && (4.8)\\
&= \mathrm{Tr}\left(\boldsymbol{Y}^{-q}\left(\boldsymbol{I} - \frac{\boldsymbol{Y}^{-1/2}ww^T\boldsymbol{Y}^{-1/2}}{1+\varepsilon/q}\right)^{q}\right), && (4.9)
\end{aligned}$$

where (4.7) uses the fact that $\boldsymbol{A} \preceq \boldsymbol{B}$ implies that $\mathrm{Tr}\left(\boldsymbol{A}^q\right) \leq \mathrm{Tr}\left(\boldsymbol{B}^q\right)$, (4.8) follows from the Lieb-Thirring inequality (Lemma 2.3.11), and (4.8) uses the fact that the trace is invariant under cyclic

permutations.

Let $\boldsymbol{D} = \frac{\boldsymbol{Y}^{-1/2} w w^T \boldsymbol{Y}^{-1/2}}{1+\varepsilon/q}$. Note that $0 \preceq D \preceq \frac{\varepsilon}{q} \cdot \boldsymbol{I}$, and

$$\begin{aligned}(I-D)^q &\preceq I - qD + \frac{q(q-1)}{2} D^2 \\ &\preceq I - \left(q - \frac{\varepsilon(q-1)}{2}\right) D.\end{aligned}$$

Therefore, we have that

$$\begin{aligned}\left(I - \frac{\boldsymbol{Y}^{-1/2} w w^T \boldsymbol{Y}^{-1/2}}{1+\varepsilon/q}\right)^q &\preceq I - \left(q - \frac{\varepsilon(q-1)}{2}\right) \frac{\boldsymbol{Y}^{-1/2} w w^T \boldsymbol{Y}^{-1/2}}{1+\varepsilon/q} \\ &\preceq I - \left(q - \frac{\varepsilon(q-1)}{2}\right)\left(1 - \frac{\varepsilon}{q}\right) \boldsymbol{Y}^{-1/2} w w^T \boldsymbol{Y}^{-1/2} \\ &\preceq I - q\left(1 - \frac{\varepsilon(q+1)}{2q}\right) \boldsymbol{Y}^{-1/2} w w^T \boldsymbol{Y}^{-1/2} \\ &\preceq I - q\,(1-\varepsilon)\, \boldsymbol{Y}^{-1/2} w w^T \boldsymbol{Y}^{-1/2}.\end{aligned}$$

This implies that

$$\begin{aligned}\operatorname{Tr}(Y + ww^T)^{-q} &\leq \operatorname{Tr}\left(Y^{-q}\left(I - q(1-\varepsilon) Y^{-1/2} w w^T Y^{-1/2}\right)\right) \\ &\leq \operatorname{Tr}\left(Y^{-q}\right) - q(1-\varepsilon)\; w^T Y^{-(q+1)} w,\end{aligned}$$

which proves the first statement.

Now for the second inequality. Let $Z = uI - A$. By the Sherman-Morrison Formula (Lemma 2.3.7), it holds that

$$\operatorname{Tr}(\boldsymbol{Z} - ww^T)^{-q} = \operatorname{Tr}\left(\boldsymbol{Z}^{-1} + \frac{\boldsymbol{Z}^{-1} w w^T \boldsymbol{Z}^{-1}}{1 - w^T \boldsymbol{Z}^{-1} w}\right)^q.$$

By the assumption of $w^T \boldsymbol{Z}^{-1} w \leq \varepsilon/q$, it holds that

$$\begin{aligned}\operatorname{Tr}(\boldsymbol{Z} - ww^T)^{-q} &\leq \operatorname{Tr}\left(\boldsymbol{Z}^{-1} + \frac{\boldsymbol{Z}^{-1} w w^T \boldsymbol{Z}^{-1}}{1 - \varepsilon/q}\right)^q && (4.10) \\ &= \operatorname{Tr}\left(\boldsymbol{Z}^{-1/2}\left(\boldsymbol{I} + \frac{\boldsymbol{Z}^{-1/2} w w^T \boldsymbol{Z}^{-1/2}}{1 - \varepsilon/q}\right) \boldsymbol{Z}^{-1/2}\right)^q \\ &\leq \operatorname{Tr}\left(\boldsymbol{Z}^{-q/2}\left(\boldsymbol{I} + \frac{\boldsymbol{Z}^{-1/2} w w^T \boldsymbol{Z}^{-1/2}}{1 - \varepsilon/q}\right)^q \boldsymbol{Z}^{-q/2}\right) && (4.11) \\ &= \operatorname{Tr}\left(\boldsymbol{Z}^{-q}\left(\boldsymbol{I} + \frac{\boldsymbol{Z}^{-1/2} w w^T \boldsymbol{Z}^{-1/2}}{1 - \varepsilon/q}\right)^q\right), && (4.12)\end{aligned}$$

where (4.10) uses the fact that $\boldsymbol{A} \preceq \boldsymbol{B}$ implies $\operatorname{Tr}(\boldsymbol{A}^q) \leq \operatorname{Tr}(\boldsymbol{B}^q)$, (4.11) follows from the Lieb-Thirring inequality (Lemma 2.3.11), and (4.12) uses the fact that the trace is invariant under cyclic permutations.

Let $\boldsymbol{E} = \boldsymbol{Z}^{-1/2} w w^T \boldsymbol{Z}^{-1/2}$. Combing $E \preceq \frac{\varepsilon}{q} \cdot \boldsymbol{I}$ with the assumption that $q \geq 10$ and $\varepsilon \leq 1/10$, we

have that

$$\begin{aligned}\left(I+\frac{E}{1-\varepsilon/q}\right)^q &\preceq I+\frac{qE}{1-\varepsilon/q}+\frac{q(q-1)}{2}\left(1+\frac{\varepsilon/q}{1-\varepsilon/q}\right)^{q-2}\left(\frac{E}{1-\varepsilon/q}\right)^2\\ &\preceq I+q\left(1+1.1\frac{\varepsilon}{q}\right)E+1.4\frac{q(q-1)}{2}E^2\\ &\preceq I+q\left(1+0.3\varepsilon\right)E+0.7\varepsilon qE\\ &\preceq I+q\left(1+\varepsilon\right)E.\end{aligned}$$

Therefore, we have that

$$\mathrm{Tr}(Z-ww^T)^{-q}\leq \mathrm{Tr}\left(Z^{-q}\right)+q(1+\varepsilon)\ w^TZ^{-(q+1)}w,$$

which proves the second statement. □

The following lemma shows that, with our choice of updated matrices and barrier values, the value of the potential function will never increase.

Lemma 4.3.3. *It holds for any iteration j that $\Phi_{u_{j+1},\ell_{j+1}}(A_{j+1})\leq\Phi_{u_j,\ell_j}(A_j)$.*

Proof. Let $w_1w_1^T,\cdots,w_{N_j}w_{N_j}^T$ be the matrices picked in iteration j, and define for any $0\leq i\leq N_j$ that $B_i=A_j+\sum_{t=1}^i w_tw_t^T$. We study the change of the potential function after adding a rank-1 matrix within each iteration. For this reason, we use

$$\overline{\Delta}_u=\frac{\Delta_{u,j}}{N_j}=(1+2\varepsilon)\cdot\frac{\varepsilon}{q\cdot\sum_{t=1}^m R_t(A_j,u_j,\ell_j)},$$

and

$$\overline{\Delta}_\ell=\frac{\Delta_{\ell,j}}{N_j}=(1-2\varepsilon)\cdot\frac{\varepsilon}{q\cdot\sum_{t=1}^m R_t(A_j,u_j,\ell_j)}$$

to express the average change of the barrier values $\Delta_{u,j}$ and $\Delta_{\ell,j}$. We further define for $0\leq j\leq N_j$ that $\hat{u}_i=u_j+i\cdot\overline{\Delta}_u,\qquad\hat{\ell}_i=\ell_j+i\cdot\overline{\Delta}_\ell$.

Assuming $\mathbf{0}\preceq W_j\preceq\frac{1}{2}(u_j\boldsymbol{I}-\boldsymbol{A_j})$, we claim that

$$w_iw_i^T\preceq\frac{2\varepsilon}{q}\cdot(\hat{u}_iI-B_{i-1})\ \text{ and }\ w_iw_i^T\preceq\frac{2\varepsilon}{q}\cdot\left(B_{i-1}-\hat{\ell}_iI\right),\tag{4.13}$$

for any $1\leq i\leq N_j$. Based on this, we apply Lemma 4.3.2 and get that

$$\begin{aligned}\Phi_{\hat{u}_i,\hat{\ell}_i}\left(B_{i-1}+w_iw_i^T\right)&\leq\Phi_{\hat{u}_i,\hat{\ell}_i}(B_{i-1})+q(1+2\varepsilon)\mathrm{Tr}\left((\hat{u}_iI-B_{i-1})^{-(q+1)}\mathbf{E}\left[w_iw_i^T\right]\right)\\&\quad-q(1-2\varepsilon)\mathrm{Tr}\left(\left(B_{i-1}-\hat{\ell}_iI\right)^{-(q+1)}\mathbf{E}\left[w_iw_i^T\right]\right)\\&=\Phi_{\hat{u}_i,\hat{\ell}_i}(B_{i-1})+q\cdot\overline{\Delta}_u\cdot\mathrm{Tr}\left((\hat{u}_iI-B_{i-1})^{-(q+1)}\right)\\&\quad-q\cdot\overline{\Delta}_\ell\cdot\mathrm{Tr}\left((B_{i-1}-\hat{\ell}_iI)^{-(q+1)}\right).\end{aligned}\tag{4.14}$$

We define a function $f_i:\mathbb{R}\to\mathbb{R}$ by

$$f_i(x)=\mathrm{Tr}\left(\left(\hat{u}_{i-1}+x\cdot\overline{\Delta}_u\right)I-B_{i-1}\right)^{-q}+\mathrm{Tr}\left(B_{i-1}-\left(\hat{\ell}_{i-1}+x\cdot\overline{\Delta}_\ell\right)I\right)^{-q}.$$

Notice that

$$\frac{\mathrm{d}f_i(x)}{\mathrm{d}x} = -q\cdot\overline{\Delta}_u\cdot\mathrm{Tr}\left(\left(\hat{u}_{i-1}+x\cdot\overline{\Delta}_u\right)I - B_{i-1}\right)^{-(q+1)} + q\cdot\overline{\Delta}_\ell\cdot\mathrm{Tr}\left(B_{i-1}-\left(\hat{\ell}_{i-1}+x\cdot\overline{\Delta}_\ell\right)I\right)^{-(q+1)}.$$

Since f is convex, we have that

$$\frac{\mathrm{d}f_i(x)}{\mathrm{d}x}\Big|_{x=1} \geq f_i(1)-f_i(0) = \Phi_{\hat{u}_i,\hat{\ell}_i}(B_{i-1}) - \Phi_{\hat{u}_{i-1},\hat{\ell}_{i-1}}(B_{i-1}). \tag{4.15}$$

Putting (4.14) and (4.15) together, we have that

$$\Phi_{\hat{u}_i,\hat{\ell}_i}(B_i) \leq \Phi_{\hat{u}_i,\hat{\ell}_i}(B_{i-1}) - \frac{\mathrm{d}f_i(x)}{\mathrm{d}x}\Big|_{x=1} \leq \Phi_{\hat{u}_{i-1},\hat{\ell}_{i-1}}(B_{i-1}).$$

Repeating this argument we have that

$$\Phi_{u_{j+1},\ell_{j+1}}(A_{j+1}) = \Phi_{\hat{u}_{N_j},\hat{\ell}_{N_j}}\left(B_{N_j}\right) \leq \Phi_{\hat{u}_0,\hat{\ell}_0}(B_0) = \Phi_{u_j,\ell_j}(A_j),$$

which proves the statement.

So, it suffices to prove (4.13). Since $vv^T \preceq (v^TB^{-1}v)B$ for any vector v and PSD matrix B, we have that

$$\frac{v_iv_i^T}{R_i(A_j,u_j,\ell_j)} \preceq \frac{v_iv_i^T}{v_i^T(u_jI-A_j)^{-1}v_i} \preceq u_jI - A_j.$$

By the assumption of $W_j \preceq \frac{1}{2}(u_jI-A_j)$, it holds that

$$w_iw_i^T = \frac{\varepsilon}{qR_i(A_j,u_j,\ell_j)}v_iv_i^T \preceq \frac{\varepsilon}{q}\left(u_jI-A_j\right) \preceq \frac{2\varepsilon}{q}\left(\hat{u}_iI-B_{i-1}\right).$$

This proves the first statement of the claim.

For the second statement, notice that

$$\ell_{j+1}-\ell_j = \Delta_{\ell,j} \leq \frac{\varepsilon N_j}{q\sum_{t=1}^m R_t(A_j,u_j,\ell_j)} \leq \frac{1}{2}\cdot\lambda_{\min}(A_j-\ell_jI),$$

and hence

$$w_iw_i^T \preceq \frac{\varepsilon}{q}\left(A_j-\ell_jI\right) \preceq \frac{2\varepsilon}{q}\left(A_j-\hat{\ell}_iI\right) \preceq \frac{2\varepsilon}{q}\left(B_{i-1}-\hat{\ell}_iI\right).$$

□

4.3.2 Analysis of the approximation guarantee

In this subsection we will prove that the algorithm produces a linear-sized $O(\varepsilon)$-spectral sparsifier. We assume that the algorithm finishes after k iterations, and will prove that the condition number of A_k is small, which follows from our setting of parameters.

Lemma 4.3.4. *The output matrix A_k has condition number at most $1+O(\varepsilon)$.*

Proof. Since the condition number of A_k is at most $\frac{u_k}{\ell_k} = \left(1-\frac{u_k-\ell_k}{u_k}\right)^{-1}$, it suffices to prove that $(u_k-\ell_k)/u_k = O(\varepsilon)$.

Since the increase rate of $\Delta_{u,j}-\Delta_{\ell,j}$ with respect to $\Delta_{u,j}$ for any iteration j is

$$\frac{\Delta_{u,j}-\Delta_{\ell,j}}{\Delta_{u,j}} = \frac{(1+2\varepsilon)-(1-2\varepsilon)}{1+2\varepsilon} = \frac{4\varepsilon}{1+2\varepsilon} \leq 4\varepsilon,$$

we have that

$$\frac{u_k - \ell_k}{u_k} = \frac{2 \cdot (2n)^{1/q} + \sum_{j=0}^{k-1} (\Delta_{u,j} - \Delta_{\ell,j})}{(2n)^{1/q} + \sum_{j=0}^{k-1} \Delta_{u,j}} \leq \frac{2 \cdot (2n)^{1/q} + \sum_{j=0}^{k-1} (\Delta_{u,j} - \Delta_{\ell,j})}{(2n)^{1/q} + (4\varepsilon)^{-1} \sum_{j=0}^{k-1} (\Delta_{u,j} - \Delta_{\ell,j})}.$$

By the ending condition of the algorithm, it holds that $u_k - \ell_k \geq 4 \cdot (2n)^{1/q}$, i.e.

$$\sum_{j=0}^{k-1} (\Delta_{u,j} - \Delta_{\ell,j}) \geq 2 \cdot (2n)^{1/q}.$$

Hence, it holds that

$$\frac{u_k - \ell_k}{u_k} \leq \frac{2 \cdot (2n)^{1/q} + 2 \cdot (2n)^{1/q}}{(2n)^{1/q} + (4\varepsilon)^{-1} 2 \cdot (2n)^{1/q}} \leq 8\varepsilon,$$

which finishes the proof. □

Now we prove that the algorithm finishes in $O\left(\frac{qn^{3/q}}{\varepsilon^2}\right)$ iterations, and picks $O\left(qn/\varepsilon^2\right)$ vectors in total.

Lemma 4.3.5. *The following statements hold:*

1. *With probability at least* $4/5$*, the algorithm finishes in* $\frac{10qn^{3/q}}{\varepsilon^2}$ *iterations.*
2. *With probability at least* $4/5$*, the algorithm chooses at most* $10qn/\varepsilon^2$ *vectors.*

Proof. Notice that after iteration j the barrier gap $u_j - \ell_j$ is increased by

$$\begin{aligned}\Delta_{u,j} - \Delta_{\ell,j} &= \frac{4\varepsilon^2}{q} \frac{N_j}{\sum_{i=1}^{m} R_i(A_j, u_j, \ell_j)} \\ &= \frac{4\varepsilon^2}{q} \frac{1}{n^{2/q}} \cdot \min\left\{\lambda_{\min}(u_j I - A_j), \lambda_{\min}(A_j - \ell_j I)\right\} \\ &\geq \frac{4\varepsilon^2}{q} \frac{1}{n^{2/q}} \cdot \left(\Phi_{u_j,\ell_j}(A_j)\right)^{-1/q}.\end{aligned}$$

Since the algorithm finishes within k iterations if $\sum_{j=0}^{k-1}(\Delta_{u,j} - \Delta_{\ell,j}) \geq 2 \cdot (2n)^{1/q}$, it holds that

$$\begin{aligned}\Pr\left[\text{algorithm finishes within } k \text{ iterations}\right] &\geq \Pr\left[\sum_{j=0}^{k-1}(\Delta_{u,j} - \Delta_{\ell,j}) \geq 2 \cdot (2n)^{1/q}\right] \\ &\geq \Pr\left[\sum_{j=0}^{k-1} \frac{4\varepsilon^2}{qn^{2/q}} \cdot \left(\Phi_{u_j,\ell_j}(A_j)\right)^{-1/q} \geq 2 \cdot (2n)^{1/q}\right] \\ &= \Pr\left[\sum_{j=0}^{k-1} \left(\Phi_{u_j,\ell_j}(A_j)\right)^{-1/q} \geq \frac{q}{2\varepsilon^2} \cdot \left(2n^3\right)^{1/q}\right] \\ &\geq \Pr\left[\sum_{j=0}^{k-1} \left(\Phi_{u_j,\ell_j}(A_j)\right)^{1/q} \leq 2 \cdot \frac{k^2\varepsilon^2}{q} \cdot \left(\frac{1}{2n^3}\right)^{1/q}\right],\end{aligned}$$

where the last inequality follows from the fact that

$$\left(\sum_{j=0}^{k-1}\left(\Phi_{u_j,\ell_j}(A_j)\right)^{-1/q}\right)\cdot\left(\sum_{j=0}^{k-1}\left(\Phi_{u_j,\ell_j}(A_j)\right)^{1/q}\right)\geq k^2.$$

By Lemma 4.3.1, every picked matrix W_j in iteration j satisfies $0\preceq W_j\preceq\frac{1}{2}\cdot(u_jI-A)$ with probability at least $1-\frac{\varepsilon^2}{100qn}$, and with probability $9/10$ all matrices picked in $k=10qn/\varepsilon^2$ iterations satisfy the condition above. Also, by Lemma 4.3.3 we have that

$$\sum_{j=0}^{k-1}(\Phi_{u_j,\ell_j}(A_j))^{1/q}=\sum_{j=0}^{k-1}(\Phi_{u_j,\ell_j}(A_j))^{1/q}\leq\sum_{j=0}^{k-1}\left(\Phi_{u_j,\ell_j}(A_j)\right)^{1/q}\leq k, \tag{4.16}$$

since the initial value of the potential function is at most 1. Therefore, it holds that

$$\begin{aligned}
&\Pr\left[\text{algorithm finishes in more than } k \text{ iterations}\right]\\
&\leq\Pr\left[\sum_{j=0}^{k-1}\left(\Phi_{u_j,\ell_j}(A_j)\right)^{1/q}\geq 2\cdot\frac{k^2\varepsilon^2}{q}\cdot\left(\frac{1}{2n^3}\right)^{1/q}\right]\\
&\leq\Pr\left[\sum_{j=0}^{k-1}\left(\Phi_{u_j,\ell_j}(A_j)\right)^{1/q}\geq 2\cdot\frac{k^2\varepsilon^2}{q}\cdot\left(\frac{1}{2n^3}\right)^{1/q}\text{ and }\forall j: W_j\preceq\frac{1}{2}(u_jI-A_j)\right]\\
&\quad+\Pr\left[\exists j: W_j\not\preceq\frac{1}{2}(u_jI-A_j)\right]\leq\frac{q}{2\cdot k\varepsilon^2}\cdot\left(2n^3\right)^{1/q}+1/10\leq 1/5,
\end{aligned}$$

where the second last inequity follows from Markov's inequality and (4.16), and the last inequality follows by our choice of k. This proves the first statement.

Now for the second statement. Notice that for every vector chosen in iteration j, the barrier gap $\Delta_{u,j}-\Delta_{\ell,j}$ is increased on average by

$$\frac{\Delta_{u,j}-\Delta_{\ell,j}}{N_j}=\frac{4\varepsilon^2}{q\sum_{i=1}^m R_i(A_j,u_j,\ell_j)}.$$

To bound $R_i(A_j,u_j,\ell_j)$, let the eigenvalues of matrix A_j be $\lambda_1,\cdots,\lambda_n$. Then, it holds that

$$\begin{aligned}
\sum_{i=1}^m R_i(\boldsymbol{A_j},u_j,\ell_j)&=\sum_{i=1}^m\vec{v}_i^T(u_j\boldsymbol{I}-A_j)^{-1}\boldsymbol{v}_i+\sum_{i=1}^m\vec{v}_i^T(\boldsymbol{A_j}-\ell_j\boldsymbol{I})^{-1}\vec{v}_i\\
&=\sum_{i=1}^n\frac{1}{u_j-\lambda_i}+\sum_{i=1}^n\frac{1}{\lambda_i-\ell_j}\\
&\leq\left(\sum_{i=1}^n(u_j-\lambda_i)^{-q}+\sum_{i=1}^n(\lambda_i-\ell_j)^{-q}\right)^{1/q}(2n)^{1-1/q}\\
&=\left(\Phi_{u_j,\ell_j}(\boldsymbol{A_j})\right)^{1/q}\cdot(2n)^{1-1/q}.
\end{aligned}$$

Therefore, we have that

$$\frac{\Delta_{u,j}-\Delta_{\ell,j}}{N_j}\geq\frac{4\varepsilon^2}{q}\cdot\frac{1}{(2n)^{1-1/q}\cdot(\Phi_{u_j,\ell_j}(A_j))^{1/q}}. \tag{4.17}$$

Let $v_1,\cdots,v_z$ be the vectors sampled by the algorithm, and v_j is picked in iteration τ_j, where

$1 \le j \le z$. We first assume that the algorithm could check the ending condition after adding every single vector. In such case, it holds that

$$\begin{aligned}
&\Pr\left[\text{algorithm finishes after choosing } z \text{ vectors}\right] \\
&\ge \Pr\left[\sum_{j=1}^{z} \frac{4\varepsilon^2}{q} \cdot \frac{1}{(2n)^{1-1/q} \cdot (\Phi_{u_{\tau_j},\ell_{\tau_j}}(A_{\tau_j}))^{1/q}} \ge 2 \cdot (2n)^{1/q}\right] \\
&= \Pr\left[\sum_{j=1}^{z} (\Phi_{u_{\tau_j},\ell_{\tau_j}}(A_{\tau_j}))^{-1/q} \ge qn/\varepsilon^2\right].
\end{aligned}$$

Following the same proof as the first part and noticing that in the final iteration the algorithm chooses at most $O(n)$ extra vectors, we obtain the second statement. □

4.3.3 Proof of the main results

Now we analyze the runtime of the algorithm, and prove the main results. We first analyze the algorithm for sparsifying the sum of rank-1 PSD matrices, and prove Theorem 4.1.1.

Proof of Theorem 4.1.1. By Theorem 3.7.3, we can first find a ε-approximation with $O(n \log n/\varepsilon^2)$ vectors in time $\tilde{O}(Z + n^\omega \varepsilon^{-2})$. Note that if $q < \log n$, we are done.

Otherwise, by Lemma 4.3.5, with probability at least $4/5$ the algorithm chooses at most $10qn/\varepsilon^2$ vectors, and by Lemma 4.3.4 the condition number of A_k is at most $1+O(\varepsilon)$, implying that the matrix A_k is a $(1+O(\varepsilon))$-approximation of I. These two results together prove that A_k is a linear-sized spectral sparsifier.

For the runtime, Lemma 4.3.5 proves that the algorithm finishes in $\frac{10qn^{3/q}}{\varepsilon^2}$ iterations, and it is easy to see that all the required quantities in each iteration can be approximately computed in $\tilde{O}(m \cdot n^{\omega-1})$ time using fast matrix multiplication. Therefore, the total runtime of the algorithm is $\tilde{O}\left(Z + n^\omega \varepsilon^{-2} + \frac{q \cdot m}{\varepsilon^2} \cdot n^{\omega-1+3/q}\right)$. Using $q < \log n$ and $m = O(n \log n/\varepsilon^2)$, we have

$$\tilde{O}\left(Z + n^{\omega+3/q}\varepsilon^{-4}\right).$$

□

Next we apply our algorithm in the graph setting, and prove Theorem 4.1.2.

Proof of Theorem 4.1.2. For any edge $e = \{u, v\}$ in graph $G = (V, E)$, we define $b_e \in \mathbb{R}^n$, where $b_e(w) = 1$ if $w = u$, $b_e(w) = -1$ if $w = v$, and $b_e(w) = 0$ otherwise. Then the Laplacian matrix of G can be written as $L = \sum_{e \in E[G]} b_e b_e^T$. By setting $v_e = L^{-1/2} b_e$ for $e \in E[G]$, it is easy to see that constructing a spectral sparsifier of G is equivalent to sparsifying the matrix $\sum_{e \in E[G]} v_e v_e^T$. We apply the same analysis as in the proof of Theorem 4.1.1, and hence the output is a linear-sized spectral sparsifier of graph G. So it suffices to analyze the runtime of the algorithm.

By Lemma 4.3.1 and the Union Bound, with probability at least $9/10$ all the matrices picked in $k = \frac{10qn^{3/q}}{\varepsilon^2}$ iterations satisfy $W_j \preceq \frac{1}{2}(u_j I - A_j)$. Conditioning on the event, with constant probability $\mathbf{E}\left[\Phi_{u_j,\ell_j}(A_j)\right] \le 2$ for all iterations j, and by Markov's inequality with high probability it holds that $\Phi_{u_j,\ell_j}(A_j) = O\left(qn/\varepsilon^2\right)$ for all iterations j.

On the other hand, notice that it holds for any $1 \le j \le n$ that

$$(u - \lambda_j)^{-q} \le \sum_{i=1}^{n} (u - \lambda_i)^{-q} < \Phi_{u,\ell}(\boldsymbol{A}),$$

where $\lambda_1, \ldots, \lambda_n$ are the eigenvalues of A. This implies that $\lambda_j < u - (\Phi_{u,\ell}(\boldsymbol{A}))^{-1/q}$. Similarly, it holds that $\lambda_j > \ell + (\Phi_{u,\ell}(\boldsymbol{A}))^{-1/q}$ for any $1 \le j \le n$. Therefore, we have that

$$\left(\ell_j + O\left(\left(\frac{\varepsilon^2}{qn}\right)^{1/q}\right)\right)\boldsymbol{I} \prec \boldsymbol{A_j} \prec \left(u_j - O\left(\left(\frac{\varepsilon^2}{qn}\right)^{1/q}\right)\right)\boldsymbol{I}.$$

Since both of u_j and ℓ_j are of the order $O(n^{1/q})$, we set $\eta = O\left((\varepsilon/n)^{2/q}\right)$ and obtain that

$$(\ell_j + |\ell_j|\eta)I \prec A_j \prec (1-\eta)u_j I.$$

Hence, we apply Lemma 4.4.3 and Lemma 4.4.4 to compute all required quantities in each iteration up to constant approximation in time

$$\tilde{O}\left(\frac{m}{\varepsilon^2 \cdot \eta}\right) = \tilde{O}\left(\frac{m \cdot n^{2/q}}{\varepsilon^{2+2/q}}\right).$$

Since by Lemma 4.3.5 the algorithm finishes in $\frac{10qn^{3/q}}{\varepsilon^2}$ iterations with probability at least $4/5$, the total runtime of the algorithm is

$$\tilde{O}\left(\frac{q \cdot m \cdot n^{5/q}}{\varepsilon^{4+4/q}}\right).$$

Again, we can assume $m = \tilde{O}(n \log n/\varepsilon^2)$ and $q < \log n$ by existing algorithm by using existing algorithm for graph sparsification. Hence, we have the promised runtime. □

4.4 Appendix

In this section we study fast approximation of the required quantities for constructing graph sparsifiers, and show that the number of sampled vectors N_j and the relative leverage scores $\{R_i(A_j, u_j, \ell_j)\}_{i=1}^m$ used in each iteration can be approximately computed in almost-linear time. For simplicity we drop the subscript j expressing the iterations in this subsection. We will assume that the following assumption holds on A, and show that the matrix under consideration always satisfies this condition.

Assumption 4.4.1. *Let L and $\tilde{L}$ be the Laplacian matrices of graph G and its subgraph after reweighting. Let $A = L^{-1/2}\tilde{L}L^{-1/2}$, and assume that*

$$(\ell + |\ell|\eta) \cdot I \prec A \prec (1-\eta)u \cdot I$$

holds for some $0 < \eta < 1$.

Lemma 4.4.2. *Under Assumption 4.4.1, the following statements hold:*

1. *We can construct a matrix S_u such that*

$$S_u \approx_{\varepsilon/10} (uI - A)^{-1/2},$$

and $S_u = p(A)$ for a polynomial p of degree $O\left(\frac{\log(1/\varepsilon\eta)}{\eta}\right)$.

2. *We can construct a matrix S_ℓ such that*

$$S_\ell \approx_{\varepsilon/10} (A - \ell I)^{-1/2}.$$

Moreover, S_ℓ is of the form $(A')^{-1/2}q((A')^{-1})$,where q is a polynomial of degree $O\left(\frac{\log(1/\varepsilon\eta)}{\eta}\right)$ and $A' = L^{-1/2}L'L^{-1/2}$ for some Laplacian matrix L'.

Proof. By Taylor expansion, it holds that

$$(1-x)^{-1/2} = 1 + \sum_{k=1}^{\infty} \prod_{j=0}^{k-1} \left(j + \frac{1}{2}\right) \frac{x^k}{k!}.$$

We define for any $T \in \mathbb{N}$ that

$$p_T(x) = 1 + \sum_{k=1}^{T} \prod_{j=0}^{k-1} \left(j + \frac{1}{2}\right) \frac{x^k}{k!}.$$

Then, it holds for any $0 < x < 1 - \eta$ that

$$\begin{aligned} p_T(x) \leq (1-x)^{-1/2} = p_T(x) + \sum_{k=T+1}^{\infty} \prod_{j=0}^{k-1} \left(j + \frac{1}{2}\right) \frac{x^k}{k!} \\ \leq p_T(x) + \sum_{k=T+1}^{\infty} x^k \\ \leq p_T(x) + \frac{(1-\eta)^{T+1}}{\eta}. \end{aligned}$$

Hence, it holds that

$$(uI - A)^{-1/2} = u^{-1/2}(I - u^{-1}A)^{-1/2} \succeq u^{-1/2} p_T(u^{-1}A),$$

and

$$(uI - A)^{-1/2} \preceq u^{-1/2} \left(p_T(u^{-1}A) + \frac{(1-\eta)^{T+1}}{\eta} \cdot I \right),$$

since $u^{-1}A \preceq (1-\eta)I$. Notice that $u^{-1/2}I \preceq (uI - A)^{-1/2}$, and therefore

$$(uI - A)^{-1/2} \preceq u^{-1/2} p_T(u^{-1}A) + \frac{(1-\eta)^{T+1}}{\eta} \cdot (uI - A)^{-1/2}.$$

Setting $T = \frac{c \log(1/(\varepsilon\eta))}{\eta}$ for some constant c and defining $S_u = u^{-1/2} p_T(u^{-1}A)$ gives us that

$$S_u \approx_{\varepsilon/10} (uI - A)^{-1/2}.$$

Now for the second statement. Our construction of S_ℓ is based on the case distinction ($\ell > 0$, and $\ell \leq 0$).

Case 1 ($\ell > 0$): Notice that

$$(A - \ell I)^{-1/2} = A^{-1/2}(I - \ell A^{-1})^{-1/2},$$

and

$$p_T(\ell A^{-1}) \preceq \left(I - \ell A^{-1}\right)^{-1/2} \preceq p_T\left(\ell A^{-1}\right) + \frac{(1-\eta/2)^{T+1}}{\eta/2} \cdot I.$$

Using the same analysis as before, we have that

$$A^{-1/2}(I - \ell A^{-1})^{-1/2} \approx_{\varepsilon/10} A^{-1/2} p_T(\ell A^{-1}).$$

By defining $S_\ell = A^{-1/2} p_T(\ell A^{-1})$, i.e., $A' = A$ and $q\left((A')^{-1}\right) = p_T(\ell A^{-1})$, we have that

$$S_\ell \approx_{\varepsilon/10} (A - \ell I)^{-1/2}.$$

Case 2 ($\ell \leq 0$): We look at the matrix

$$A - \ell I = L^{-1/2}\tilde{L}L^{-1/2} - \ell I = L^{-1/2}(\tilde{L} - \ell L)L^{-1/2}.$$

Notice that $\tilde{L} - \ell L$ is a Laplacian matrix, and hence this reduces to the case of $\ell = 0$, for which we simply set $S_\ell = (A - \ell I)^{-1/2}$. Therefore, we can write S_ℓ as a desired form, where $A' = A - \ell I$ and polynomial $q = 1$. □

The following two lemmas present nearly-linear time algorithms for computing the required quantities within each iteration. We remark that an almost-linear time algorithm for computing similar quantities was shown in [9].

Lemma 4.4.3. *Let $A = \sum_{i=1}^m v_i v_i^T$, and suppose that A satisfies Assumption 4.4.1. Then, we can compute $\{r_i\}_{i=1}^m$ and $\{t_i\}_{i=1}^m$ in $\tilde{O}\left(\frac{m}{\varepsilon^2\eta}\right)$ time such that*

$$(1-\varepsilon)r_i \leq v_i^T(uI - A)^{-1}v_i \leq (1+\varepsilon)r_i,$$

and

$$(1-\varepsilon)t_i \leq v_i^T(A - \ell I)^{-1}v_i \leq (1+\varepsilon)t_i.$$

Proof. Define $u_i = L^{1/2}v_i$ for any $1 \leq i \leq m$. By Lemma 4.4.2, we have that

$$\begin{aligned} v_i^T(uI - A)^{-1}v_i &\approx_{3\varepsilon/10} \|p(A)v_i\|^2 \\ &= \left\| p\left(L^{-1/2}\tilde{L}L^{-1/2}\right) L^{-1/2}u_i \right\|^2 \\ &= \left\| L^{1/2} p\left(L^{-1}\tilde{L}\right) L^{-1}u_i \right\|^2. \end{aligned}$$

Let $L = B^T B$ for some $B \in \mathbb{R}^{m\times n}$. Then, it holds that

$$v_i^T(uI - A)^{-1}v_i \approx_{3\varepsilon/10} \left\| Bp\left(L^{-1}\tilde{L}\right) L^{-1}u_i \right\|^2.$$

We invoke the Johnson-Lindenstrauss Lemma and find a random matrix $Q \in \mathbb{R}^{O(\log n/\varepsilon^2)\times m}$: With high probability, it holds that

$$v_i^T(uI - A)^{-1}v_i \approx_{4\varepsilon/10} \left\| QBp\left(L^{-1}\tilde{L}\right) L^{-1}u_i \right\|^2.$$

We apply a nearly-linear time Laplacian solver to compute $\left\| QBp\left(L^{-1}\tilde{L}\right) L^{-1}u_i \right\|^2$ for all $\{u_i\}_{i=1}^m$ up to $(1 \pm \varepsilon/10)$-multiplicative error in time $\tilde{O}\left(\frac{m}{\varepsilon^2\eta}\right)$. This gives the desired $\{r_i\}_{i=1}^m$.

The computation for $\{t_i\}_{i=1}^m$ is similar. By Lemma 4.4.2, it holds for any $1 \leq i \leq m$ that

$$\begin{aligned} v_i^T(A - \ell I)^{-1}v_i &\approx_{3\varepsilon/10} \left\| (A')^{-1/2} q((A')^{-1})v_i \right\|^2 \\ &= \left\| (A')^{-1/2} q\left(L^{1/2}(L')^{-1}L^{1/2}\right) L^{-1/2}u_i \right\|^2 \\ &= \left\| (A')^{-1/2}L^{-1/2} q(L(L')^{-1})u_i \right\|^2. \end{aligned}$$

Let $L' = (B')^T(B')$ for some $B' \in \mathbb{R}^{m\times n}$. Then, it holds that

$$\begin{aligned} v_i^T(A-\ell I)^{-1}v_i &\approx_{3\varepsilon/10} \left\|(L')^{-1/2}q\left(L(L')^{-1}\right)u_i\right\|^2 \\ &= \left\|(L')^{1/2}(L')^{-1}q\left(L(L')^{-1}\right)u_i\right\|^2 \\ &= \left\|B'(L')^{-1}q\left(L(L')^{-1}\right)u_i\right\|^2. \end{aligned}$$

We invoke the Johnson-Lindenstrauss Lemma and a nearly-linear time Laplacian solver as before to obtain required $\{t_i\}_{i=1}^m$. The total runtime is $\tilde{O}\left(\frac{m}{\eta\varepsilon^2}\right)$. □

Lemma 4.4.4. *Under Assumption 4.4.1, we can compute values α, β in $\tilde{O}\left(\frac{m}{\eta\varepsilon^3}\right)$ time such that*

$$(1-\varepsilon)\alpha \le \lambda_{\min}(uI - A) \le (1+\varepsilon)\alpha$$

and

$$(1-\varepsilon)\beta \le \lambda_{\min}(A - \ell I) \le (1+\varepsilon)\beta.$$

Proof. By Lemma 4.4.2, we have that $S_u \approx_{\varepsilon/10} (uI-A)^{-1/2}$. Hence, $\lambda_{\max}(S_u)^{-2} \approx_{3\varepsilon/10} \lambda_{\min}(uI-A)$, and it suffices to estimate $\lambda_{\max}(S_u)$. Since

$$\lambda_{\max}(S_u) \le \left(\mathrm{Tr}\left(S_u^{2k}\right)\right)^{1/2k} \le n^{1/2k}\lambda_{\max}(S_u),$$

by picking $k = \log n/\varepsilon$ we have that $\left(\mathrm{Tr}(S_u^{2k})\right)^{1/2k} \approx_{\varepsilon/2} \lambda_{\max}(S_u)$. Notice that

$$\mathrm{Tr}\left(S_u^{2k}\right) = \mathrm{Tr}\left(p^{2k}\left(L^{-1/2}\tilde{L}L^{-1/2}\right)\right) = \mathrm{Tr}\left(p^{2k}\left(L^{-1}\tilde{L}\right)\right).$$

Set $\tilde{L} = \tilde{B}^T\tilde{B}$ for some matrix $\tilde{B} \in \mathbb{R}^{m\times n}$, and we have that

$$\mathrm{Tr}\left(S_u^{2k}\right) = \mathrm{Tr}\left(p^{2k}\left(\tilde{B}L^{-1}\tilde{B}^T\right)\right).$$

Since we can apply $p^k\left(\tilde{B}L^{-1}\tilde{B}^T\right)$ to vectors in $\tilde{O}\left(\frac{m}{\eta\varepsilon}\right)$ time, we invoke the Johnson-Lindenstrauss Lemma and approximate $\mathrm{Tr}\left(S_u^{2k}\right)$ in $\tilde{O}\left(\frac{m}{\eta\varepsilon^3}\right)$ time.

We approximate $\lambda_{\min}(A-\ell I)$ in a similar way. Notice that

$$\begin{aligned} \mathrm{Tr}\left(S_\ell^{4k}\right) &= \mathrm{Tr}\left((A')^{-1/2}q((A')^{-1})\right)^{4k} \\ &= \mathrm{Tr}\left(q((A')^{-1})(A')^{-1}q((A')^{-1})\right)^{2k}. \end{aligned}$$

Let z be a polynomial defined by $z(x) = xq^2(x)$ and $L' = (B')^T(B')$. Then, we have that

$$\mathrm{Tr}(S_\ell^{4k}) = \mathrm{Tr}\left(z^{2k}((A')^{-1})\right) = \mathrm{Tr}\left(z^{2k}\left(L^{1/2}(L')^{-1}L^{1/2}\right)\right).$$

Applying the same analysis as before, we can estimate the trace in $\tilde{O}\left(\frac{m}{\eta\varepsilon^3}\right)$ time. □

Chapter 5

ℓ_p Regression and John Ellipsoid

5.1 Introduction

In this chapter, we study sparsification for ℓ_p regression problems.

For ℓ_2 regression, Lemma 2.3.3 shows that given a tall matrix, we can sample $O(n \log n)$ rows of $\boldsymbol{A}$ to form a matrix $\widetilde{\boldsymbol{A}}$ such that, for all vectors $\boldsymbol{x}$, $\|\widetilde{\boldsymbol{A}}\boldsymbol{x}\|_2 \approx \|\boldsymbol{A}\boldsymbol{x}\|_2$. Similarly, for ℓ_p regression, it is known that an approximation can be obtained by sampling rows with a certain probability (called ℓ_p Lewis weights). The main goal of this chapter is to show how to approximate ℓ_p Lewis weights by solving $\widetilde{O}(1)$ linear systems (Theorem 5.3.4). Since we know how to solve linear systems in input sparsity time, this allows us to compute ℓ_p Lewis Weights in nearly input sparsity time.

This chapter is organized as follows: In Section 5.2, we define ℓ_p Lewis weights; in Section 5.3, we show how to approximate ℓ_p Lewis weights efficiently; in Section 5.4, we discuss some geometric implications of ℓ_p Lewis weights.

5.2 Lewis Weights

Unlike leverage scores, ℓ_p Lewis weights do not have closed formulas and are defined by the following non-linear equation.

Definition 5.2.1. For a matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and $1 \leq p < \infty$, the ℓ_p Lewis weight is the unique vector $\boldsymbol{w} \in \mathbb{R}^m$ such that

$$\boldsymbol{w}_i = \boldsymbol{\sigma}_i(\boldsymbol{W}^{1/2-1/p}\boldsymbol{A}) \quad \text{for all } i \in [m]$$

where $\boldsymbol{\sigma}_i(\boldsymbol{A}) = (\boldsymbol{A}(\boldsymbol{A}^\top \boldsymbol{A})^{-1}\boldsymbol{A})_{ii}$.

The following result show how to use ℓ_p Lewis weights to construct sparsifiers for ℓ_p regression. They follow from a collection of results for Banach spaces (See [56] for the summary of each result).

Theorem 5.2.2 ([169, 38, 245, 246, 56]). *Let $\boldsymbol{w}$ be the ℓ_p Lewis weight of some given matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and $1 \leq p < \infty$. Given an error parameter $0 < \varepsilon < 1$ and sampling values $\boldsymbol{p}$ that*

$$\boldsymbol{p}_i \geq f(p, n, \varepsilon, \delta)\boldsymbol{w}_i$$

p	Success probability: $1-\delta$	Number of rows: $n \cdot f(p,n,\varepsilon,\delta)$
$p=1$	$1-1/n^C$	$O(n\log(n/\varepsilon)/\varepsilon^2)$
$p=1$	$1-1/C$	$O(n\log n/\varepsilon^2)$
$1<p<2$	$1-1/C$	$O(n\log(n/\varepsilon)\log(\log n/\varepsilon)^2/\varepsilon^2)$
$p>2$	$1-1/n^C$	$O(n^{p/2}\log n\log(1/\varepsilon)/\varepsilon^5)$

Table 5.1: $n \cdot f(p,n,\varepsilon,\delta)$: Number of rows needed for ℓ_p sampling. The constants in big O notation depend on C.

where $f(p,n,\varepsilon,\delta)$ is shown in the Table 5.1. Let $\boldsymbol{S}$ be the diagonal matrix with independently chosen entries. $\boldsymbol{S}_{ii} = \frac{1}{\boldsymbol{p}_i^{1/p}}$ with probability $\boldsymbol{p}_i$ and 0 otherwise. Then, with probability at least $1-\delta$, we have that

$$(1-\varepsilon)\|\boldsymbol{A}\boldsymbol{x}\|_p \leq \|\boldsymbol{S}\boldsymbol{A}\boldsymbol{x}\|_p \leq (1+\varepsilon)\|\boldsymbol{A}\boldsymbol{x}\|_p$$

for all $\boldsymbol{x} \in \mathbb{R}^n$.

This theorem shows if we have a good upper estimate of ℓ_p Lewis weights, then we can simple $\widetilde{O}(\max(n, n^{p/2}))$ rows to approximate the original matrix. Since $\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_p = \|[\boldsymbol{A}, \boldsymbol{b}]\begin{pmatrix}\boldsymbol{x}\\ -1\end{pmatrix}\|_p$, Lewis weights allow us to reduce a ℓ_p regression problem involving a tall matrix to a matrix with only $\widetilde{O}(\max(n, n^{p/2}))$ rows.

5.3 Approximate Lewis Weights

In this section, we show how to compute ℓ_p Lewis weights by solving $\widetilde{O}(1)$ linear systems for $1 \leq p < \infty$. For $p < 4$, this can be done by the following algorithm [56].

Algorithm 5: Approximate ℓ_p Lewis weights for $p < 4$

Input: a matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, accuracy $0 < \varepsilon < 1$, $1 \leq p < 4$.

Let $\boldsymbol{v}_i^{(1)} = n/m$ for all $i \in [m]$ and $T = \Theta\left(\frac{\log(m/\varepsilon)}{4-p}\right)$.

for $k = 1, \cdots, T$ **do**

Let $\tilde{\boldsymbol{\sigma}}$ be a $c_1(4-p)\varepsilon$-approximation of $\boldsymbol{\sigma}_i\left(\left(\boldsymbol{V}^{(k)}\right)^{1/2-1/p}\boldsymbol{A}\right)$ for some small enough constant c_1.

$\boldsymbol{v}_i^{(k+1)} = \left(\left(\boldsymbol{v}_i^{(k)}\right)^{2/p-1}\tilde{\boldsymbol{\sigma}}_i\right)^{p/2}$.

end

Output: $\boldsymbol{v}^{(T+1)}$.

Theorem 5.3.1 ([56]). *For any matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, accuracy $0 < \varepsilon < 1$ and $1 \leq p < 4$, the algorithm 5 outputs a vector $\boldsymbol{v}$ in time $\widetilde{O}\left(\mathcal{T}\varepsilon^{-2}\right)$ with high probability such that*

$$(1-\varepsilon)\boldsymbol{w}_i \leq \boldsymbol{v}_i \leq (1+\varepsilon)\boldsymbol{w}_i \tag{5.1}$$

where $\boldsymbol{w}$ is the ℓ_p Lewis weight for $\boldsymbol{A}$ and $\mathcal{T}$ be the time needed to apply $(\boldsymbol{A}^\top\boldsymbol{D}\boldsymbol{A})^{-1}$ to a vector for any diagonal matrix $\boldsymbol{D}$.

This algorithm relies on the stability of ℓ_p Lewis weights for $1 \leq p < 4$, namely, if we rescale each row of $\boldsymbol{A}$ by a constant, the Lewis weight of $\boldsymbol{A}$ changes only by a constant. However, this is not true for large p. For $p \sim \infty$, the Lewis weights can change arbitrarily even with a tiny rescaling and hence it is almost impossible to approximate Lewis weight in the sense of (5.1). To avoid this inability issue, we define the ε-approximate ℓ^p Lewis weight as follows:

Definition 5.3.2. For a matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ and $1 \leq p < \infty$, we call a vector $\boldsymbol{w} \in \mathbb{R}^m$ is an ε-approximate ℓ^p Lewis weight for $\boldsymbol{A}$ if

$$(1-\varepsilon)\boldsymbol{\sigma}_i(\boldsymbol{W}^{1/2-1/p}\boldsymbol{A}) \leq \boldsymbol{w}_i \leq (1+\varepsilon)\boldsymbol{\sigma}_i(\boldsymbol{W}^{1/2-1/p}\boldsymbol{A}).$$

For $1 \leq p < 4$, an ε-approximate ℓ^p Lewis weight is indeed a $O(\varepsilon)$ approximate of the true ℓ^p Lewis weight in the sense of (5.1). Although this is not true for larger p, the following lemma shows that ε-approximate ℓ^p Lewis weights suffice for the sampling purpose.

Lemma 5.3.3. *Given an error parameter $0 < \varepsilon < \frac{1}{2}$. Let $\boldsymbol{w}$ be an ε-approximate ℓ_p Lewis weight of some given matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ and $1 \le p < \infty$. Let $\boldsymbol{p}$ be the sampling values such that*

$$\boldsymbol{p}_i \ge f(p, n, \varepsilon, \delta)\boldsymbol{w}_i$$

where $f(p, n, \varepsilon, \delta)$ is shown in the Table 5.1. Let $\boldsymbol{S}$ be the diagonal matrix with independently chosen entries. $\boldsymbol{S}_{ii} = \frac{1}{\boldsymbol{p}_i^{1/p}}$ with probability $\boldsymbol{p}_i$ and 0 otherwise. Then, with probability at least $1 - \delta$, we have that

$$(1 - 2\varepsilon)\|\boldsymbol{A}\boldsymbol{x}\|_p \le \|\boldsymbol{S}\boldsymbol{A}\boldsymbol{x}\|_p \le (1 + 2\varepsilon)\|\boldsymbol{A}\boldsymbol{x}\|_p$$

for all $\boldsymbol{x} \in \mathbb{R}^n$.

Proof. Let $\boldsymbol{d}_i = \boldsymbol{\sigma}_i(\boldsymbol{W}^{1/2-1/p}\boldsymbol{A})/\boldsymbol{w}_i$, $\widetilde{\boldsymbol{A}} = \boldsymbol{D}^{-1/2+1/p}\boldsymbol{A}$ and $\widetilde{\boldsymbol{W}} = \boldsymbol{D}\boldsymbol{W}$. From the definition of $\boldsymbol{\sigma}_i$ (Definition 5.2.1), we see that

$$\begin{aligned}\boldsymbol{\sigma}_i(\widetilde{\boldsymbol{W}}^{1/2-1/p}\widetilde{\boldsymbol{A}}) &= \boldsymbol{\sigma}_i(\boldsymbol{W}^{1/2-1/p}\boldsymbol{A}) \\ &= \boldsymbol{d}_i\boldsymbol{w}_i = \widetilde{\boldsymbol{w}}_i.\end{aligned}$$

Hence, $\widetilde{\boldsymbol{w}}$ is the ℓ_p Lewis weight of $\widetilde{\boldsymbol{A}}$ and theorem 5.2.2 shows that

$$(1 - \varepsilon)\|\widetilde{\boldsymbol{A}}\boldsymbol{x}\|_p \le \|\boldsymbol{S}\widetilde{\boldsymbol{A}}\boldsymbol{x}\|_p \le (1 + \varepsilon)\|\widetilde{\boldsymbol{A}}\boldsymbol{x}\|_p.$$

From the definition of ε-approximate ℓ_p Lewis weight, we know that $1 - \varepsilon \le \boldsymbol{d}_i \le 1 + \varepsilon$ and hence

$$(1 - 2\varepsilon)\|\boldsymbol{A}\boldsymbol{x}\|_p \le \|\boldsymbol{S}\boldsymbol{A}\boldsymbol{x}\|_p \le (1 + 2\varepsilon)\|\boldsymbol{A}\boldsymbol{x}\|_p.$$

□

Note that if we have a constant approximate of leverage scores, we can construct an arbitrarily good approximation by sampling more rows. However, this is not true for ℓ_p Lewis weights; we need a good approximation of Lewis weight to get a good approximate of the original matrix.

Now, we have everything to describe our result, a simple multiplicative weight update algorithm.

Algorithm 6: Approximate ℓ_p Lewis weights for $p \ge 2$

Input: a matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, accuracy $0 < \varepsilon < 1$, $p \ge 2$.

Let $\boldsymbol{v}_i^{(1)} = n/m$ for all $i \in [m]$ and $T = \Theta\left(\frac{\log(m/n)}{\varepsilon}\right)$.

for $k = 1, \cdots, T$ **do**

  Let $\tilde{\boldsymbol{\sigma}}$ be a $\frac{\varepsilon}{4}$-approximation of $\boldsymbol{\sigma}_i\left(\left(\boldsymbol{V}^{(k)}\right)^{1/2-1/p}\boldsymbol{A}\right)$.

  $\boldsymbol{v}_i^{(k+1)} = \tilde{\boldsymbol{\sigma}}_i$.

end

Output: $\frac{1}{T}\sum_{k=1}^{T}\boldsymbol{v}^{(k)}$.

Theorem 5.3.4. *For any matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, accuracy $0 < \varepsilon < 1$ and $p \ge 2$, the algorithm 6 outputs an ε-approximate ℓ_p Lewis weight in time $\widetilde{O}(\mathcal{T}\varepsilon^{-3})$ where $\mathcal{T}$ be the time needed to apply $(\boldsymbol{A}^\top\boldsymbol{D}\boldsymbol{A})^{-1}$ to a vector for any diagonal matrix $\boldsymbol{D}$.*

Since we know how to solve linear systems in input sparsity time (Chapter 3), this theorem gives an input sparsity time algorithm for approximating Lewis weights. This algorithm is based on the following convexity result.

Lemma 5.3.5. *For any matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, $i \in [m]$ and $p \geq 2$, the function*

$$\phi_i(\boldsymbol{w}) = \log(\boldsymbol{a}_i^\top (\boldsymbol{A}^\top \boldsymbol{W}^{1-2/p}\boldsymbol{A})^{-1}\boldsymbol{a}_i \boldsymbol{w}_i^{-2/p})$$

is convex.

Proof. For any vector $\boldsymbol{h} \in \mathbb{R}^m$, we define $\phi_i(t) = \phi_i(\boldsymbol{w} + t\boldsymbol{h})$. Let $\boldsymbol{M} = \left(\boldsymbol{A}^\top \boldsymbol{W}^{1-2/p}\boldsymbol{A}\right)^{-1}$, we have that

$$\begin{aligned}
\left.\frac{d}{dt}\right|_{t=0} \phi_i(t) &= -\frac{2}{p}\frac{\boldsymbol{h}_i}{\boldsymbol{w}_i} - \frac{(1-2/p)\boldsymbol{a}_i^\top \boldsymbol{M}\boldsymbol{A}^\top \boldsymbol{H}\boldsymbol{W}^{-2/p}\boldsymbol{A}\boldsymbol{M}\boldsymbol{a}_i}{\boldsymbol{a}_i^\top \boldsymbol{M}\boldsymbol{a}_i} \\
\left.\frac{d^2}{dt^2}\right|_{t=0} \phi_i(t) &= \frac{2}{p}\frac{\boldsymbol{h}_i^2}{\boldsymbol{w}_i^2} - \frac{(1-2/p)^2\left(\boldsymbol{a}_i^\top \boldsymbol{M}\boldsymbol{A}^\top \boldsymbol{H}\boldsymbol{W}^{-2/p}\boldsymbol{A}\boldsymbol{M}\boldsymbol{a}_i\right)^2}{(\boldsymbol{a}_i^\top \boldsymbol{M}\boldsymbol{a}_i)^2} \\
&\quad - \frac{(-2/p)(1-2/p)\boldsymbol{a}_i^\top \boldsymbol{M}\boldsymbol{A}^\top \boldsymbol{H}^2\boldsymbol{W}^{-1-2/p}\boldsymbol{A}\boldsymbol{M}\boldsymbol{a}_i}{\boldsymbol{a}_i^\top \boldsymbol{M}\boldsymbol{a}_i} \\
&\quad + \frac{2(1-2/p)^2\boldsymbol{a}_i^\top \boldsymbol{M}\boldsymbol{A}^\top \boldsymbol{H}\boldsymbol{W}^{-2/p}\boldsymbol{A}\boldsymbol{M}\boldsymbol{A}^\top \boldsymbol{H}\boldsymbol{W}^{-2/p}\boldsymbol{A}\boldsymbol{M}\boldsymbol{a}_i}{\boldsymbol{a}_i^\top \boldsymbol{M}\boldsymbol{a}_i}.
\end{aligned}$$

By Cauchy-Schwarz inequality, we have that

$$\left(\boldsymbol{a}_i^\top \boldsymbol{M}\boldsymbol{A}^\top \boldsymbol{H}\boldsymbol{W}^{-2/p}\boldsymbol{A}\boldsymbol{M}\boldsymbol{a}_i\right)^2 \leq \left(\boldsymbol{a}_i^\top \boldsymbol{M}\boldsymbol{A}^\top \boldsymbol{H}\boldsymbol{W}^{-2/p}\boldsymbol{A}\boldsymbol{M}\boldsymbol{A}^\top \boldsymbol{H}\boldsymbol{W}^{-2/p}\boldsymbol{A}\boldsymbol{M}\boldsymbol{a}_i\right)\left(\boldsymbol{a}_i^\top \boldsymbol{M}\boldsymbol{a}_i\right)$$

and hence

$$\begin{aligned}
\left.\frac{d^2}{dt^2}\right|_{t=0} \phi_i(t) &\geq \frac{2}{p}\left(\frac{\boldsymbol{h}_i^2}{\boldsymbol{w}_i^2} + \frac{(1-2/p)\boldsymbol{a}_i^\top \boldsymbol{M}\boldsymbol{A}^\top \boldsymbol{H}^2\boldsymbol{W}^{-1-2/p}\boldsymbol{A}\boldsymbol{M}\boldsymbol{a}_i}{\boldsymbol{a}_i^\top \boldsymbol{M}\boldsymbol{a}_i}\right) \\
&\quad + \frac{(1-2/p)^2\boldsymbol{a}_i^\top \boldsymbol{M}\boldsymbol{A}^\top \boldsymbol{H}\boldsymbol{W}^{-2/p}\boldsymbol{A}\boldsymbol{M}\boldsymbol{A}^\top \boldsymbol{H}\boldsymbol{W}^{-2/p}\boldsymbol{A}\boldsymbol{M}\boldsymbol{a}_i}{\boldsymbol{a}_i^\top \boldsymbol{M}\boldsymbol{a}_i} \\
&\geq 0.
\end{aligned}$$

Since this holds for any any vector $\boldsymbol{h} \in \mathbb{R}^m$, this shows that ϕ_i is convex. □

Now, we prove Theorem 5.3.4.

Proof of Theorem 5.3.4. Let $\overline{\boldsymbol{v}} = \frac{1}{T}\sum_{k=1}^T \boldsymbol{v}_i^{(k)}$. Let $\boldsymbol{d}_i^{(k)} = \tilde{\boldsymbol{\sigma}}_i/\boldsymbol{\sigma}_i\left(\left(\boldsymbol{V}^{(k)}\right)^{1/2-1/p}\boldsymbol{A}\right)$. Then, we have that

$$\begin{aligned}
\boldsymbol{v}_i^{(k+1)} &= \boldsymbol{d}_i^{(k)}\boldsymbol{\sigma}_i\left(\left(\boldsymbol{V}^{(k)}\right)^{1/2-1/p}\boldsymbol{A}\right) \\
&= \boldsymbol{d}_i^{(k)}\left(\boldsymbol{v}_i^{(k)}\right)^{1-2/p}\boldsymbol{a}_i^\top\left(\boldsymbol{A}^\top\left(\boldsymbol{V}^{(k)}\right)^{1-2/p}\boldsymbol{A}\right)^{-1}\boldsymbol{a}_i.
\end{aligned}$$

Hence, we have

$$\log \boldsymbol{v}_i^{(k+1)} = \log \boldsymbol{v}_i^{(k)} + \log\left(\boldsymbol{a}_i^\top\left(\boldsymbol{A}^\top\left(\boldsymbol{V}^{(k)}\right)^{1-2/p}\boldsymbol{A}\right)^{-1}\boldsymbol{a}_i\left(\boldsymbol{v}_i^{(k)}\right)^{-2/p}\right) + \log(\boldsymbol{d}_i^{(k)}).$$

Using $\boldsymbol{v}_i^{(1)} = \frac{n}{m}$ and $\boldsymbol{v}_i^{(T+1)} \leq 1$, we have that

$$\begin{aligned} 0 &\geq \log \boldsymbol{v}_i^{(T+1)} \\ &= \log \boldsymbol{v}_i^{(1)} + \sum_{k=1}^{T} \log \left(\boldsymbol{a}_i^\top \left(\boldsymbol{A}^\top \left(\boldsymbol{V}^{(k)} \right)^{1-2/p} \boldsymbol{A} \right)^{-1} \boldsymbol{a}_i \left(\boldsymbol{v}_i^{(k)} \right)^{-2/p} \right) + \sum_{k=1}^{T} \log(\boldsymbol{d}_i^{(k)}). \end{aligned}$$

By Lemma 5.3.5, we have that

$$\log(\boldsymbol{a}_i^\top (\boldsymbol{A}^\top \overline{\boldsymbol{V}}^{1-2/p} \boldsymbol{A})^{-1} \boldsymbol{a}_i \overline{\boldsymbol{v}}_i^{-2/p}) \leq \frac{1}{T} \sum_{k=1}^{T} \log \left(\boldsymbol{a}_i^\top \left(\boldsymbol{A}^\top \left(\boldsymbol{V}^{(k)} \right)^{1-2/p} \boldsymbol{A} \right)^{-1} \boldsymbol{a}_i \left(\boldsymbol{v}_i^{(k)} \right)^{-2/p} \right).$$

Hence, we have that

$$\begin{aligned} \log \left(\frac{\boldsymbol{\sigma}_i(\overline{\boldsymbol{V}}^{1/2-1/p} \boldsymbol{A})}{\overline{\boldsymbol{v}}_i} \right) &= \log(\boldsymbol{a}_i^\top (\boldsymbol{A}^\top \overline{\boldsymbol{V}}^{1-2/p} \boldsymbol{A})^{-1} \boldsymbol{a}_i \overline{\boldsymbol{v}}_i^{-2/p}) \\ &\leq \frac{\log\left(\frac{m}{n}\right)}{T} - \frac{1}{T} \sum_{i=1}^{T} \log(\boldsymbol{d}_i^{(k)}) \\ &\leq \frac{\log\left(\frac{m}{n}\right)}{T} + \frac{1}{2}\varepsilon. \end{aligned}$$

By setting T large enough, we have that

$$(1-\varepsilon)\boldsymbol{\sigma}_i(\overline{\boldsymbol{V}}^{1/2-1/p} \boldsymbol{A}) \leq \overline{\boldsymbol{v}}_i \leq (1+\varepsilon)\boldsymbol{\sigma}_i(\overline{\boldsymbol{V}}^{1/2-1/p} \boldsymbol{A}).$$

□

5.4 Approximate John Ellipsoids

In this section, we discuss some geometric implications of ℓ_p Lewis weight. [270] shows that the ℓ_p Lewis weight is given by the following optimization problem.

Lemma 5.4.1. *Given a matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$. Let $\boldsymbol{M}$ be the maximizer of the following optimization problem*

$$\max_{\boldsymbol{M}} \log\det \boldsymbol{M} \text{ subject to } \sum_{i=1}^{m} \left(\boldsymbol{a}_i^\top \boldsymbol{M} \boldsymbol{a}_i \right)^{p/2} \leq n \text{ and } \boldsymbol{M} \succeq \boldsymbol{0}. \tag{5.2}$$

Then, the ℓ_p Lewis weight of $\boldsymbol{A}$ is given by $\boldsymbol{w}_i = \left(\boldsymbol{a}_i^\top \boldsymbol{M} \boldsymbol{a}_i \right)^{p/2}$. Furthermore, we have that $\boldsymbol{M}^{-1} = \sum_i \boldsymbol{w}_i^{1-2/p} \boldsymbol{a}_i \boldsymbol{a}_i^\top$.

To understand this optimization problem, we define the ℓ_p Lewis Ellipsoids and John ellipsoids as follows:

Definition 5.4.2. For any matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, we define the ℓ_p Lewis Ellipsoid of $\boldsymbol{A}$ by $E = \{\boldsymbol{x} \in \mathbb{R}^n \text{ such that } \boldsymbol{x}^\top \boldsymbol{A}^\top \boldsymbol{W}^{1-2/p} \boldsymbol{A} \boldsymbol{x} \leq 1\}$ where $\boldsymbol{w}$ is the ℓ_p Lewis weight of $\boldsymbol{A}$. For any convex set K, we define the John ellipsoid of K is the maximum volume ellipsoid contained inside K

We note that $\sum_{i=1}^{m} \left(\boldsymbol{a}_i^\top \boldsymbol{M} \boldsymbol{a}_i \right)^{p/2}$ is proportional to the average value of $\|\boldsymbol{A}\boldsymbol{x}\|_p^p$ inside the ellipsoid $\left\{ x^\top \boldsymbol{M}^{-1} \boldsymbol{x} \leq 1 \right\}$. Therefore, the ℓ_p Lewis Ellipsoids is the maximum volume ellipsoid E such that the average value of $\|\boldsymbol{A}\boldsymbol{x}\|_p^p$ inside that ellipsoid is less than certain value. In particular, the ℓ_∞ Lewis

Ellipsoid of $\boldsymbol{A}$ exactly corresponds to the John ellipsoid of the symmetric polytope $\{\|\boldsymbol{A}\boldsymbol{x}\|_\infty \leq 1\}$. Therefore, our algorithm for computing Lewis weights gives an efficient algorithm for approximating John ellipsoids for symmetric polytopes.

Lemma 5.4.3. *Given a symmetric polytope $P = \{\boldsymbol{x}$ such that $\|\boldsymbol{A}\boldsymbol{x}\|_\infty \leq 1\}$. Then, we can find $\boldsymbol{v}$ such that*

$$\boldsymbol{\sigma}_i(\boldsymbol{V}^{1/2}\boldsymbol{A}) \leq \boldsymbol{v}_i \leq (1+\varepsilon)\boldsymbol{\sigma}_i(\boldsymbol{V}^{1/2}\boldsymbol{A})$$

in time $\widetilde{O}(\mathcal{T}\varepsilon^{-3})$ where $\mathcal{T}$ be the time needed to apply $(\boldsymbol{A}^\top \boldsymbol{D}\boldsymbol{A})^{-1}$ to a vector for any diagonal matrix $\boldsymbol{D}$. Furthermore, the ellipsoid $E = \{\boldsymbol{x} \in \mathbb{R}^n$ such that $\boldsymbol{x}^\top \boldsymbol{A}^\top \boldsymbol{V}\boldsymbol{A}\boldsymbol{x} \leq 1\}$ satisfies the rounding condition

$$E \subset P \subset \sqrt{(1+\varepsilon)n}E. \tag{5.3}$$

Proof. By Lemma 5.3.4, we can find $\boldsymbol{v}$ such that $(1-\frac{\varepsilon}{3})\boldsymbol{\sigma}_i(\boldsymbol{V}^{1/2}\boldsymbol{A}) \leq \boldsymbol{v}_i \leq (1+\frac{\varepsilon}{3})\boldsymbol{\sigma}_i(\boldsymbol{V}^{1/2}\boldsymbol{A})$ in time $\widetilde{O}(\mathcal{T}\varepsilon^{-3})$. By scaling $\boldsymbol{v}$ up $1/(1-\varepsilon/3)$, we get the first guarantee.

By $\boldsymbol{\sigma}_i(\boldsymbol{V}^{1/2}\boldsymbol{A}) \leq \boldsymbol{v}_i$, we know that for all i

$$\boldsymbol{a}_i^\top \left(\boldsymbol{A}^\top \boldsymbol{V}\boldsymbol{A}\right)^{-1} \boldsymbol{a}_i \leq 1.$$

Since $\left(\boldsymbol{a}_i^\top \boldsymbol{x}\right)^2 \leq \left(\boldsymbol{a}_i^\top \left(\boldsymbol{A}^\top \boldsymbol{V}\boldsymbol{A}\right)^{-1} \boldsymbol{a}_i\right)\left(\boldsymbol{x}^\top \boldsymbol{A}^\top \boldsymbol{V}\boldsymbol{A}\boldsymbol{x}\right)$, this shows that $E \subset P$.

On the other hand, for any $\boldsymbol{x} \in P$, we have that

$$\begin{aligned}
\boldsymbol{x}^\top \boldsymbol{A}^\top \boldsymbol{V}\boldsymbol{A}\boldsymbol{x} &\leq \sum_{i=1}^m \boldsymbol{v}_i \|\boldsymbol{A}\boldsymbol{x}\|_\infty^2 \\
&\leq \sum_{i=1}^m (1+\varepsilon)\boldsymbol{\sigma}_i(\boldsymbol{V}^{1/2}\boldsymbol{A}) \\
&= (1+\varepsilon)n.
\end{aligned}$$

Therefore, we have that $P \subset \sqrt{(1+\varepsilon)n}E$. □

Remark 5.4.4. We call the ellipsoid found in this theorem is an approximate John ellipsoid because it almost satisfies the optimality condition of John ellipsoid, i.e. $\boldsymbol{v}_i = \boldsymbol{\sigma}_i(\boldsymbol{V}^{1/2}\boldsymbol{A})$. Although this does not show that the ellipsoid is actually close to the true John ellipsoid, this has similar rounding guarantee (5.3) as John ellipsoid and hence satisfies the requirement for most of the applications such as volume computation [262] and solving linear systems (next chapter).

Chapter 6

Faster Linear Programming

6.1 Introduction

In this chapter, we present a new algorithm for the following linear programs

$$\begin{array}{rcl} \min & & \boldsymbol{c}^T\boldsymbol{x}. \\ \boldsymbol{x} \in \mathbb{R}^m & : & \boldsymbol{A}^T\boldsymbol{x} = \boldsymbol{b} \\ \forall i \in [m] & : & l_i \leq x_i \leq u_i \end{array} \tag{6.1}$$

Our algorithm is based on interior point methods, a framework for solving linear programming. Roughly speaking, interior point methods approximate the domain by a sequence of ellipsoids and the number of iterations depends on how good are these approximations. In Chapter 5, we showed a family of ellipsoids that is easy to compute and gives a good approximations for arbitrary symmetric polytopes. Using these ellipsoids, we develop an interior point method that takes $\widetilde{O}(\sqrt{\text{rank}(\boldsymbol{A})}L)$[1] iterations (Theorem 6.8.1). Furthermore, we show how to achieve this convergence rate while only solving $\widetilde{O}(1)$ linear systems and performing additional $\widetilde{O}(\text{nnz}(\boldsymbol{A}))$ work in each iteration. This is the first polynomial improvement on the running time of solving linear programming over 25 years.

Applying our techniques to the linear programming formulation of the maximum flow problem we achieve a running time of $\widetilde{O}(|E|\sqrt{|V|}\log(U))$ for solving the problem on a graph with $|E|$ edges, $|V|$ vertices, and capacity ratio U (Theorem 6.9.3). This improves upon the previous fastest running time of $\widetilde{O}(|E|\min\{|E|^{1/2}, |V|^{2/3}\}\log(U))$ achieved over 15 years ago by Goldberg and Rao [104][2] and the previous fastest running times for solving dense directed unit capacity graphs of $O(|E|\min\{|E|^{1/2}, |V|^{2/3}\})$ achieved by Even and Tarjan [85] over 35 years ago and of $\widetilde{O}(|E|^{10/7})$ achieved recently by Mądry [183].

6.1.1 Organization

The rest of this chapter is structured as follows. In Section 6.3, we provide some background information on linear programming and interior point methods and show how to obtain the "first" $\widetilde{O}(\sqrt{\text{rank}(\boldsymbol{A})}L)$ iterations algorithm for the standard dual linear program (6.2). In Section 6.4, we explain why this algorithm is not sufficient for the maximum flow problem. In Section 6.5 we introduce our framework to use our alternative barrier function more efficiently and in Section 6.6 we present the key lemmas used to analyze progress along paths and in Section 6.7 we introduce the weight function we use to find paths. In Section 6.8 provide our linear programming algorithm for the linear program (6.1). In Section 6.9 we use these results to achieve our desired running times for the maximum flow

[1] Throughout this thesis L denotes the standard "bit complexity" of the linear program, a quantity less than the number of bits needed to represent (6.2). For integral $\boldsymbol{A}$, $\boldsymbol{b}$, and $\boldsymbol{c}$ this quantity is often defined as $L \stackrel{\text{def}}{=} \log(1 + d_{max}) + \log(1 + \max\{\|\boldsymbol{c}\|_\infty, \|\boldsymbol{b}\|_\infty\})$ where d_{max} is the largest absolute value of the determinant of a square sub-matrix of $\boldsymbol{A}$ [136].

[2] In this thesis we are primarily concerned with "weakly" polynomial time algorithms. The current fastest "strongly" polynomial running time for solving this problem is $O(nm)$ [218].

problem and its generalizations.

6.2 Previous Work

Linear programming and maximum flow are extremely well studied problems with long histories. Here we focus on recent history that particularly influenced our work.

6.2.1 Convergence Rate of Interior Point Methods

Given a matrix, $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, and vectors, $\boldsymbol{b} \in \mathbb{R}^m$ and $\boldsymbol{c} \in \mathbb{R}^n$, solving the linear program[3]

$$\min_{\boldsymbol{x} \in \mathbb{R}^n \,:\, \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b}} \boldsymbol{c}^T \boldsymbol{x} \tag{6.2}$$

is a core algorithmic task for both the theory and practice of computer science.

In 1984, Karmarkar [136] provided the first proof of an interior point method running in polynomial time. This method required $O(mL)$ iterations where the running time of each iteration was dominated by the time needed to solve a linear system of the form $\left(\boldsymbol{A}^T \boldsymbol{D} \boldsymbol{A}\right) \boldsymbol{x} = \boldsymbol{y}$ for some positive diagonal matrix $\boldsymbol{D} \in \mathbb{R}^{m \times m}$ and some $\boldsymbol{y} \in \mathbb{R}^n$. Using low rank matrix updates and preconditioning Karmarkar achieved a running time of $O(m^{3.5}L)$.

In 1988, Renegar improved the convergence rate of interior point methods to $O\left(\sqrt{m}L\right)$ iterations [227]. He presented a path following method where the iterations had comparable complexity to Karmarkar's method. Using a combination of techniques involving low rank updates, preconditioning and fast matrix multiplication, the amortized complexity of each iteration was improved [254, 107, 211] yielding the current state of the art running time of $\tilde{O}(m^{1.5}nL)$ [257].

In 1989, Vaidya [258] proposed two barrier functions which were shown to yield $O((m\text{rank}(\boldsymbol{A}))^{1/4}\, L)$ and $O(\text{rank}(\boldsymbol{A})L)$ iteration linear programming algorithms [255, 258, 253]. Unfortunately each iteration of these methods required explicit computation of the projection matrix $\boldsymbol{D}^{1/2}\boldsymbol{A}(\boldsymbol{A}^T\boldsymbol{D}\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{D}^{1/2}$ for a positive diagonal matrix $\boldsymbol{D} \in \mathbb{R}^{m \times m}$. This was improved by Anstreicher [15] who showed it sufficed to compute the diagonal of this projection matrix. Unfortunately both methods do not yield faster running times unless $m \gg n$.

In 1994 Nesterov and Nemirovski [211] showed that path-following methods can in principle be applied to any solve convex optimization problem, given a suitable barrier function. The number of iterations of their method depends on a square root of a quantity known as the *self-concordance* of the barrier. They showed that for any convex set in $\mathbb{R}^n$, there exists a barrier function, called the *universal barrier* function, with self-concordance $O(n)$ and therefore in principle any convex optimization problem with n variables can be solved in $O\left(\sqrt{n}L\right)$ iterations. However, this result is generally considered to be only of theoretical interest as the universal barrier function is defined as the volume of certain polytopes, a problem which in full-genearality is NP-hard and can only be computed approximately in $O(n^c)$ for some large constant c [180].

These results suggested that you can solve linear programs closer to the $\tilde{O}(\sqrt{\text{rank}(\boldsymbol{A})}L)$ bound achieved by the universal barrier only if you pay more in each iteration. In this chapter we show that this is not the case and up to polylogarithmic factors we achieve the convergence rate of the universal barrier function while only having iterations of cost comparable to that of Renegar's algorithm.

[3] This expression is the dual of a linear program written in standard form. It is well known that all linear programs can be written as (6.2). Note that this notation of m and n differs from that in some papers. Here m denotes the number of constraints and n denotes the number of variables. To avoid confusion we state many of our results in terms of $\sqrt{\text{rank}(\boldsymbol{A})}$ instead of $\sqrt{n}$.

Year	Author	Number of Iterations	Nature of iterations
1984	Karmarkar [136]	$O(mL)$	Linear system solve
1986	Renegar [227]	$O(\sqrt{m}L)$	Linear system solve
1989	Vaidya [253]	$O((m\text{rank}(\boldsymbol{A}))^{1/4}L)$	Expensive linear algebra
1994	Nesterov and Nemirovskii [211]	$O(\sqrt{\text{rank}(\boldsymbol{A})}L)$	Volume computation
-	This chapter	$\tilde{O}(\sqrt{\text{rank}(\boldsymbol{A})}L)$	$\widetilde{O}(1)$ Linear system solves

Figure 6-1: Interior Point Iteration Improvements

6.2.2 Recent Breakthroughs on the Maximum Flow Problem

A beautiful line of work on solving the maximum flow on undirected graphs began with a result of Benzcur and Karger in which they showed how to reduce approximately computing minimum s-t cuts on arbitrary undirected graphs to the same problem on sparse graphs, i.e. those with $|E| = \widetilde{O}(|V|)$ [33]. Pushing these ideas further Karger showed how to perform a similar reduction for the maximum flow problem [134] and Karger and Levine showed how to compute the exact maximum flow in an unweighted undirected graph with maximum flow value F in $\widetilde{O}(|E| + |V|F)$ time [135].

In a breakthrough result of Spielman and Teng [243] they showed that a particular class of linear systems, Laplacians, can be solved in nearly linear time. Leveraging these fast Laplacian system solvers Christiano, Kelner, Mądry, and Spielman [51] showed how to compute $(1-\varepsilon)$ approximations to the maximum flow problem on undirected graphs in $\widetilde{O}(|E| \cdot |V|^{1/3}\varepsilon^{-11/3})$. Later Lee, Rao and Srivastava [161] improved the running time to $\widetilde{O}(|E| \cdot |V|^{1/3}\varepsilon^{-2/3})$ for unweighted undirected graphs. This line of work culminated in recent results of Sherman [233] and Kelner, Lee, Orecchia, and Sidford (Chapter 12) who showed how to solve the problem in $O(|E|^{1+o(1)}\varepsilon^{-2})$, using congestion-approximators, oblivious routings, and techniques developed by Mądry [182].

In 2008, Daitch and Spielman [63] showed that, by careful application of interior point techniques, fast Laplacian system solvers [243], and a novel method for solving M-matrices, they could match (up to polylogarithmic factors) the running time of Goldberg Rao and achieve a running time of $\widetilde{O}(|E|^{3/2}\log^2(U))$ not just for maximum flow but also for the minimum cost flow and lossy generalized minimum cost flow problems (see Fig 6-2).

Very recently Mądry [183] achieved an astounding running time of $\widetilde{O}(|E|^{10/7})$ for solving the maximum flow problem on un-capacitated directed graphs by a novel application and modification of interior point methods. This shattered numerous barriers providing the first general improvement over the running time of $O(|E|\min\{|E|^{1/2}, |V|^{2/3}\})$ for solving unit capacity graphs proven over 35 years ago by Even and Tarjan [85] in 1975.

While our algorithm for solving the maximum flow problem is new, we make extensive use of these breakthroughs. We use sampling techniques first discovered in the context of graph sparsification (previous two chapters) to re-weight the graph so that we make progress at a rate commensurate with the number of vertices and not the number of edges. We use fast Laplacian system solvers as in [51, 161] to make the cost of interior point iterations cheap when applied to the linear program formulations analyzed by Daitch and Spielman [63]. Furthermore, as in Mądry [183] we use weights to change the central path (albeit for a slightly different purpose). We believe this further emphasizes the power of these tools as general purpose techniques for algorithm design.

Year	Author	Running Time
1972	Edmonds, Karp [84]	$\tilde{O}(\|E\|^2 \log(U))$
1984	Tardos [247]	$O(\|E\|^4)$
1984	Orlin [217]	$\tilde{O}(\|E\|^3)$
1986	Galil, Tardos [98]	$\tilde{O}\left(\|E\|\|V\|^2\right)$
1987	Goldberg, Tarjan [106]	$\tilde{O}(\|E\|\|V\| \log(U))$
1988	Orlin [215]	$\tilde{O}(\|E\|^2)$
2008	Daitch, Spielman [63]	$\tilde{O}(\|E\|^{3/2} \log^2(U))$
-	This chapter	$\tilde{O}(\|E\|\sqrt{\|V\|} \log^2(U))$

Figure 6-2: Minimum Cost Flow Running Times Improvements

6.3 Simple Case

In this section, we introduce the notation of ν-self-concordant barrier functions and explain how such barrier functions can be used to develop an $O(\sqrt{\nu}L)$ iterations linear programming algorithm. Then, we construct an $\tilde{O}(n)$-self-concordant barrier function which can be computed in polynomial time. This barrier function will only be used to give the intuition on how we are getting our main algorithm. The reader well versed in path following methods can likely skip to subsection 6.3.5 and to the more curious reader we encourage them to consider some of the many wonderful expositions on this subject [209, 273, 107] for further reading.

6.3.1 The Setup

Given a matrix, $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, and vectors, $\boldsymbol{b} \in \mathbb{R}^m$ and $\boldsymbol{c} \in \mathbb{R}^n$, the goal of this section is to solve the following linear program

$$\min_{\boldsymbol{x} \in \mathbb{R}^n \,:\, \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b}} \boldsymbol{c}^T \boldsymbol{x} \tag{6.3}$$

This is called the dual of the *standard form* of a linear program and all linear programs can be expressed in this form. We call a vector $\boldsymbol{x} \in \mathbb{R}^m$ *feasible* if $\boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b}$, we call $\boldsymbol{c}^T\boldsymbol{x}$ the *cost* of such a vector. therefore our goal is to either compute a minimum cost feasible vector or determine that none exists.

We assume that $\boldsymbol{A}$ is full rank, i.e. $\text{rank}(\boldsymbol{A}) = n$, and that $m \geq n$. Nevertheless, we still write many of our results using $\text{rank}(\boldsymbol{A})$ rather than n for two reasons. First, this notation makes clear that $\text{rank}(\boldsymbol{A})$ is referring to the smaller of the two quantities m and n. Second, if $\text{rank}(\boldsymbol{A}) < n$, then we can reduce the number of variables to $\text{rank}(\boldsymbol{A})$ by a change of basis. Hence, we only need to solve linear programs in the full rank version.

6.3.2 Central Path

Interior point methods solve (6.3) by maintaining a point $\boldsymbol{x}$ that is in the *interior* of the feasible region, i.e. $\boldsymbol{x} \in \Omega$ where

$$\Omega \stackrel{\text{def}}{=} \{\boldsymbol{x} \in \mathbb{R}^n \;:\; \boldsymbol{A}\boldsymbol{x} > \boldsymbol{b}\}.$$

These methods attempt to iteratively decrease the cost of $\boldsymbol{x}$ while maintaining strict feasibility. This is often done by considering some measurement of the distance to feasibility such as $\boldsymbol{s} \stackrel{\text{def}}{=} \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}$, called the *slacks*, and creating some penalty for these distances approaching 0. By carefully balancing penalties for small $\boldsymbol{s}(\boldsymbol{x})$ and penalties for large $\boldsymbol{c}^T\boldsymbol{x}$, these methods eventually compute a point close

enough to the optimum solution that it can be computed exactly.

Path following methods fix ratios between the the penalty for large $\boldsymbol{c}^T\boldsymbol{x}$ and the penalty for small $\boldsymbol{s}$ and alternate between steps of optimizing with respect to this ratio and changing the ratio. These methods typically encode the penalties through a *barrier function* $\phi : \Omega \to \mathbb{R}$ such that $\phi(\boldsymbol{x}) \to \infty$ as $\boldsymbol{s}(\boldsymbol{x})_i \to 0$ for any $i \in [m]$ and they encode the ratio through some parameter $t > 0$. Formally, they attempt to solve optimization problems of the following form for increasing values of t

$$\min_{\boldsymbol{x}\in\mathbb{R}^m} f_t(\boldsymbol{x}) \quad \text{where} \quad f_t(\boldsymbol{x}) \stackrel{\text{def}}{=} t \cdot \boldsymbol{c}^T\boldsymbol{x} + \phi(\boldsymbol{x}) \tag{6.4}$$

Since $\phi(\boldsymbol{x}) \to \infty$ as $\boldsymbol{s}(\boldsymbol{x})_i \to 0$, the minimizer of $f_t(\boldsymbol{x})$, denoted $\boldsymbol{x}^*(t)$, is in Ω for all t. As t increases the effect of the cost vector on $\boldsymbol{x}^*(t)$ increases and the distance from the boundary of the feasible region as measured by $\boldsymbol{s}(\boldsymbol{x})$ decreases. One can think of the points $\{\boldsymbol{x}^*(t) \mid t > 0\}$ as a path in $\mathbb{R}^n$, called *the central path*, where $\boldsymbol{x}^*(t)$ approaches a solution to (6.3) as $t \to \infty$. A standard choice of barrier is the *standard log barrier*, $\phi(\boldsymbol{x}) \stackrel{\text{def}}{=} -\sum_{i=1}^m \log(\boldsymbol{s}(\boldsymbol{x})_i)$ and for this choice of barrier we refer to $\{\boldsymbol{x}^*(t) \mid t > 0\}$ as the *standard central path*.

Path following methods typically follow the following framework:

(1) *Compute Initial Point:* Compute an approximation $\boldsymbol{x}^*(t)$ for some t.

(2) *Follow the central path*: Repeatedly increase t and compute an approximation to $\boldsymbol{x}^*(t)$.

(3) *Round to optimal solution*: Use the approximation to $\boldsymbol{x}^*(t)$ to compute the solution to (6.3).

Usually, steps (1) and (3) are not the bottleneck of interior point methods (including our algorithm) and can be carried out by standard interior point techniques. In this chapter, we ignore the details for step (1) and (3) and refer the interested reader to the paper version of this chapter [165, 166]. In the following subsection, we provide a simple technique for performing (2) and serves as the foundation for the algorithms considered in the remainder of this section.

6.3.3 Following the Path

Typically, path following methods compute an approximation to $\boldsymbol{x}^*(t)$ using Newton's method. While for an arbitrary current point $\boldsymbol{x} \in \Omega$ and $t > 0$, the function $f_t(\boldsymbol{x})$ can be ill-behaved, in a region near $\boldsymbol{x}^*(t)$, the Hessian $\nabla^2 f_t(\boldsymbol{x})$ changes fairly slowly. More precisely, if one considers the second order approximation of $f_t(\boldsymbol{z})$ around some point $\boldsymbol{x} \in \Omega$ "close enough" to $\boldsymbol{x}^*(t)$

$$f_t(\boldsymbol{z}) \approx f_t(\boldsymbol{x}) + \langle \nabla f_t(\boldsymbol{x}), \boldsymbol{z} - \boldsymbol{x}\rangle + \frac{1}{2}(\boldsymbol{z} - \boldsymbol{x})^T (\nabla^2 f_t(\boldsymbol{x}))(\boldsymbol{z} - \boldsymbol{x}) ,$$

and applies one step of Newton's method, i.e. minimizes this quadratic approximation to compute

$$\boldsymbol{x}^{(new)} := \boldsymbol{x} - (\nabla^2 f_t(\boldsymbol{x}))^{-1}\nabla f_t(\boldsymbol{x})$$

for $\boldsymbol{s} \stackrel{\text{def}}{=} \boldsymbol{s}(\boldsymbol{x})$, then this procedure rapidly converges to $\boldsymbol{x}^*(t)$.

To quantify this, we measure *centrality*, i.e. how close the current point $\boldsymbol{x} \in \Omega$ is to $\boldsymbol{x}^*(t)$, by the size of this *Newton step* in the Hessian induced norm. For $\boldsymbol{x} \in \Omega$ and Newton step $\boldsymbol{h}_t(\boldsymbol{x}) \stackrel{\text{def}}{=} (\nabla^2 f_t(\boldsymbol{x}))^{-1}\nabla f_t(\boldsymbol{x})$, we denote centrality by $\delta_t(\boldsymbol{x}) \stackrel{\text{def}}{=} \|\boldsymbol{h}_t(\boldsymbol{x})\|_{\nabla^2 f_t(\boldsymbol{x})}$.

For the standard log barrier function, $\phi(\boldsymbol{x}) \stackrel{\text{def}}{=} -\sum_{i\in[m]} \log(\boldsymbol{s}(\boldsymbol{x})_i)$, a standard analysis of Newton's method shows that if $\delta_t(\boldsymbol{x}) \le \frac{1}{2}$, then for $\boldsymbol{x}^{(new)} := \boldsymbol{x} - \boldsymbol{h}(\boldsymbol{x})$, we have $\delta_t(\boldsymbol{x}^{(new)}) \le 2\delta_t(\boldsymbol{x})^2$. Furthermore, if $\delta_t(\boldsymbol{x}) \le \frac{1}{4}$, it is not hard to show that for $t' = t(1+m^{-1/2}/4)$ we have $\delta_{t'}(\boldsymbol{x}) \le 1/2$. Combining these facts yields that in $O(\sqrt{m})$ iterations we can double t while maintaining a *nearly centered* $\boldsymbol{x}$,

i.e. $\delta_t(\boldsymbol{x})$ at most a constant. With some additional work, it can be shown that by maintaining a nearly centered $\boldsymbol{x}$ and increasing t at most $O(\sqrt{m}L)$ times one can compute a solution to (6.3) exactly. Therefore, this barrier function gives an $O(\sqrt{m}L)$ iterations algorithm for solving (6.3).

6.3.4 Self-Concordant Barrier Functions

In a seminal result of Nesterov and Nemirovski [211], they studied the sufficient conditions (called *self-concordance*) on ϕ that allows us to follow the path efficiently. They showed that given the ability to construct a *ν-self-concordant barrier* for a convex set, one can minimize linear functions over that convex set with a convergence rate of $O(\sqrt{\nu})$.

Definition 6.3.1. Given a open convex set $K \subset \mathbb{R}^n$. We call ϕ is a ν-self-concordant barrier function for K if the following conditions are satisfies

- ϕ is convex and thrice continuously differentiable,
- $\phi(\boldsymbol{x}_i) \to \infty$ for any sequence $\boldsymbol{x}_i \in K$ converging to boundary of K.
- $|D^3\phi(\boldsymbol{x})[\boldsymbol{h},\boldsymbol{h},\boldsymbol{h}]| \leq 2|D^2\phi(\boldsymbol{x})[\boldsymbol{h},\boldsymbol{h}]|^{3/2}$ for any $\boldsymbol{x} \in K$, $\boldsymbol{h} \in \mathbb{R}^n$,
- $|DF(\boldsymbol{x})[\boldsymbol{h}]| \leq \sqrt{\nu}|D^2\phi(\boldsymbol{x})[\boldsymbol{h},\boldsymbol{h}]|^{1/2}$ for any $\boldsymbol{x} \in K$, $\boldsymbol{h} \in \mathbb{R}^n$.

For example, the standard log barrier function is a m-self concordant barrier function for polytopes. Nesterov and Nemirovski further showed that, for any convex set in $\mathbb{R}^n$, there exists a barrier function, called the *universal barrier* function, with self-concordance $O(n)$ [211]. However, the universal barrier function is defined via the volume of certain polytopes, which is very difficult to compute.

6.3.5 A $\tilde{O}(n)$-Self-Concordant Barrier Function

In this section, we showed that for any polytope, there is an easy-to-compute barrier function with self-concordance $\tilde{O}(n)$. This barrier function is based on the following intuition on interior point methods. In high level, interior point methods replace the key difficulty of linear programming, the hard constraints, by barrier functions and solve the linear programs by a sequence of linear systems induced by barrier functions. Since these linear system corresponds to minimizing linear functions over ellipsoids and these ellipsoids comes from the second order approximation of barrier functions, interior point methods, to a certain degree, use a sequence of ellipsoids to approximate polytopes. Self-concordance can be thought as a geometric condition that relates how good those ellipsoids approximate the domain. In particular, the following lemma shows that the second order approximation of the barrier function at the minimum point indeed has a geometric implication, it approximates the domain.

Theorem 6.3.2 ([206, Thm 4.2.6]). *Given a ν-self-concordant barrier function ϕ for some convex set $\Omega \subset \mathbb{R}^n$. Let x_ϕ be the minimizer of ϕ and let the Dikin ellipse $E = \{x \in \mathbb{R}^n : (x - x_\phi)^T\nabla^2\phi(x_\phi)(x - x_\phi) \leq 1\}$. Then, we have that*

$$E \subset \Omega \subset (\nu + 2\sqrt{\nu})E. \tag{6.5}$$

This shows that in order to construct a $\tilde{O}(n)$-self-concordant barrier function, we must able to find an ellipse that gives a $\tilde{O}(n)$ approximation of the convex set Ω. There are quite a few ellipsoids satisfying the condition (6.5). In contrast to other known ellipsoids that gives an approximation guarantee such as second moment, John ellipsoid is defined by a convex optimization problem and therefore can be computed in weakly polynomial time.

Lemma 6.3.3 ([128]). *Given a convex $\Omega \subset \mathbb{R}^n$. We define the John ellipsoid $J(\Omega)$ as the largest volume ellipsoid contained inside Ω. Then, we have that*

$$J(\Omega) \subset K \subset nJ(\Omega).$$

There are different ways to write down the John ellipsoid as a convex problem. Our barrier function is motivated by the following formulation, called *D-optimal design.*

Lemma 6.3.4 ([143]). *For any polytope $\Omega = \{\boldsymbol{A}\boldsymbol{x} > \boldsymbol{b}\}$, the John ellipsoid $J(K)$ is given by $\{\boldsymbol{y} \in \mathbb{R}^n : (\boldsymbol{y}-\boldsymbol{x})^T \boldsymbol{A}^T \boldsymbol{W} \boldsymbol{A}(\boldsymbol{y}-\boldsymbol{x}) \leq 1\}$ where $(\boldsymbol{x}, \boldsymbol{w})$ is the saddle point of the following convex concave problem*

$$\min_{\boldsymbol{x}} \max_{\sum w_i = n, w_i \geq 0} \ln\det\left(\boldsymbol{A}^T \boldsymbol{S}^{-1} \boldsymbol{W} \boldsymbol{S}^{-1} \boldsymbol{A}\right) \tag{6.6}$$

where $\boldsymbol{S}$ and $\boldsymbol{W}$ are diagonal $m \times m$ matrices with $\boldsymbol{S}_{ii} = a_i^T \boldsymbol{x} - b_i$ and $\boldsymbol{W}_{ii} = w_i$.

Motivated by Theorem 6.3.2, we want to find a barrier function whose minimizer is exactly the center of John ellipsoid. From (6.6), we see that the function

$$\phi_\infty(\boldsymbol{x}) \stackrel{\text{def}}{=} \max_{\sum w_i = n, w_i \geq 0} \ln\det\left(\boldsymbol{A}^T \boldsymbol{S}^{-1} \boldsymbol{W} \boldsymbol{S}^{-1} \boldsymbol{A}\right)$$

satisfies this requirement exactly.

However, $\phi_\infty(\boldsymbol{x})$ is not even continuously differentiable. To see this, we define $J(\Omega, x)$ be the maximum volume ellipsoid inside K and centered at x. We note that $\phi_\infty(\boldsymbol{x}) = c\log(\mathrm{vol}(J(\Omega, x)))$ for some universal constant c. Consider the convex set $\Omega = [0, 1]$. For any $x < 1/2$, $J(\Omega, x)$ is the "ellipsoid" $[0, 2x]$ and in some sense this ellipsoid does not see the constraint $x \leq 1$. Therefore, the function $\phi_\infty(x)$ is not continuously differentiable at $1/2$.

To make $\phi_\infty(\boldsymbol{x})$ smooth, we employ the standard approach, adding a strongly concave term into the objective function $\ln\det\left(\boldsymbol{A}^T \boldsymbol{S}^{-1} \boldsymbol{W} \boldsymbol{S}^{-1} \boldsymbol{A}\right)$. In general, if a smooth function $f(x, y)$ is strongly concave in y, then $\max_y f(x, y)$ is smooth. Since the domain of w is a simplex, a natural choice for the strongly concave term is the entropy function. Therefore, we consider the following function

$$\max_{\sum w_i = n, w_i \geq 0} \ln\det\left(\boldsymbol{A}^T \boldsymbol{S}^{-1} \boldsymbol{W} \boldsymbol{S}^{-1} \boldsymbol{A}\right) + \rho(\boldsymbol{w})$$

with $\rho(\boldsymbol{w}) = -\sum_{i=1}^m w_i \ln w_i$. By some direct calculations, one can show this is indeed a $\tilde{O}(n)$ self-concordant barrier function. In fact, there are many other choices of ρ that give similar result. For example, we considered $\rho(\boldsymbol{w}) = -\sum_{i=1}^m w_i^{1+1/\log(m/n)}$ in our original paper. In the appendix, we proved the following:

Theorem 6.3.5. *For any polytope $\Omega = \{\boldsymbol{A}x > b\}$, we define the barrier $p(x)$ for Ω as $\max_{w_i \geq 0} f(x, w)$ where*

$$f(x, w) = \ln\det(\boldsymbol{A}^T \boldsymbol{S}^{-1} \boldsymbol{W} \boldsymbol{S}^{-1} \boldsymbol{A}) - \frac{n}{m}\sum_{i=1}^m w_i \ln w_i - \frac{n}{m}\sum_{i=1}^m \ln s_i \tag{6.7}$$

and $\boldsymbol{S}$ and $\boldsymbol{W}$ are diagonal $m \times m$ matrices with $\boldsymbol{S}_{ii} = a_i^T \boldsymbol{x} - b_i$ and $\boldsymbol{W}_{ii} = w_i$. Then, p is an $O\left(n \ln^6\left(\frac{me}{n}\right)\right)$-self concordant barrier function (after rescaling). More precisely, we have

1. *For any $x \in \Omega^\circ$, $(\nabla p(x))^T \left(\nabla^2 p(x)\right)^{-1} (\nabla p(x)) \leq O(n)$.*

2. *For any $x \in \Omega^\circ$, $h \in \mathbb{R}^n$, we have $\left|\nabla^3 p(x)[h, h, h]\right| \leq O\left(\ln^3\left(\frac{me}{n}\right)\right) \left|\nabla^2 p(x)[h, h]\right|^{3/2}$.*

We remark that both the scaling n/m and the extra term $\sum_{i=1}^{m} \ln s_i$ is not essential. In fact, one consider the ℓ_p Lewis ellipsoids instead and use the following barrier function

$$\phi_p(x) \stackrel{\text{def}}{=} \min_{\boldsymbol{M}} -\log\det \boldsymbol{M} \text{ subject to } \sum_{i=1}^{m} \left(\frac{\boldsymbol{a}_i^\top \boldsymbol{M} \boldsymbol{a}_i}{\boldsymbol{s}_i^2(x)}\right)^{p/2} \leq n \text{ and } \boldsymbol{M} \succeq \boldsymbol{0}.$$

One can show that $\phi_{\log(m)}$ is an $\widetilde{O}(n)$-self concordant barrier function. However, this barrier function is more difficult to analyze and hence we left it as an 15 pages calculus exercise.

6.4 Difficulties

In this section, we discuss two issues arises from using the barrier function defined in the previous section and outline our approaches used in the rest of this chapter for fixing them.

6.4.1 Avoiding Matrix Multiplication

The minimax formula involves the maximizer $\boldsymbol{w}$ in (6.7), which we call weight function $g(\boldsymbol{x})$. Computing $g(\boldsymbol{x})$ requires computing the diagonal of some projection matrices as in Vaidya and Anstreicher's work [257, 15] and is therefore not efficient enough for our purposes. Although we can compute $g(\boldsymbol{x})$ approximately up to certain multiplicative coordinate-wise error using dimension reduction techniques (Section 6.7.1), this error is still too much for path following to handle the directly as multiplicatively changing weights can hurt our measures of progress too much.

To avoid the minimax formulation, we note that the barrier function $p(\boldsymbol{x})$ has similar behaviors as $-\sum \boldsymbol{w}_i \ln \boldsymbol{s}_i(\boldsymbol{x})$ around $\boldsymbol{x}$ if $\boldsymbol{w}$ is close to the maximizer in (6.7) for that given $\boldsymbol{x}$. Therefore, rather than using the minimax formula, we maintain separate weights $\boldsymbol{w}$ and current point $\boldsymbol{x}$ and use the barrier

$$\phi(\boldsymbol{x}, \boldsymbol{w}) = -\sum_{i \in [m]} \boldsymbol{w}_i \ln \boldsymbol{s}_i(\boldsymbol{x}).$$

We then maintain two invariants, (1) $\boldsymbol{x}$ is centered, i.e. $\boldsymbol{x}$ close to the minimum point of $t \cdot \boldsymbol{c}^T \boldsymbol{x} + \phi(\boldsymbol{x}, \boldsymbol{w})$ and (2) $\boldsymbol{w}$ close to $\boldsymbol{g}(\boldsymbol{x})$ multiplicatively.

We separate the problem of maintaining these invariants into two steps. First, we show that a Newton step for $\boldsymbol{x}$ improves centrality without moving $\boldsymbol{w}$ too far away from $\boldsymbol{g}(\boldsymbol{x})$. Second, we show that given a multiplicative approximation to $\boldsymbol{g}(\boldsymbol{x})$ and bounds for how much $\boldsymbol{g}(\boldsymbol{x})$ may have changed, we can maintain the invariant that $\boldsymbol{g}(\boldsymbol{x})$ is close to $\boldsymbol{w}$ multiplicatively without moving $\boldsymbol{w}$ too much. We formulate this as a general two player game and prove that there is an efficient strategy to maintain our desired invariant. Combining these and standard techniques in path-following methods, we obtain an $\tilde{O}(\sqrt{\text{rank}(\boldsymbol{A})}L)$ iterations path-following algorithm for solving (6.2) where each iterations consists of $\tilde{O}(1)$ linear system solves.

6.4.2 Generalizing The Result

Unfortunately, this result is insufficient to produce faster algorithms for the maximum flow problem and its generalizations. Given an arbitrary minimum cost maximum flow instance there is a natural linear program that one can use to express the problem:

$$\begin{array}{cc} \min & \boldsymbol{c}^T \boldsymbol{x} \\ \boldsymbol{x} \in \mathbb{R}^m & : \ \boldsymbol{A}^T \boldsymbol{x} = \boldsymbol{b} \\ \forall i \in [m] & : \ l_i \leq x_i \leq u_i \end{array} \tag{6.8}$$

where the variables x_i denote the flow on an edge, the l_i and u_i denote lower and upper bounds on how much flow we can put on the edge, and $\boldsymbol{A}$ is the incidence matrix associated with the graph [63]. In this formulation, rank($\boldsymbol{A}$) is less than the number of vertices in the graph and using fast Laplacian system solvers [243, 147, 146, 141, 162, 170, 223] we can solve linear systems involving $\boldsymbol{A}$ in time nearly linear in the number of edges in the graph. Thus, if we could solve (6.8) in time comparable to that we achieve for solving (6.2) this would immediately yield a $\widetilde{O}(m\sqrt{n}\log^{O(1)}(U))$ algorithm for the maximum flow problem. Unfortunately, it is not clear how to apply our previous results in this more general setting and naive attempts to write (6.8) in the form of (6.2) without increasing rank($\boldsymbol{A}$) fail.

Even more troubling, achieving a faster than $\widetilde{O}(\sqrt{m}L)$ iterations interior point method for solving general linear programs in this form would break a long-standing barrier for the convergence rate of interior point methods. To the best of the authors knowledge, there is no general purpose interior point method that achieves a convergence rate faster than the self concordance of the best barrier of the feasible region. Furthermore, using lower bounds results of Nesterov and Nemirovski [211, Proposition 2.3.6], it is not hard to see that any general barrier for (6.8) must have self-concordance $\Omega(m)$.

After all, linear programs of the form (6.2) are easier because each step of Newton steps lies in some ellipsoids that is strictly contained inside the domain $\boldsymbol{Ax} \geq \boldsymbol{b}$. However, such restriction implies that it takes at least $O(\sqrt{m})$ iterations to go from one point in the box $\{l_i \leq x_i \leq u_i\}$ to another point and that is too slow. Therefore, we need to explicitly bound the size of the Newton step in both the ℓ_∞ norm and a weighted ℓ_2 norm. Hence, we measure the progress by the size of the Newton step in a mixed norm of the form $\|\cdot\| = \|\cdot\|_\infty + C_{\text{norm}}\|\cdot\|_{\boldsymbol{W}}$ to keep track of these two quantities simultaneously.

Measuring of Newton step size both with respect to the mixed norm helps to explain how our method outperforms the self-concordance of the best barrier for the space. Self-concordance is based on ℓ_2 analysis and the lower bounds for self-concordance are precisely the failure of the sphere to approximate a box. While ideally we would just perform optimization over the ℓ_∞ box directly, ℓ_∞ is ripe with degeneracies that makes this analysis difficult. Nevertheless, unconstrained minimization over a box is quite simple and by working with this mixed norm and choosing weights to improve the conditioning, we are taking advantage of the simplicity of minimizing ℓ_∞ over most of the domain and only paying for the n-self-concordance of a barrier for the smaller subspace induce by the requirement that $\boldsymbol{A}^T\boldsymbol{x} = \boldsymbol{b}$.

6.5 Weighted Path Finding

In this section, we formally define the formulation of linear program we solve (Subsection 6.5.1), introduce the weighted central path we follow (Subsection 6.5.3), define key properties of the path (Subsection 6.5.4) and the weights (Subsection 6.5.5) that we will use to produce an efficient path finding scheme.

6.5.1 The Problem

As we explained in subsection 6.4.2, the standard dual linear program is not suitable for the maximum flow application. From now on, our goal is to efficiently solve the following linear program

$$\begin{aligned} \min_{\boldsymbol{x} \in \mathbb{R}^m \ :\ \boldsymbol{A}^T\boldsymbol{x} = \boldsymbol{b}} \quad & \boldsymbol{c}^T\boldsymbol{x} \\ \forall i \in [m] \ :\ & l_i \leq x_i \leq u_i \end{aligned} \tag{6.9}$$

where $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, $\boldsymbol{b} \in \mathbb{R}^n$, $\boldsymbol{c} \in \mathbb{R}^m$, $l_i \in \mathbb{R} \cup \{-\infty\}$, and $u_i \in \mathbb{R} \cup \{+\infty\}$.[4] We assume that for all $i \in [m]$ the domain of variable x_i, $\text{dom}(x_i) \stackrel{\text{def}}{=} \{x : l_i \leq x \leq u_i\}$, is non-degenerate. In particular we assume that $\text{dom}(x_i)$ is not the empty set, a singleton, or the entire real line, i.e. $l_i < u_i$ and either $l_i \neq -\infty$ or $u_i \neq +\infty$. Furthermore we make the standard assumptions that $\boldsymbol{A}$ has full column rank, and therefore $m \geq n$, and we assume that the interior of the polytope, $\Omega \stackrel{\text{def}}{=} \{\boldsymbol{x} \in \mathbb{R}^m : \boldsymbol{A}^T\boldsymbol{x} = \boldsymbol{b}, l_i < x_i < u_i\}$, is non-empty.

The linear program (6.9) is a generalization of standard form, the case where for all $i \in [m]$ we have $l_i = 0$ and $u_i = +\infty$. While it is well known that all linear programs can be written in standard form, the transformations to rewrite (6.9) in standard form may increase the rank of $\boldsymbol{A}$ and therefore we solve (6.9) directly.

6.5.2 Coordinate Barrier Functions

Rather than working directly with the different domain of the x_i we take a slightly more general approach and for the remainder of the chapter assume that for all $i \in [m]$ we have a *barrier function*, $\phi_i : \text{dom}(x_i) \to \mathbb{R}$, such that

$$\lim_{x \to l_i} \phi_i(x) = \lim_{x \to u_i} \phi_i(x) = +\infty.$$

More precisely, we assume that each ϕ_i is a *1-self-concordant barrier function.* Recall from definition 6.3.1, ϕ is a *1-self-concordant barrier function* if

$$\left|\phi'''(x)\right| \leq 2(\phi''(x))^{3/2} \text{ for all } x \in \text{dom}(\phi) \tag{6.10}$$

and

$$\left|\phi'(x)\right| \leq \sqrt{\phi''(x)} \text{ for all } x \in \text{dom}(\phi). \tag{6.11}$$

The first condition (6.10) bounds how quickly the second order approximation to the function can change and the second condition (6.11) bounds how much force the barrier can exert.

The existence of a self-concordant barrier for the domain is a standard assumption for interior point methods [211]. However, for completeness, here we show how for each possible setting of the l_i and u_i there is an explicit 1-self-concordant barrier function we can use:

- *Case (1): l_i finite and $u_i = \infty$*: Here we use a *log barrier* defined as $\phi_i(x) \stackrel{\text{def}}{=} -\log(x - l_i)$. For this barrier we have

  $$\phi_i'(x) = -\frac{1}{x - l_i} \quad , \quad \phi_i''(x) = \frac{1}{(x - l_i)^2} \quad \text{, and} \quad \phi_i'''(x) = -\frac{2}{(x - l_i)^3}$$

  and therefore clearly $|\phi_i'''(x)| = 2(\phi_i''(x))^{3/2}$, $|\phi_i'(x)| = \sqrt{\phi_i''(x)}$, and $\lim_{x \to l_i} \phi_i(x) = \infty$.

- *Case (2): $l_i = -\infty$ and u_i finite*: Here we use a *log barrier* defined as $\phi_i(x) \stackrel{\text{def}}{=} -\log(u_i - x)$. For this barrier we have

  $$\phi_i'(x) = \frac{1}{u_i - x} \quad , \quad \phi_i''(x) = \frac{1}{(u_i - x)^2} \quad \text{, and} \quad \phi_i'''(x) = -\frac{2}{(u_i - x)^3}$$

  and therefore clearly $|\phi_i'''(x)| = 2(\phi_i''(x))^{3/2}$, $|\phi_i'(x)| = \sqrt{\phi_i''(x)}$, and $\lim_{x \to u_i} \phi_i(x) = \infty$.

[4] Typically (6.9) is written as $\boldsymbol{Ax} = \boldsymbol{b}$ rather than $\boldsymbol{A}^T\boldsymbol{x} = \boldsymbol{b}$. We chose this formulation to be consistent with the previous section and to be consistent with the standard use of n to denote the number of vertices and m to denote the number of edges in a graph in the linear program formulation of flow problems.

- *Case (3): l_i finite and u_i finite*: Here we use a *trigonometric barrier*[5] defined as $\phi_i(x) \stackrel{\text{def}}{=} -\log\cos(a_i x + b_i)$ for $a_i = \frac{\pi}{u_i - l_i}$ and $b_i = -\frac{\pi}{2}\frac{u_i + l_i}{u_i - l_i}$. Note for this choice as $x \to u_i$ we have $a_i x + b_i \to \frac{\pi}{2}$ and as $x \to l_i$ we have $a_i x + b_i \to \frac{-\pi}{2}$ and in both cases $\phi_i(x) \to \infty$. Furthermore,

$$\phi_i'(x) = a_i \tan(a_i x + b_i) \quad , \quad \phi_i''(x) = \frac{a_i^2}{\cos^2(a_i x + b_i)} \quad , \text{ and } \quad \phi_i''' = \frac{2a_i^3 \sin(a_i x + b_i)}{\cos^3(a_i x + b_i)}.$$

  Therefore, we have $|\phi_i'''(x)| \leq 2(\phi''(x))^{3/2}$ and $|\phi_i'(x)| \leq \sqrt{\phi_i''(x)}$.

For the remainder of this chapter we will simply assume that we have a 1 self-concordant barrier ϕ_i for each of the $\text{dom}(\phi_i)$ and will not use any more structure about the barriers.

While there is much theory regarding properties of self-concordant barrier functions we will primarily use two common properties about self-concordant barriers functions. The first property, Lemma 6.5.1, shows that the Hessian of the barrier cannot change to quickly, and the second property, Lemma 6.5.2 we use to reason about how the force exerted by the barrier changes over the domain.

Lemma 6.5.1 ([206, Theorem 4.1.6]). *Suppose ϕ is a 1-self-concordant barrier function. For all $s \in \text{dom}(\phi)$ if $r \stackrel{\text{def}}{=} \sqrt{\phi''(s)}\,|s - t| < 1$ then $t \in \text{dom}(\phi)$ and*

$$(1 - r)\sqrt{\phi''(s)} \leq \sqrt{\phi''(t)} \leq \frac{\sqrt{\phi''(s)}}{1 - r}.$$

Lemma 6.5.2 ([206, Theorem 4.2.4]). *Suppose ϕ is a 1-self-concordant barrier function. For all $x, y \in \text{dom}(\phi)$, we have*

$$\phi'(x) \cdot (y - x) \leq 1.$$

6.5.3 The Weighted Central Path

Our linear programming algorithm maintains a feasible point $\boldsymbol{x} \in \Omega$, weights $\boldsymbol{w} \in \mathbb{R}^m_{>0}$, and minimizes the following *penalized objective function*

$$\min_{\boldsymbol{A}^T \boldsymbol{x} = \boldsymbol{b}} f_t(\boldsymbol{x}, \boldsymbol{w}) \quad \text{where} \quad f_t(\boldsymbol{x}, \boldsymbol{w}) \stackrel{\text{def}}{=} t \cdot \boldsymbol{c}^T \boldsymbol{x} + \sum_{i \in [m]} w_i \phi_i(\boldsymbol{x}_i) \tag{6.12}$$

for increasing t and small $\boldsymbol{w}$. For every fixed set of *weights*, $\boldsymbol{w} \in \mathbb{R}^m_{>0}$ the set of points $\boldsymbol{x}_{\boldsymbol{w}}(t) = \arg\min_{\boldsymbol{x} \in \Omega} f_t(\boldsymbol{x}, \boldsymbol{w})$ for $t \in [0, \infty)$ forms a path through the interior of the polytope that we call the *weighted central path*. We call $\boldsymbol{x}_{\boldsymbol{w}}(0)$ a *weighted center* of the polytope and note that $\lim_{t \to \infty} \boldsymbol{x}_{\boldsymbol{w}}(0)$ is a solution to the linear program.

Our weighted path finding algorithm follows a simple iterative scheme. We assume we have a feasible point $\{\boldsymbol{x}, \boldsymbol{w}\} \in \{\Omega \times \mathbb{R}^m_{>0}\}$ and a weight function $\boldsymbol{g}(\boldsymbol{x}) : \Omega \to \mathbb{R}^m_{>0}$, such that for any point $\boldsymbol{x} \in \mathbb{R}^m_{>0}$ the function $\boldsymbol{g}(\boldsymbol{x})$ returns a good set of weights that suggest a possibly better weighted path. Although the weight function we use is motivated by the barrier function introduced in Theorem 6.3.5, we do not assume anything on $\boldsymbol{g}$ in this section. Our algorithm then repeats the following.

1. If $\boldsymbol{x}$ close to $\arg\min_{\boldsymbol{y} \in \Omega} f_t(\boldsymbol{y}, \boldsymbol{w})$, then increase t.
2. Otherwise, use projected Newton step to update $\boldsymbol{x}$ and move $\boldsymbol{w}$ closer to $\boldsymbol{g}(\boldsymbol{x})$.
3. Repeat.

[5] The "trigonometric barrier" we use arises as the (unique) solution of the ODE $\phi''' = 2(\phi'')^{3/2}$ such that the function value goes to infinity up at u_i and l_i.

In the remainder of this section we present how we measure both the quality of a current feasible point $\{\boldsymbol{x}, \boldsymbol{w}\} \in \{\Omega \times \mathbb{R}^m_{>0}\}$ and the quality of the weight function. In Section 6.5.4 we derive and present both how we measure how close $\{\boldsymbol{x}, \boldsymbol{w}\}$ is to the weighted central path and the step we take to improve this *centrality*. Then in Section 6.5.5 we present how we measure the quality of a weight function, i.e. how good the weighted paths it finds are.

6.5.4 Measuring Centrality.

Here we explain how we measure the distance from $\boldsymbol{x}$ to the minimum of $f_t(\boldsymbol{x}, \boldsymbol{w})$ for fixed $\boldsymbol{w}$. This distance is a measure of how close $\boldsymbol{x}$ is to the weighted central path and we refer to it as the *centrality* of $\boldsymbol{x}$, denote $\delta_t(\boldsymbol{x}, \boldsymbol{w})$.

To motivate our centrality measure, we first compute a projected Newton step for $\boldsymbol{x}$. For all $\boldsymbol{x} \in \Omega$, we define $\boldsymbol{\phi}(\boldsymbol{x}) \in \mathbb{R}^m$ by $\boldsymbol{\phi}(\boldsymbol{x})_i = \phi_i(\boldsymbol{x}_i)$ for $i \in [m]$. We define $\boldsymbol{\phi}'(\boldsymbol{x})$, $\boldsymbol{\phi}''(\boldsymbol{x})$, and $\boldsymbol{\phi}'''(\boldsymbol{x})$ similarly and let $\boldsymbol{\Phi}, \boldsymbol{\Phi}', \boldsymbol{\Phi}'', \boldsymbol{\Phi}'''$ denote the diagonal matrices corresponding to these matrices. Using this, we have[6]

$$\nabla_x f_t(\boldsymbol{x}, \boldsymbol{w}) = t \cdot \boldsymbol{c} + \boldsymbol{w}\boldsymbol{\phi}'(\boldsymbol{x}) \quad \text{and} \quad \nabla_{xx} f_t(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{W}\boldsymbol{\Phi}''(\boldsymbol{x}).$$

Therefore, a Newton step for $\boldsymbol{x}$ is given by

$$\begin{aligned} \boldsymbol{h}_t(\boldsymbol{x}, \boldsymbol{w}) &= -\left(\boldsymbol{W}\boldsymbol{\Phi}''(\boldsymbol{x})\right)^{-1/2} \boldsymbol{P}_{\boldsymbol{A}^T(\boldsymbol{W}\boldsymbol{\Phi}''(\boldsymbol{x}))^{-1/2}} \left(\boldsymbol{W}\boldsymbol{\Phi}''(\boldsymbol{x})\right)^{-1/2} \nabla_x f_t(\boldsymbol{x}, \boldsymbol{w}) \\ &= -\boldsymbol{\Phi}''(\boldsymbol{x})^{-1/2} \boldsymbol{P}_{\boldsymbol{x},\boldsymbol{w}} \boldsymbol{W}^{-1} \boldsymbol{\Phi}''(\boldsymbol{x})^{-1/2} \nabla_x f_t(\boldsymbol{x}, \boldsymbol{w}) \end{aligned} \tag{6.13}$$

where $\boldsymbol{P}_{\boldsymbol{A}^T(\boldsymbol{W}\boldsymbol{\Phi}''(\boldsymbol{x}))^{-1/2}}$ is the orthogonal projection onto the kernel of $\boldsymbol{A}^T(\boldsymbol{W}\boldsymbol{\Phi}''(\boldsymbol{x}))^{-1/2}$ and $\boldsymbol{P}_{\boldsymbol{x},\boldsymbol{w}}$ is the orthogonal projection onto the kernel of $\boldsymbol{A}^T(\boldsymbol{\Phi}''(\boldsymbol{x}))^{-1/2}$ with respect to the norm $\|\cdot\|_{\boldsymbol{W}}$, i.e.

$$\boldsymbol{P}_{\boldsymbol{x},\boldsymbol{w}} \stackrel{\text{def}}{=} \boldsymbol{I} - \boldsymbol{W}^{-1}\boldsymbol{A}_x \left(\boldsymbol{A}_x^T \boldsymbol{W}^{-1} \boldsymbol{A}_x\right)^{-1} \boldsymbol{A}_x^T \quad \text{for} \quad \boldsymbol{A}_x \stackrel{\text{def}}{=} \boldsymbol{\Phi}''(\boldsymbol{x})^{-1/2}\boldsymbol{A}. \tag{6.14}$$

As with standard convergence analysis of Newton's method, we wish to keep the Newton step size in the Hessian norm, i.e. $\|\boldsymbol{h}_t(\boldsymbol{x}, \boldsymbol{w})\|_{\boldsymbol{W}\boldsymbol{\Phi}''(\boldsymbol{x})} = \|\sqrt{\boldsymbol{\phi}''(\boldsymbol{x})}\boldsymbol{h}_t(\boldsymbol{x}, \boldsymbol{w})\|_{\boldsymbol{W}}$, small and the multiplicative change in the Hessian, $\|\sqrt{\boldsymbol{\phi}''(\boldsymbol{x})}\boldsymbol{h}_t(\boldsymbol{x}, \boldsymbol{w})\|_\infty$, small (See Lemma 6.5.2). While in the unweighted case we can bound the multiplicative change by the change in the hessian norm (since $\|\cdot\|_\infty \leq \|\cdot\|_2$), here we would like to use small weights and this comparison would be insufficient.

To track both these quantities simultaneously, we define the *mixed norm* for all $\boldsymbol{y} \in \mathbb{R}^m$ by

$$\|\boldsymbol{y}\|_{\boldsymbol{w}+\infty} \stackrel{\text{def}}{=} \|\boldsymbol{y}\|_\infty + C_{\text{norm}} \|\boldsymbol{y}\|_{\boldsymbol{W}} \tag{6.15}$$

for some $C_{\text{norm}} > 0$ that we define later. Note that $\|\cdot\|_{\boldsymbol{w}+\infty}$ is indeed a norm for $\boldsymbol{w} \in \mathbb{R}^m_{>0}$ as in this case both $\|\cdot\|_\infty$ and $\|\cdot\|_{\boldsymbol{W}}$ are norms. However, rather than measuring centrality by the quantity $\|\sqrt{\boldsymbol{\phi}''(\boldsymbol{x})}\boldsymbol{h}_t(\boldsymbol{x}, \boldsymbol{w})\|_{\boldsymbol{w}+\infty} = \left\|\boldsymbol{P}_{\boldsymbol{x},\boldsymbol{w}}\left(\frac{\nabla_x f_t(\boldsymbol{x},\boldsymbol{w})}{\boldsymbol{w}\sqrt{\boldsymbol{\phi}''}}\right)\right\|_{\boldsymbol{w}+\infty}$, we instead find it more convenient to use the following idealized form

$$\delta_t(\boldsymbol{x}, \boldsymbol{w}) \stackrel{\text{def}}{=} \min_{\boldsymbol{\eta} \in \mathbb{R}^n} \left\| \frac{\nabla_x f_t(\boldsymbol{x}, \boldsymbol{w}) - \boldsymbol{A}\boldsymbol{\eta}}{\boldsymbol{w}\sqrt{\boldsymbol{\phi}''(\boldsymbol{x})}} \right\|_{\boldsymbol{w}+\infty}.$$

We justify this definition by showing these two quantities differ by at most a multiplicative factor of $\|\boldsymbol{P}_{\boldsymbol{x},\boldsymbol{w}}\|_{\boldsymbol{w}+\infty}$ as follows

$$\delta_t(\boldsymbol{x}, \boldsymbol{w}) \leq \left\|\sqrt{\boldsymbol{\phi}''(\boldsymbol{x})}\boldsymbol{h}_t(\boldsymbol{x}, \boldsymbol{w})\right\|_{\boldsymbol{w}+\infty} \leq \|\boldsymbol{P}_{\boldsymbol{x},\boldsymbol{w}}\|_{\boldsymbol{w}+\infty} \cdot \delta_t(\boldsymbol{x}, \boldsymbol{w}). \tag{6.16}$$

[6] Recall that $\boldsymbol{w}\boldsymbol{\phi}'(\boldsymbol{x})$ denotes the entry-wise multiplication of the vectors $\boldsymbol{w}$ and $\boldsymbol{\phi}'(\boldsymbol{x})$.

This a direct consequence of the more general Lemma 6.11.1 that we prove in the appendix.

We summarize this section with the following definition.

Definition 6.5.3 (Centrality Measure)**.** For $\{\boldsymbol{x}, \boldsymbol{w}\} \in \{\Omega \times \mathbb{R}^m_{>0}\}$ and $t \geq 0$, we let $\boldsymbol{h}_t(\boldsymbol{x}, \boldsymbol{w})$ denote the *projected newton step* for $\boldsymbol{x}$ on the penalized objective f_t given by

$$\boldsymbol{h}_t(\boldsymbol{x}, \boldsymbol{w}) \stackrel{\text{def}}{=} -\frac{1}{\sqrt{\vec{\phi}''(\boldsymbol{x})}} \boldsymbol{P}_{\boldsymbol{x},\boldsymbol{w}} \left(\frac{\nabla_x f_t(\boldsymbol{x}, \boldsymbol{w})}{\boldsymbol{w}\sqrt{\vec{\phi}''(\boldsymbol{x})}} \right)$$

where $\boldsymbol{P}_{\boldsymbol{x},\boldsymbol{w}}$ is the orthogonal projection onto the kernel of $\boldsymbol{A}^T(\boldsymbol{\Phi}'')^{-1/2}$ with respect to the norm $\|\cdot\|_{\boldsymbol{W}}$ (see (6.15)). We measure the *centrality* of $\{\boldsymbol{x}, \boldsymbol{w}\}$ by

$$\delta_t(\boldsymbol{x}, \boldsymbol{w}) \stackrel{\text{def}}{=} \min_{\boldsymbol{\eta} \in \mathbb{R}^n} \left\| \frac{\nabla_x f_t(\boldsymbol{x}, \boldsymbol{w}) - \boldsymbol{A}\boldsymbol{\eta}}{\boldsymbol{w}\sqrt{\boldsymbol{\phi}''(\boldsymbol{x})}} \right\|_{\boldsymbol{w}+\infty} \tag{6.17}$$

where for all $\boldsymbol{y} \in \mathbb{R}^m$ we let $\|\boldsymbol{y}\|_{\boldsymbol{w}+\infty} \stackrel{\text{def}}{=} \|\boldsymbol{y}\|_\infty + C_{\text{norm}}\|\boldsymbol{y}\|_{\boldsymbol{W}}$ for some $C_{\text{norm}} > 0$ we define later.

6.5.5 The Weight Function

With the Newton step and centrality conditions defined, the specification of our algorithm becomes more clear. Our algorithm is as follows

1. If $\delta_t(\boldsymbol{x}, \boldsymbol{w})$ is small, then increase t.
2. Set $\boldsymbol{x}^{(\text{new})} \leftarrow \boldsymbol{x} + \boldsymbol{h}_t(\boldsymbol{x}, \boldsymbol{w})$ and move $\boldsymbol{w}^{(\text{new})}$ towards $\boldsymbol{g}(\boldsymbol{x}^{(\text{new})})$.
3. Repeat.

To prove this algorithm converges, we need to show what happens to $\delta_t(\boldsymbol{x}, \boldsymbol{w})$ when we change t, $\boldsymbol{x}$ and $\boldsymbol{w}$. At the heart of this chapter is understanding what conditions we need to impose on the weight function $\boldsymbol{g}(\boldsymbol{x}) : \Omega \to \mathbb{R}^m_{>0}$ so that we can bound this change in $\delta_t(\boldsymbol{x}, \boldsymbol{w})$ and hence achieve fast converge rates. In Lemma 6.6.1 we show that the effect of changing t on δ_t is bounded by C_{norm} and $\|\boldsymbol{g}(\boldsymbol{x})\|_1$, in Lemma 6.6.2 we show that the effect that a Newton Step on $\boldsymbol{x}$ has on δ_t is bounded by $\left\|\boldsymbol{P}_{\boldsymbol{x},\boldsymbol{g}(\boldsymbol{x})}\right\|_{\boldsymbol{g}(\boldsymbol{x})+\infty}$, and in Lemma 6.6.3 and 6.6.4 we show the change of $\boldsymbol{w}$ as $\boldsymbol{g}(\boldsymbol{x})$ changes is bounded by $\left\|\boldsymbol{G}(\boldsymbol{x})^{-1}\boldsymbol{G}'(\boldsymbol{x})(\boldsymbol{\Phi}''(\boldsymbol{x}))^{-1/2}\right\|_{\boldsymbol{g}(\boldsymbol{x})+\infty}$.

Hence for the remainder of the chapter we assume we have a weight function $\boldsymbol{g}(\boldsymbol{x}) : \Omega \to \mathbb{R}^m_{>0}$ and make the following assumptions regarding our weight function. In Section 6.7 we prove that such weight function exists.

Definition 6.5.4 (Weight Function)**.** A *weight function* is a differentiable function from $\boldsymbol{g} : \Omega \to \mathbb{R}^m_{>0}$ such that for constants $c_1(\boldsymbol{g})$, $c_\gamma(\boldsymbol{g})$, and $c_\delta(\boldsymbol{g})$, we have the following for all $\boldsymbol{x} \in \Omega$:

- *Size* : The size $c_1(\boldsymbol{g}) = \|\boldsymbol{g}(\boldsymbol{x})\|_1$.
- *Slack Sensitivity*: The slack sensitivity $c_\gamma(\boldsymbol{g})$ satisfies $1 \leq c_\gamma(\boldsymbol{g}) \leq \frac{5}{4}$ and $\|\boldsymbol{P}_{\boldsymbol{x},\boldsymbol{w}}\|_{\boldsymbol{w}+\infty} \leq c_\gamma(\boldsymbol{g})$ for any $\boldsymbol{w}$ such that $\frac{4}{5}\boldsymbol{g}(\boldsymbol{x}) \leq \boldsymbol{w} \leq \frac{5}{4}\boldsymbol{g}(\boldsymbol{x})$.
- *Step Consistency* : The step consistency $c_\delta(\boldsymbol{g})$ satisfies $c_\delta(\boldsymbol{g}) \cdot c_\gamma(\boldsymbol{g}) < 1$ and

$$\left\|\boldsymbol{G}(\boldsymbol{x})^{-1}\boldsymbol{G}'(\boldsymbol{x})(\boldsymbol{\Phi}''(\boldsymbol{x}))^{-1/2}\right\|_{\boldsymbol{g}(\boldsymbol{x})+\infty} \leq c_\delta \leq 1.$$

- *Uniformity* : The weight function satisfies $\|\boldsymbol{g}(\boldsymbol{x})\|_\infty \leq 2$.

6.5.6 Notation

Here we introduce various notation that we will use throughout the chapter (excluding Section 6.13). This should be used primarily for reference as we reintroduce notation as needed later in the chapter. For a matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and vector $\boldsymbol{x} \in \mathbb{R}^m_{>0}$ we define the following matrices and vectors

- Projection matrix $\boldsymbol{P}_{\boldsymbol{A}}(\boldsymbol{x}) \in \mathbb{R}^{m \times m}$: $\boldsymbol{P}_{\boldsymbol{A}}(\boldsymbol{x}) \stackrel{\text{def}}{=} \boldsymbol{X}^{1/2}\boldsymbol{A}(\boldsymbol{A}^T\boldsymbol{X}\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{X}^{1/2}$.
- Leverage scores $\boldsymbol{\sigma}_{\boldsymbol{A}}(\boldsymbol{x}) \in \mathbb{R}^m$: $\boldsymbol{\sigma}_{\boldsymbol{A}}(\boldsymbol{x}) \stackrel{\text{def}}{=} \boldsymbol{diag}(\boldsymbol{P}_{\boldsymbol{A}}(\boldsymbol{x}))$.
- Leverage matrix $\boldsymbol{\Sigma}_{\boldsymbol{A}}(\boldsymbol{x}) \in \mathbb{R}^{m \times m}$: $\boldsymbol{\Sigma}_{\boldsymbol{A}}(\boldsymbol{x}) \stackrel{\text{def}}{=} \boldsymbol{Diag}(\boldsymbol{P}_{\boldsymbol{A}}(\boldsymbol{x}))$.
- Projection Laplacian $\boldsymbol{\Lambda}_{\boldsymbol{A}}(\boldsymbol{x}) \in \mathbb{R}^{m \times m}$: $\boldsymbol{\Lambda}_{\boldsymbol{A}}(\boldsymbol{x}) \stackrel{\text{def}}{=} \boldsymbol{\Sigma}_{\boldsymbol{A}}(\boldsymbol{x}) - \boldsymbol{P}_{\boldsymbol{A}}(\boldsymbol{x})^{(2)}$.

6.6 Progressing Along Weighted Paths

In this section, we provide the main lemmas we need for an $\tilde{O}(\sqrt{\text{rank}(\boldsymbol{A})}\log(U/\varepsilon))$ iterations weighted path following algorithm for (6.9) assuming a weight function satisfying Definition 6.5.5. In Section 6.6.1, 6.6.2, and 6.6.3 we show how centrality, $\delta_t(\boldsymbol{x}, \boldsymbol{w})$, is affected by changing t, $\boldsymbol{x} \in \Omega$, and $\boldsymbol{w} \in \mathbb{R}^m_{>0}$ respectively. In Section 6.6.5 we then show how to use these Lemmas to improve centrality using approximate computations of the weight function, $\boldsymbol{g} : \Omega \to \mathbb{R}^m_{>0}$.

6.6.1 Changing t

Here we bound how much centrality increases as we increase t. We show that this rate of increase is governed by C_{norm} and $\|\boldsymbol{w}\|_1$.

Lemma 6.6.1. *For all $\{\boldsymbol{x}, \boldsymbol{w}\} \in \{\Omega \times \mathbb{R}^m_{>0}\}$, $t > 0$ and $\alpha \geq 0$, we have*

$$\delta_{(1+\alpha)t}(\boldsymbol{x}, \boldsymbol{w}) \quad \leq \quad (1+\alpha)\delta_t(\boldsymbol{x}, \boldsymbol{w}) + \alpha\left(1 + C_{norm}\sqrt{\|\boldsymbol{w}\|_1}\right).$$

Proof. Let $\boldsymbol{\eta}_t \in \mathbb{R}^n$ be such that

$$\delta_t(\boldsymbol{x}, \boldsymbol{w}) = \left\| \frac{\nabla_x f_t(\boldsymbol{x}, \boldsymbol{w}) + \boldsymbol{A}\boldsymbol{\eta}_t}{\boldsymbol{w}\sqrt{\boldsymbol{\phi}''(\boldsymbol{x})}} \right\|_{\boldsymbol{w}+\infty} = \left\| \frac{t \cdot \boldsymbol{c} + \boldsymbol{w}\boldsymbol{\phi}'(\boldsymbol{x}) + \boldsymbol{A}\boldsymbol{\eta}_t}{\boldsymbol{w}\sqrt{\boldsymbol{\phi}''(\boldsymbol{x})}} \right\|_{\boldsymbol{w}+\infty}.$$

Applying this to the definition of $\delta_{(1+\alpha)t}$ and using that $\|\cdot\|_{\boldsymbol{w}+\infty}$ is a norm then yields

$$\begin{aligned}
\delta_{(1+\alpha)t}(\boldsymbol{x}, \boldsymbol{w}) &= \min_{\boldsymbol{\eta} \in \mathbb{R}^n} \left\| \frac{(1+\alpha)t \cdot \boldsymbol{c} + \boldsymbol{w}\boldsymbol{\phi}'(\boldsymbol{x}) + \boldsymbol{A}\boldsymbol{\eta}}{\boldsymbol{w}\sqrt{\boldsymbol{\phi}''(\boldsymbol{x})}} \right\|_{\boldsymbol{w}+\infty} \\
&\leq \left\| \frac{(1+\alpha)t \cdot \boldsymbol{c} + \boldsymbol{w}\boldsymbol{\phi}'(\boldsymbol{x}) + \boldsymbol{A}(1+\alpha)\boldsymbol{\eta}_t}{\boldsymbol{w}\sqrt{\boldsymbol{\phi}''(\boldsymbol{x})}} \right\|_{\boldsymbol{w}+\infty} \\
&\leq (1+\alpha)\left\| \frac{t \cdot \boldsymbol{c} + \boldsymbol{w}\boldsymbol{\phi}'(\boldsymbol{x}) + \boldsymbol{A}\boldsymbol{\eta}_t}{\boldsymbol{w}\sqrt{\boldsymbol{\phi}''(\boldsymbol{x})}} \right\|_{\boldsymbol{w}+\infty} + \alpha\left\| \frac{\boldsymbol{w}\boldsymbol{\phi}'(\boldsymbol{x})}{\boldsymbol{w}\sqrt{\boldsymbol{\phi}''(\boldsymbol{x})}} \right\|_{\boldsymbol{w}+\infty} \\
&= (1+\alpha)\delta_t(\boldsymbol{x}, \boldsymbol{w}) + \alpha\left(\left\| \frac{\boldsymbol{\phi}'(\boldsymbol{x})}{\sqrt{\boldsymbol{\phi}''(\boldsymbol{x})}} \right\|_\infty + C_{\text{norm}} \left\| \frac{\boldsymbol{\phi}'(\boldsymbol{x})}{\sqrt{\boldsymbol{\phi}''(\boldsymbol{x})}} \right\|_{\boldsymbol{W}} \right)
\end{aligned}$$

Using that $|\boldsymbol{\phi}_i'(\boldsymbol{x})| \leq \sqrt{\boldsymbol{\phi}_i''(\boldsymbol{x})}$ for all $i \in [m]$ and $\boldsymbol{x} \in \mathbb{R}^m$ by Equation (6.11) yields the result. □

6.6.2 Changing x

Here we analyze the effect of a Newton step on $\boldsymbol{x}$ on centrality. We show for sufficiently central $\{\boldsymbol{x}, \boldsymbol{w}\} \in \{\Omega \times \mathbb{R}^m_{>0}\}$ and $\boldsymbol{w}$ sufficiently close to $\boldsymbol{g}(\boldsymbol{x})$ Newton steps converge quadratically.

Lemma 6.6.2. *Let $\{\boldsymbol{x}_0, \boldsymbol{w}\} \in \{\Omega \times \mathbb{R}^m_{>0}\}$ such that $\delta_t(\boldsymbol{x}_0, \boldsymbol{w}) \leq \frac{1}{10}$ and $\frac{4}{5}\boldsymbol{g}(\boldsymbol{x}) \leq \boldsymbol{w} \leq \frac{5}{4}\boldsymbol{g}(\boldsymbol{x})$ and consider a Newton step $\boldsymbol{x}_1 = \boldsymbol{x}_0 + \boldsymbol{h}_t(\boldsymbol{x}, \boldsymbol{w})$. Then, $\delta_t(\boldsymbol{x}_1, \boldsymbol{w}) \leq 4\left(\delta_t(\boldsymbol{x}_0, \boldsymbol{w})\right)^2$.*

Proof. Let $\boldsymbol{\phi}_0 \stackrel{\text{def}}{=} \boldsymbol{\phi}(\boldsymbol{x}_0)$ and let $\boldsymbol{\phi}_1 \stackrel{\text{def}}{=} \boldsymbol{\phi}(\boldsymbol{x}_1)$. By the definition of $\boldsymbol{h}_t(\boldsymbol{x}_0, \boldsymbol{w})$ and the formula of $\boldsymbol{P}_{\boldsymbol{x}_0, \boldsymbol{w}}$ we know that there is some $\boldsymbol{\eta}_0 \in \mathbb{R}^n$ such that

$$-\sqrt{\vec{\phi}_0''}\boldsymbol{h}_t(\boldsymbol{x}_0, \boldsymbol{w}) = \frac{t \cdot \boldsymbol{c} + \boldsymbol{w}\boldsymbol{\phi}_0' - \boldsymbol{A}\boldsymbol{\eta}_0}{\boldsymbol{w}\sqrt{\boldsymbol{\phi}_0''}}.$$

Therefore, $\boldsymbol{A}\boldsymbol{\eta}_0 = \boldsymbol{c} + \boldsymbol{w}\boldsymbol{\phi}_0' + \boldsymbol{w}\boldsymbol{\phi}_0'' h_t(\boldsymbol{x}_0, \boldsymbol{w})$. Recalling the definition of δ_t this implies that

$$\begin{aligned}
\delta_t(\boldsymbol{x}_1, \boldsymbol{w}) &= \min_{\boldsymbol{\eta} \in \mathbb{R}^n} \left\| \frac{t \cdot \boldsymbol{c} + \boldsymbol{w}\vec{\boldsymbol{\phi}}_1' - \boldsymbol{A}\boldsymbol{\eta}}{\boldsymbol{w}\sqrt{\vec{\boldsymbol{\phi}}_1''}} \right\|_{\boldsymbol{w}+\infty} \leq \left\| \frac{t \cdot \boldsymbol{c} + \boldsymbol{w}\vec{\boldsymbol{\phi}}_1' - \boldsymbol{A}\boldsymbol{\eta}_0}{\boldsymbol{w}\sqrt{\vec{\boldsymbol{\phi}}_1''}} \right\|_{\boldsymbol{w}+\infty} \\
&\leq \left\| \frac{\boldsymbol{w}(\vec{\boldsymbol{\phi}}_1' - \boldsymbol{\phi}_0') - \boldsymbol{w}\boldsymbol{\phi}_0'' \boldsymbol{h}_t(\boldsymbol{x}_0, \boldsymbol{w})}{\boldsymbol{w}\sqrt{\vec{\boldsymbol{\phi}}_1''}} \right\|_{\boldsymbol{w}+\infty} = \left\| \frac{(\vec{\boldsymbol{\phi}}_1' - \boldsymbol{\phi}_0') - \boldsymbol{\phi}_0'' \boldsymbol{h}_t(\boldsymbol{x}_0, \boldsymbol{w})}{\sqrt{\vec{\boldsymbol{\phi}}_1''}} \right\|_{\boldsymbol{w}+\infty}
\end{aligned}$$

By the mean value theorem, we have $\vec{\boldsymbol{\phi}}_1' - \boldsymbol{\phi}_0' = \vec{\boldsymbol{\phi}}''(\vec{\theta})\boldsymbol{h}_t(\boldsymbol{x}_0, \boldsymbol{w})$ for some $\vec{\theta}$ between $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$ coordinate-wise. Hence,

$$\begin{aligned}
\delta_t(\boldsymbol{x}_1, \boldsymbol{w}) &\leq \left\| \frac{\vec{\boldsymbol{\phi}}''(\vec{\theta})\boldsymbol{h}_t(\boldsymbol{x}_0, \boldsymbol{w}) - \boldsymbol{\phi}_0'' \boldsymbol{h}_t(\boldsymbol{x}_0, \boldsymbol{w})}{\sqrt{\vec{\boldsymbol{\phi}}_1''}} \right\|_{\boldsymbol{w}+\infty} = \left\| \frac{\left(\vec{\boldsymbol{\phi}}''(\vec{\theta}) - \vec{\boldsymbol{\phi}}_0''\right)}{\sqrt{\vec{\boldsymbol{\phi}}_1''}\sqrt{\boldsymbol{\phi}_0''}} (\sqrt{\boldsymbol{\phi}_0''}\boldsymbol{h}_t(\boldsymbol{x}_0, \boldsymbol{w})) \right\|_{\boldsymbol{w}+\infty} \\
&\leq \left\| \frac{\vec{\boldsymbol{\phi}}''(\vec{\theta}) - \vec{\boldsymbol{\phi}}_0''}{\sqrt{\vec{\boldsymbol{\phi}}_1''}\sqrt{\boldsymbol{\phi}_0''}} \right\|_{\infty} \cdot \left\| \sqrt{\boldsymbol{\phi}_0''}\boldsymbol{h}_t(\boldsymbol{x}_0, \boldsymbol{w}) \right\|_{\boldsymbol{w}+\infty}.
\end{aligned}$$

To bound the first term, we use Lemma 6.5.1 as follows

$$\begin{aligned}
\left\| \frac{\left(\vec{\boldsymbol{\phi}}''(\vec{\theta}) - \vec{\boldsymbol{\phi}}_0''\right)}{\sqrt{\vec{\boldsymbol{\phi}}_1''}\sqrt{\vec{\boldsymbol{\phi}}_0''}} \right\|_{\infty} &\leq \left\| \frac{\boldsymbol{\phi}''(\vec{\theta})}{\boldsymbol{\phi}_0''} - \mathbf{1} \right\|_{\infty} \cdot \left\| \frac{\sqrt{\vec{\boldsymbol{\phi}}_0''}}{\sqrt{\vec{\boldsymbol{\phi}}_1''}} \right\|_{\infty} \\
&\leq \left| \left(1 - \left\| \sqrt{\vec{\boldsymbol{\phi}}_0''}\boldsymbol{h}_t(\boldsymbol{x}_0, \boldsymbol{w}) \right\|_{\infty}\right)^{-2} - 1 \right| \cdot \left(1 - \left\| \sqrt{\vec{\boldsymbol{\phi}}_0''}\boldsymbol{h}_t(\boldsymbol{x}_0, \boldsymbol{w}) \right\|_{\infty}\right)^{-1}.
\end{aligned}$$

Using (6.16), i.e. Lemma 6.11.1, the bound $c_\gamma \leq 2$, and the assumption on $\delta_t(\boldsymbol{x}_0, \boldsymbol{w})$, we have

$$\left\| \sqrt{\vec{\boldsymbol{\phi}}_0''}\boldsymbol{h}_t(\boldsymbol{x}_0, \boldsymbol{w}) \right\|_{\infty} \leq \left\| \sqrt{\vec{\boldsymbol{\phi}}_0''}\boldsymbol{h}_t(\boldsymbol{x}_0, \boldsymbol{w}) \right\|_{\boldsymbol{w}+\infty} \leq c_\gamma \cdot \delta_t(\boldsymbol{x}_0, \boldsymbol{w}) \leq \frac{1}{5}.$$

Using $\left((1-t)^{-2} - 1\right) \cdot (1-t)^{-1} \leq 4t$ for $t \leq 1/5$, we have

$$\left\| \frac{\left(\vec{\boldsymbol{\phi}}''(\vec{\theta}) - \vec{\boldsymbol{\phi}}_0''\right)}{\sqrt{\vec{\boldsymbol{\phi}}_1''}\sqrt{\vec{\boldsymbol{\phi}}_0''}} \right\|_{\infty} \leq 4 \left\| \sqrt{\vec{\boldsymbol{\phi}}_0''} h_t(\boldsymbol{x}_0, \boldsymbol{w}) \right\|_{\infty}.$$

Combining the above formulas yields that $\delta_t(\boldsymbol{x}_1, \boldsymbol{w}) \leq 4\left(\delta_t(\boldsymbol{x}_0, \boldsymbol{w})\right)^2$ as desired. □

6.6.3 Changing $\boldsymbol{w}$

In the previous subsection we used the assumption that the weights, $\boldsymbol{w}$, were multiplicatively close to the output of the weight function, $\boldsymbol{g}(\boldsymbol{x})$, for the current point $\boldsymbol{x} \in \Omega$. In order to maintain this invariant when we change $\boldsymbol{x}$ we will need to change $\boldsymbol{w}$ to move it closer to $\boldsymbol{g}(\boldsymbol{x})$. Here we bound how much $\boldsymbol{g}(\boldsymbol{x})$ can move as we move $\boldsymbol{x}$ (Lemma 6.6.3) and we bound how much changing $\boldsymbol{w}$ can hurt centrality (Lemma 6.6.4). Together these lemmas will allow us to show that we can keep $\boldsymbol{w}$ close to $\boldsymbol{g}(\boldsymbol{x})$ while still improving centrality (Section 6.6.5).

Lemma 6.6.3. *For all $t \in [0,1]$, let $\boldsymbol{x}_t \stackrel{\text{def}}{=} \boldsymbol{x}_0 + t\boldsymbol{\Delta}_x$ for $\boldsymbol{\Delta}_x \in \mathbb{R}^m$, $\boldsymbol{x}_t \in \Omega$, $\boldsymbol{g}_t = \boldsymbol{g}(\boldsymbol{x}_t)$ and $\varepsilon = \left\|\sqrt{\vec{\phi}_0''}\boldsymbol{\Delta}_x\right\|_{\boldsymbol{g}_0+\infty} \leq 0.1$. Then*

$$\left\|\log\left(\boldsymbol{g}_1\right) - \log\left(\boldsymbol{g}_0\right)\right\|_{\boldsymbol{g}_0+\infty} \leq c_\delta\varepsilon(1+4\varepsilon) \leq 0.2$$

and for all $s, t \in [0,1]$ and for all $\boldsymbol{y} \in \mathbb{R}^m$ we have

$$\|\boldsymbol{y}\|_{\boldsymbol{g}_s+\infty} \leq (1+2\varepsilon)\|\boldsymbol{y}\|_{\boldsymbol{g}_t+\infty}. \tag{6.18}$$

Proof. Let $\boldsymbol{q} : [0,1] \to \mathbb{R}^m$ be given by $\boldsymbol{q}(t) \stackrel{\text{def}}{=} \log\left(\boldsymbol{g}_t\right)$ for all $t \in [0,1]$. Then, we have

$$\boldsymbol{q}'(t) = \boldsymbol{G}_t^{-1}\boldsymbol{G}_t'\boldsymbol{\Delta}_x.$$

Let $Q(t) \stackrel{\text{def}}{=} \|\vec{q}(t) - \vec{q}(0)\|_{\boldsymbol{g}_0+\infty}$. Using Jensen's inequality we have that for all $u \in [0,1]$,

$$Q(u) \leq \overline{Q}(u) \stackrel{\text{def}}{=} \int_0^u \left\|\boldsymbol{G}_t^{-1}\boldsymbol{G}_t'\left(\vec{\phi}_t''\right)^{-1/2}\right\|_{\boldsymbol{g}_0+\infty}\left\|\sqrt{\vec{\phi}_t''}\boldsymbol{\Delta}_x\right\|_{\boldsymbol{g}_0+\infty} dt.$$

Using Lemma 6.5.1 and $\varepsilon \leq \frac{1}{10}$, we have for all $t \in [0,1]$,

$$\begin{aligned}\left\|\sqrt{\vec{\phi}_t''}\boldsymbol{\Delta}_x\right\|_{\boldsymbol{g}_0+\infty} &\leq \left\|\sqrt{\vec{\phi}_t''}/\sqrt{\vec{\phi}_0''}\right\|_\infty\left\|\sqrt{\vec{\phi}_0''}\boldsymbol{\Delta}_x\right\|_{\boldsymbol{g}_0+\infty} \\ &\leq \left(1 - \left\|\sqrt{\vec{\phi}_0''}\boldsymbol{\Delta}_x\right\|_\infty\right)^{-1}\left\|\sqrt{\vec{\phi}_0''}\boldsymbol{\Delta}_x\right\|_{\boldsymbol{g}_0+\infty} \leq \frac{\varepsilon}{1-\varepsilon}.\end{aligned}$$

Thus, we have

$$\overline{Q}(u) \leq \frac{\varepsilon}{1-\varepsilon}\int_0^u \left\|\boldsymbol{G}_t^{-1}\boldsymbol{G}_t'\left(\vec{\phi}_t''\right)^{-1/2}\right\|_{\boldsymbol{g}_0+\infty} dt. \tag{6.19}$$

Note that $\overline{Q}$ is monotonically increasing. Let $\theta = \sup_{u\in[0,1]}\left\{\overline{Q}(u) \leq c_\delta\varepsilon(1+4\varepsilon)\right\}$. Since $\overline{Q}(\theta) \leq \frac{1}{2}$, we know that for all $s, t \in [0,\theta]$, we have

$$\left\|\frac{\boldsymbol{g}(\boldsymbol{x}_s) - \boldsymbol{g}(\boldsymbol{x}_t)}{\boldsymbol{g}(\boldsymbol{x}_t)}\right\|_\infty \leq \|\vec{q}(s) - \vec{q}(t)\|_\infty + \|\vec{q}(s) - \vec{q}(t)\|_\infty^2$$

and therefore

$$\|\boldsymbol{g}_s/\boldsymbol{g}_t\|_\infty \leq \left(1 + \|\vec{q}(s) - \vec{q}(t)\|_\infty + \|\vec{q}(s) - \vec{q}(t)\|_\infty^2\right)^2 \leq \left(1 + c_\delta\varepsilon(1+4\varepsilon)\right)^2$$

Consequently,

$$\|\boldsymbol{y}\|_{\boldsymbol{g}_s+\infty} \leq (1+c_\delta\varepsilon(1+4\varepsilon))\,\|\boldsymbol{y}\|_{\boldsymbol{g}_t+\infty} \leq (1+2\varepsilon)\,\|\boldsymbol{y}\|_{\boldsymbol{g}_t+\infty}.$$

Using (6.19), we have for all $u \in [0,\theta]$,

$$\begin{aligned} Q(u) \leq \overline{Q}(u) &\leq \frac{\varepsilon}{1-\varepsilon}\int_0^u \left\|\boldsymbol{G}_t^{-1}\boldsymbol{G}_t'\left(\vec{\phi}_t''\right)^{-1/2}\right\|_{\boldsymbol{g}_0+\infty} dt \\ &\leq \frac{\varepsilon}{1-\varepsilon}\int_0^u (1+2\varepsilon)\left\|\boldsymbol{G}_t^{-1}\boldsymbol{G}_t'\left(\vec{\phi}_t''\right)^{-1/2}\right\|_{\boldsymbol{g}_t+\infty} dt \\ &\leq \frac{\varepsilon}{1-\varepsilon}(1+2\varepsilon)\,c_\delta\theta \leq c_\delta\varepsilon(1+4\varepsilon). \end{aligned}$$

Consequently, we have that $\theta = 1$ and we have the desired result with $Q(1) \leq c_\delta\varepsilon(1+4\varepsilon) < \frac{1}{5}$. □

Lemma 6.6.4. *Let $\boldsymbol{v},\boldsymbol{w} \in \mathbb{R}^m_{>0}$ such that $\varepsilon = \|\log(\boldsymbol{w}) - \log(\boldsymbol{v})\|_{\boldsymbol{w}+\infty} \leq 0.1$. Then for $\boldsymbol{x} \in \Omega$ we have*

$$\delta_t(\boldsymbol{x},\boldsymbol{v}) \leq (1+4\varepsilon)(\delta_t(\boldsymbol{x},\boldsymbol{w}) + \varepsilon).$$

Proof. Let $\boldsymbol{\eta}_w$ be such that

$$\delta_t(\boldsymbol{x},\boldsymbol{w}) = \left\|\frac{\boldsymbol{c}+\boldsymbol{w}\vec{\phi}'(\boldsymbol{x}) - \boldsymbol{A}\boldsymbol{\eta}_w}{\boldsymbol{w}\sqrt{\vec{\phi}''(\boldsymbol{x})}}\right\|_{\boldsymbol{w}+\infty} \tag{6.20}$$

Furthermore, the assumption shows that $(1+\varepsilon)^{-2}\boldsymbol{w}_i \leq \boldsymbol{v}_i \leq (1+\varepsilon)^2\boldsymbol{w}_i$ for all i. Using these, we bound the energy with the new weights as follows

$$\begin{aligned} \delta_t(\boldsymbol{x},\boldsymbol{v}) = \min_{\eta}\left\|\frac{\boldsymbol{c}+\boldsymbol{v}\vec{\phi}'(\boldsymbol{x}) - \boldsymbol{A}\boldsymbol{\eta}}{\boldsymbol{v}\sqrt{\vec{\phi}''(\boldsymbol{x})}}\right\|_{\boldsymbol{v}+\infty} &\leq \left\|\frac{\boldsymbol{c}+\boldsymbol{v}\vec{\phi}'(\boldsymbol{x}) - \boldsymbol{A}\boldsymbol{\eta}_w}{\boldsymbol{v}\sqrt{\vec{\phi}''(\boldsymbol{x})}}\right\|_{\boldsymbol{v}+\infty} \\ &\leq (1+\varepsilon)\left\|\frac{\boldsymbol{c}+\boldsymbol{v}\vec{\phi}'(\boldsymbol{x}) - \boldsymbol{A}\boldsymbol{\eta}_w}{\boldsymbol{v}\sqrt{\vec{\phi}''(\boldsymbol{x})}}\right\|_{\boldsymbol{w}+\infty} \\ &\leq (1+\varepsilon)\cdot\left(\left\|\frac{\boldsymbol{c}+\boldsymbol{w}\vec{\phi}'(\boldsymbol{x}) - \boldsymbol{A}\boldsymbol{\eta}_w}{\boldsymbol{v}\sqrt{\vec{\phi}''(\boldsymbol{x})}}\right\|_{\boldsymbol{w}+\infty} + \left\|\frac{(\boldsymbol{v}-\boldsymbol{w})\vec{\phi}'(\boldsymbol{x})}{\boldsymbol{v}\sqrt{\vec{\phi}''(\boldsymbol{x})}}\right\|_{\boldsymbol{w}+\infty}\right) \\ &\leq (1+\varepsilon)^3\,\delta_t(\boldsymbol{x},\boldsymbol{w}) + (1+\varepsilon)\cdot\left\|\frac{\boldsymbol{\phi}'(\boldsymbol{x})}{\sqrt{\boldsymbol{\phi}''(\boldsymbol{x})}}\right\|_\infty \cdot \left\|\frac{(\boldsymbol{v}-\boldsymbol{w})}{\boldsymbol{v}}\right\|_{\boldsymbol{w}+\infty} \end{aligned}$$

Using that $|\phi_i'(\boldsymbol{x})| \leq \sqrt{\phi_i''(\boldsymbol{x})}$ for all $i \in [m]$ by Equation (6.11), we have that

$$\begin{aligned} \delta_t(\boldsymbol{x},\boldsymbol{v}) &\leq (1+\varepsilon)^3\delta_t(\boldsymbol{x},\boldsymbol{w}) + (1+\varepsilon)^2\varepsilon \\ &\leq (1+4\varepsilon)\left(\delta_t(\boldsymbol{x},\boldsymbol{w}) + \varepsilon\right). \end{aligned}$$

□

6.6.4 The Chasing 0 Game

In the previous subsection, we saw how much the weight function, $\boldsymbol{g}(\boldsymbol{x})$, can change after a Newton step on $\boldsymbol{x}$ and we bounded how much we can move the weights without affecting centrality too much.

Here we study how to correctly move the weights even when we cannot compute the weight function exactly. To effectively use multiplicative approximations to the weight function, we need to smooth out changes to the weights by using some slowly changing approximation to the weight function. In this subsection, we present the smoothing problem in a general form that we call the *chasing 0 game* and we provide an effective strategy for playing this game. In next subsection, we show how to use this strategy to produce a efficient weighted path following scheme.

The *chasing 0 game* is as follows. There is a player, an adversary, and a point $\boldsymbol{x} \in \mathbb{R}^m$. The goal of the player is to keep the point close to $\mathbf{0}$ in ℓ_∞ norm and the goal of the adversary tries to move $\boldsymbol{x}$ away from $\mathbf{0} \in \mathbb{R}^m$. The game proceeds for an infinite number of iterations where in each iteration the adversary moves the current point $\boldsymbol{x}^{(k)} \in \mathbb{R}^m$ to some new point $\boldsymbol{y}^{(k)} \in \mathbb{R}^m$ and the player needs to respond. The player does not know $\boldsymbol{x}^{(k)}$, $\boldsymbol{y}^{(k)}$, or the move of the adversary. All the player knows is that the adversary moved the point within some convex set $U^{(k)}$ and the player knows some $\boldsymbol{z}^{(k)} \in \mathbb{R}^n$ that is close to $\boldsymbol{y}^{(k)}$ in ℓ_∞ norm. With this information the player is allowed to move the point a little more than the adversary. Formally, the player is allowed to set the next point to $\boldsymbol{x}^{(k+1)} \in \mathbb{R}^m$ such that $\boldsymbol{\Delta}^{(k)} \stackrel{\text{def}}{=} \boldsymbol{x}^{(k+1)} - \boldsymbol{y}^{(k)} \in (1+\varepsilon)U$ for some fixed $\varepsilon > 0$.

The question we would like to address is, how close the player can keep $\boldsymbol{x}^{(k+1)}$ to $\mathbf{0}$ in ℓ_∞ norm? In particular, we would like an efficient strategy for computing $\boldsymbol{\Delta}^{(k)}$ such that $\|\boldsymbol{x}^{(k)}\|_\infty$ is bounded for all $k \geq 0$.

Algorithm 7: Chasing 0 Game

Given $R > 0, \varepsilon > 0, \boldsymbol{x}^{(0)} \in \mathbb{R}^m$.
for $k = 1, 2, \cdots$ **do**
  The adversary announces symmetric convex set $U^{(k)} \subseteq \mathbb{R}^n$ and $\boldsymbol{u}^{(k)} \in U^{(k)}$.
  The adversary sets $\boldsymbol{y}^{(k)} := \boldsymbol{x}^{(k)} + \boldsymbol{u}^{(k)}$.
  The adversary announces $\boldsymbol{z}^{(k)}$ such that $\|\boldsymbol{z}^{(k)} - \boldsymbol{y}^{(k)}\|_\infty \leq R$.
  The player chooses $\boldsymbol{\Delta}^{(k)} \in (1+\varepsilon)\, U^{(k)}$.
  The player sets $\boldsymbol{x}^{(k+1)} = \boldsymbol{y}^{(k)} + \boldsymbol{\Delta}^{(k)}$.
end

We show that assuming that the $U^{(k)}$ are sufficiently bounded then there is strategy that the player can follow to ensure that that $\|\boldsymbol{x}^{(k)}\|_\infty$ is never too large. Our strategy simply consists of taking "gradient steps" using the following potential function.

Definition 6.6.5. For any $\mu \geq 0$ let $p_\mu : \mathbb{R} \to \mathbb{R}$ and $\Phi_\mu : \mathbb{R}^m \to \mathbb{R}$ be given by

$$\forall x \in \mathbb{R} \quad : \quad p_\mu(x) \stackrel{\text{def}}{=} e^{\mu x} + e^{-\mu x} \quad \text{and} \quad \Phi_\mu(\boldsymbol{x}) \stackrel{\text{def}}{=} \sum_{i \in [m]} p_\mu(x_i).$$

In other words, for all k we simply set $\boldsymbol{\Delta}^{(k)}$ to be the vector in $(1+\varepsilon)U^{(k)}$ that best minimizes the potential function of the observed position, i.e. $\Phi_\mu(\boldsymbol{z}^{(k)})$ for an appropriate choice of μ. In the following theorem we show that this suffices to keep $\Phi_\mu(\boldsymbol{x}^{(k)})$ small and that small $\Phi_\mu(\boldsymbol{x}^{(k)})$ implies small $\|\boldsymbol{x}^{(k)}\|_\infty$ and hence has the desired properties.

Theorem 6.6.6. *Suppose that each $U^{(k)}$ is a symmetric convex set that contains an ℓ_∞ ball of radius r_k and is contained in a ℓ_∞ ball of radius $R_k \leq R$. Let $0 < \varepsilon < \frac{1}{5}$ and consider the strategy*

$$\boldsymbol{\Delta}^{(k)} = (1+\varepsilon) \operatorname*{arg\,min}_{\boldsymbol{\Delta} \in U^{(k)}} \left\langle \nabla \Phi_\mu(\boldsymbol{z}^{(k)}), \boldsymbol{\Delta} \right\rangle \quad \textit{where} \quad \mu = \frac{\varepsilon}{12R}.$$

Let $\tau \stackrel{\text{def}}{=} \max_k \frac{R_k}{r_k}$ *and suppose* $\Phi_\mu(\boldsymbol{x}^{(0)}) \le \frac{12m\tau}{\varepsilon}$ *(or more specifically* $\|\boldsymbol{x}^{(0)}\|_\infty \le \frac{12R}{\varepsilon}\log\left(\frac{6\tau}{\varepsilon}\right)$ *) then*

$$\forall k \ge 0 \quad : \quad \Phi_\mu(\boldsymbol{x}^{(k+1)}) \le \left(1 - \frac{\varepsilon^2 r_k}{24R}\right)\Phi_\mu(\boldsymbol{x}^{(k)}) + \varepsilon m \frac{R_k}{2R} \le \frac{12m\tau}{\varepsilon}.$$

In particular, we have $\|\boldsymbol{x}^{(k)}\|_\infty \le \frac{12R}{\varepsilon}\log\left(\frac{12m\tau}{\varepsilon}\right)$.

To prove Theorem 6.6.6 we first provide the following lemma regarding properties of the potential function Φ_μ.

Lemma 6.6.7. *For all* $\boldsymbol{x} \in \mathbb{R}^m$*, we have*

$$e^{\mu\|\boldsymbol{x}\|_\infty} \le \Phi_\mu(\boldsymbol{x}) \le 2me^{\mu\|\boldsymbol{x}\|_\infty} \quad \text{and} \quad \mu\Phi_\mu(\boldsymbol{x}) - 2\mu m \le \|\nabla\Phi_\mu(\boldsymbol{x})\|_1 \tag{6.21}$$

Furthermore, for any symmetric convex set $U \subseteq \mathbb{R}^m$ *and any* $\boldsymbol{x} \in \mathbb{R}^m$*, let* $\boldsymbol{x}^\flat \stackrel{\text{def}}{=} \arg\max_{\boldsymbol{y}\in U}\langle \boldsymbol{x}, \boldsymbol{y}\rangle$ *and* $\|\boldsymbol{x}\|_U \stackrel{\text{def}}{=} \max_{\boldsymbol{y}\in U}\langle \boldsymbol{x}, \boldsymbol{y}\rangle$*. Then for all* $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^m$ *with* $\|\boldsymbol{x} - \boldsymbol{y}\|_\infty \le \delta \le \frac{1}{5\mu}$ *we have*

$$e^{-\mu\delta}\|\nabla\Phi_\mu(\boldsymbol{y})\|_U - \mu\|\nabla\Phi_\mu(\boldsymbol{y})^\flat\|_1 \le \left\langle \nabla\Phi_\mu(\boldsymbol{x}), \nabla\Phi_\mu(\boldsymbol{y})^\flat \right\rangle \le e^{\mu\delta}\|\nabla\Phi_\mu(\boldsymbol{y})\|_U + \mu e^{\mu\delta}\|\nabla\Phi_\mu(\boldsymbol{y})^\flat\|_1. \tag{6.22}$$

If additionally U *is contained in a* ℓ_∞ *ball of radius* R *then*

$$e^{-\mu\delta}\|\nabla\Phi_\mu(\boldsymbol{y})\|_U - \mu m R \le \|\nabla\Phi_\mu(\boldsymbol{x})\|_U \le e^{\mu\delta}\|\nabla\Phi_\mu(\boldsymbol{y})\|_U + \mu e^{\mu\delta} m R. \tag{6.23}$$

Proof. First we note that for all $x \in \mathbb{R}$ we have

$$e^{\mu|x|} \le p_\mu(x) \le 2e^{\mu|x|} \quad \text{and} \quad p'_\mu(x) = \mu\,\mathrm{sign}(x)\left(e^{\mu|x|} - e^{-\mu|x|}\right)$$

and therefore we have (6.21).

Next let $x, y \in \mathbb{R}$ such that $|x - y| \le \delta$. Note that $\left|p'_\mu(x)\right| = p'_\mu(|x|) = \mu\left(e^{\mu|x|} - e^{-\mu|x|}\right)$ and since $|x - y| \le \delta$ we have that $|x| = |y| + z$ for some $z \in (-\delta, \delta)$. Using that $p'(|x|)$ is monotonic in $|x|$ we then have

$$\begin{aligned}
|p'_\mu(x)| &= p'_\mu(|x|) = p'_\mu(|y| + z) \le p'_\mu(|y| + \delta) \\
&= \mu\left(e^{\mu|y|+\mu\delta} - e^{-\mu|y|-\mu\delta}\right) = e^{\mu\delta}p'(|y|) + \mu\left(e^{\mu\delta-\mu|y|} - e^{-\mu|y|-\mu\delta}\right) \\
&\le e^{\mu\delta}\left|p'(y)\right| + \mu e^{\mu\delta}.
\end{aligned} \tag{6.24}$$

By symmetry (i.e. replacing x and y) this implies that

$$|p'_\mu(x)| \ge e^{-\mu\delta}|p'(y)| - \mu \tag{6.25}$$

Since U is symmetric this implies that for all $i \in [m]$ we have $\mathrm{sign}(\nabla\Phi_\mu(\boldsymbol{y})^\flat)_i = \nabla\Phi_\mu(\boldsymbol{y})_i = \mathrm{sign}(y_i)$. Therefore, if for all $i \in [n]$ we have $\mathrm{sign}(x_i) = \mathrm{sign}(y_i)$, by (6.24), we see that

$$\begin{aligned}
\left\langle \nabla\Phi_\mu(\boldsymbol{x}), \nabla\Phi_\mu(\boldsymbol{y})^\flat \right\rangle &= \sum_i p'_\mu(x_i)\nabla\Phi_\mu(\boldsymbol{y})^\flat_i \\
&\le \sum_i \left(e^{\mu\delta}p'_\mu(y_i) + \mu e^{\mu\delta}\right)\nabla\Phi_\mu(\boldsymbol{y})^\flat_i \\
&\le e^{\mu\delta}\left\langle \nabla\Phi_\mu(\boldsymbol{y}), \nabla\Phi_\mu(\boldsymbol{y})^\flat \right\rangle + \mu e^{\mu\delta}\|\nabla\Phi_\mu(\boldsymbol{y})^\flat\|_1 \\
&= e^{\mu\delta}\|\nabla\Phi_\mu(\boldsymbol{y})\|_U + \mu e^{\mu\delta}\|\nabla\Phi_\mu(\boldsymbol{y})^\flat\|_1.
\end{aligned}$$

Similarly, using (6.25), we have $e^{-\mu\delta}\|\nabla\Phi_\mu(\boldsymbol{y})\|_U - \mu\|\nabla\Phi_\mu(\boldsymbol{y})^\flat\|_1 \le \left\langle \nabla\Phi_\mu(\boldsymbol{x}), \nabla\Phi_\mu(\boldsymbol{y})^\flat \right\rangle$ and hence

(6.22) holds. On the other hand if $\operatorname{sign}(x_i) \neq \operatorname{sign}(y_i)$ then we know that $|x_i| \leq \delta$ and consequently $|p'_\mu(x_i)| \leq \mu(e^{\mu\delta} - e^{-\mu\delta}) \leq \frac{\mu}{2}$ since $\delta \leq \frac{1}{5\mu}$. Thus, we have

$$e^{-\mu\delta}\left|p'_\mu(y_i)\right| - \mu \leq -\frac{\mu}{2} \leq \operatorname{sign}(y_i)\, p'_\mu(x_i) \leq 0 \leq e^{\mu\delta}\left|p'_\mu(y_i)\right| + \mu e^{\mu\delta}.$$

Taking inner product on both sides with $\nabla\Phi_\mu(\boldsymbol{y})_i^\flat$ and using definition of $\|\cdot\|_U$ and $\cdot^\flat$, we get (6.22). Thus, (6.22) holds in general.

Finally we note that since U is contained in a ℓ_∞ ball of radius R, we have $\|\boldsymbol{y}^\flat\|_1 \leq mR$ for all $\boldsymbol{y}$. Using this fact, (6.22), and the definition of $\|\cdot\|_U$, we obtain

$$e^{-\mu\delta}\|\nabla\Phi_\mu(\boldsymbol{y})\|_U - \mu mR \leq \left\langle \nabla\Phi_\mu(\boldsymbol{x}), \nabla\Phi_\mu(\boldsymbol{y})^\flat \right\rangle \leq \|\nabla\Phi_\mu(\boldsymbol{x})\|_U$$

where the last line comes from the fact $\nabla\Phi_\mu(\boldsymbol{y})^\flat \in U$ and the definition of $\|\cdot\|_U$. By symmetry (6.23) follows. □

Using Lemma 6.6.7 we prove Theorem 6.6.6.

Theorem 6.6.6. Let $\|\boldsymbol{x}\|_{U^{(k)}} = \max_{\boldsymbol{y}\in U^{(k)}} \langle \boldsymbol{x}, \boldsymbol{y}\rangle$ and $\boldsymbol{x}^{\flat(k)} = \arg\max_{\boldsymbol{y}\in U^{(k)}} \langle \boldsymbol{x}, \boldsymbol{y}\rangle$. Since $U^{(k)}$ is symmetric, we know that $\boldsymbol{\Delta}^{(k)} = -(1+\varepsilon)\left(\nabla\Phi_\mu(\boldsymbol{z}^{(k)})\right)^{\flat(k)}$ and therefore by applying the mean value theorem twice we have that

$$\begin{aligned}\Phi_\mu(\boldsymbol{x}^{(k+1)}) &= \Phi_\mu(\boldsymbol{y}^{(k)}) + \left\langle \nabla\Phi_\mu(\boldsymbol{z}), \boldsymbol{x}^{(k+1)} - \boldsymbol{y}^{(k)} \right\rangle \\ &= \Phi_\mu(\boldsymbol{x}^{(k)}) + \left\langle \nabla\Phi_\mu(\boldsymbol{y}), \boldsymbol{y}^{(k)} - \boldsymbol{x}^{(k)} \right\rangle + \left\langle \nabla\Phi_\mu(\boldsymbol{z}), \boldsymbol{x}^{(k+1)} - \boldsymbol{y}^{(k)} \right\rangle\end{aligned}$$

for some $\boldsymbol{y}$ between $\boldsymbol{y}^{(k)}$ and $\boldsymbol{x}^{(k)}$ and some $\boldsymbol{z}$ between $\boldsymbol{x}^{(k+1)}$ and $\boldsymbol{y}^{(k)}$. Now, using that $\boldsymbol{y}^{(k)} - \boldsymbol{x}^{(k)} \in U^{(k)}$ and that $\boldsymbol{x}^{(k+1)} - \boldsymbol{y}^{(k)} = \boldsymbol{\Delta}^{(k)}$ we have

$$\Phi_\mu(\boldsymbol{x}^{(k+1)}) \leq \Phi_\mu(\boldsymbol{x}^{(k)}) + \|\nabla\Phi_\mu(\boldsymbol{y})\|_{U^{(k)}} - (1+\varepsilon)\left\langle \nabla\Phi_\mu(\boldsymbol{z}), \left(\nabla\Phi_\mu(\boldsymbol{z}^{(k)})\right)^{\flat(k)} \right\rangle. \tag{6.26}$$

Since U^k is contained within the ℓ_∞ ball of radius R_k Lemma 6.6.7 shows that

$$\|\nabla\Phi_\mu(\boldsymbol{y})\|_{U^{(k)}} \leq e^{\mu R_k}\|\nabla\Phi_\mu(\boldsymbol{x}^{(k)})\|_{U^{(k)}} + m\mu R_k e^{\mu R_k}. \tag{6.27}$$

Furthermore, since $\varepsilon < \frac{1}{5}$ and $R_k \leq R$, by triangle inequality we have $\|\boldsymbol{z} - \boldsymbol{z}^{(k)}\|_\infty \leq (1+\varepsilon)R_k + R \leq 3R$ and $\|\boldsymbol{z}^{(k)} - \boldsymbol{x}^{(k)}\|_\infty \leq 2R$. Therefore, applying Lemma 6.6.7 twice yields that

$$\begin{aligned}\left\langle \nabla\Phi_\mu(\boldsymbol{z}), \nabla\Phi_\mu(\boldsymbol{z}^{(k)})^{\flat(k)} \right\rangle &\geq e^{-3\mu R}\|\nabla\Phi_\mu(\boldsymbol{z}^{(k)})\|_{U^{(k)}} - \mu m R_k \\ &\geq e^{-5\mu R}\|\nabla\Phi_\mu(\boldsymbol{x}^{(k)})\|_{U^{(k)}} - 2\mu m R_k.\end{aligned} \tag{6.28}$$

Combining (6.26), (6.27), and (6.28) then yields that

$$\Phi_\mu(\boldsymbol{x}^{(k+1)}) \leq \Phi_\mu(\boldsymbol{x}^{(k)}) - \left((1+\varepsilon)e^{-5\mu R} - e^{\mu R}\right)\|\nabla\Phi_\mu(\boldsymbol{x}^{(k)})\|_{U^{(k)}} + m\mu R_k e^{\mu R} + 2(1+\varepsilon)m\mu R_k.$$

Since we chose $\mu = \frac{\varepsilon}{12R}$, we have $1+\varepsilon \leq \frac{\varepsilon}{2} + (1+6\mu R) \leq \frac{\varepsilon}{2}e^{5\mu R} + e^{6\mu R}$. Hence, we have $(1+\varepsilon)e^{-5\mu R} - e^{\mu R} \leq \frac{\varepsilon}{2}$. Also, since $0 < \varepsilon < \frac{1}{5}$ we have

$$m\mu R_k e^{\mu R} + 2(1+\varepsilon)m\mu R_k \leq \left(e^{\mu R} + 2(1+\varepsilon)\right)m\mu R_k \leq \varepsilon m\frac{7R_k}{24R}.$$

Thus, we have

$$\Phi_\mu(\boldsymbol{x}^{(k+1)}) \leq \Phi_\mu(\boldsymbol{x}^{(k)}) - \frac{\varepsilon}{2}\|\nabla\Phi_\mu(\boldsymbol{x}^{(k)})\|_{U^{(k)}} + \varepsilon m\frac{7R_k}{24R}.$$

Using Lemma 6.6.7 and the fact that U_k contains a ℓ_∞ ball of radius r_k, we have

$$\|\nabla\Phi_\mu(\boldsymbol{x}^{(k)})\|_{U^{(k)}} \geq r_k\|\nabla\Phi_\mu(\boldsymbol{x}^{(k)})\|_1 \geq \frac{\varepsilon r_k}{12R}\left(\Phi_\mu(\boldsymbol{x}^{(k)}) - 2m\right).$$

Therefore, we have that

$$\begin{aligned}\Phi_\mu(\boldsymbol{x}^{(k+1)}) &\leq \left(1 - \frac{\varepsilon^2 r_k}{24R}\right)\Phi_\mu(\boldsymbol{x}^{(k)}) + \frac{\varepsilon r_k}{12R}m + \varepsilon m\frac{7R_k}{24R} \\ &\leq \left(1 - \frac{\varepsilon^2 r_k}{24R}\right)\Phi_\mu(\boldsymbol{x}^{(k)}) + \varepsilon m\frac{R_k}{2R}.\end{aligned}$$

Hence, if $\Phi_\mu(\boldsymbol{x}^{(k)}) \leq \frac{12m\tau}{\varepsilon}$, we have $\Phi_\mu(\boldsymbol{x}^{(k+1)}) \leq \frac{12m\tau}{\varepsilon}$. Since$\Phi_\mu(\boldsymbol{x}^{(0)}) \leq \frac{12m\tau}{\varepsilon}$ by assumption we have by induction that $\Phi_\mu(\boldsymbol{x}^{(k)}) \leq \frac{12m\tau}{\varepsilon}$ for all k. The necessary bound on $\|\boldsymbol{x}^{(k)}\|_\infty$ then follows immediately from Lemma 6.6.7. □

6.6.5 Centering

Here, we show how to use the results of the previous subsection to perform weighted path following given access only to a multiplicative approximation of the weight function. To apply the results of the previous subsection, we can think updating weight is playing this game, we want to make sure the error between $\boldsymbol{w}$ and $\boldsymbol{g}(\boldsymbol{x})$ is close to 0 while the adversary control the next point $\boldsymbol{g}(\boldsymbol{x})$ and the noise in the approximate $\boldsymbol{g}(\boldsymbol{x})$. Theorem 6.6.6 shows that we can control the error to be small in ℓ_∞ if we can approximate $\boldsymbol{g}(\boldsymbol{x})$ with small ℓ_∞ error.

Formally, we will measure its distance from the optimal weights in log scale by

$$\boldsymbol{\Psi}(\boldsymbol{x}, \boldsymbol{w}) \stackrel{\text{def}}{=} \log(\boldsymbol{g}(\boldsymbol{x})) - \log(\boldsymbol{w}). \tag{6.29}$$

Our goal will be to keep $\|\boldsymbol{\Psi}(\boldsymbol{x}, \boldsymbol{w})\|_{\boldsymbol{w}+\infty} \leq K$ for some error K that is just small enough to not impair our ability to decrease δ_t linearly and not to impair our ability to approximate $\boldsymbol{g}$. We will attempt to do this without moving $\boldsymbol{w}$ too much in $\|\cdot\|_{\boldsymbol{w}+\infty}$.

Algorithm 8: $(\boldsymbol{x}^{(\text{new})}, \boldsymbol{w}^{(\text{new})}) = \texttt{centeringInexact}(\boldsymbol{x}, \boldsymbol{w}, K)$

Let $c_k = \frac{1}{1-c_\delta c_\gamma}$,$R = \frac{K}{48c_k\log(400m)}$, $\delta_t = \delta_t(\boldsymbol{x}, \boldsymbol{w})$ and $\varepsilon = \frac{1}{2c_k}$.

Set $\boldsymbol{x}^{(\text{new})} = \boldsymbol{x} - \frac{1}{\sqrt{\vec{\phi}''(\boldsymbol{x})}}\boldsymbol{P}_{\boldsymbol{x},\boldsymbol{w}}\left(\frac{t\boldsymbol{c}-\boldsymbol{w}\vec{\phi}'(\boldsymbol{x})}{\boldsymbol{w}\sqrt{\vec{\phi}''(\boldsymbol{x})}}\right)$.

Let $U = \{\boldsymbol{x} \in \mathbb{R}^m \mid \|\boldsymbol{x}\|_{\boldsymbol{w}+\infty} \leq \left(1 - \frac{7}{8c_k}\right)\delta_t\}$.

Find $\boldsymbol{z}$ such that $\|\boldsymbol{z} - \log(\boldsymbol{g}(\boldsymbol{x}^{(\text{new})}))\|_\infty \leq R$.

Set $\boldsymbol{w}^{(\text{new})} = \exp\left(\log(\boldsymbol{w}) + (1+\varepsilon)\arg\min_{\boldsymbol{u}\in U}\left\langle\nabla\Phi_{\frac{\varepsilon}{12R}}(\boldsymbol{z} - \log(\boldsymbol{w})), \boldsymbol{u}\right\rangle\right)$.

Output $(\boldsymbol{x}^{(\text{new})}, \boldsymbol{w}^{(\text{new})})$.

The minimization problem in last step in `centeringInexact` is simply a projection onto the convex set U and it can be done in $\tilde{O}(1)$ depth and $\tilde{O}(m)$ work. See section 6.12 for details.

Theorem 6.6.8. *Assume that* $24m^{1/4} \geq c_k \stackrel{\text{def}}{=} \frac{1}{1-c_\delta c_\gamma} \geq 5$, $1 \leq C_{norm} \leq 2c_k$ *and* $K \leq \frac{1}{20c_k}$. *Let*

$\boldsymbol{\Psi}(\boldsymbol{x},\boldsymbol{w}) \stackrel{\text{def}}{=} \log(\boldsymbol{g}(\boldsymbol{x})) - \log(\boldsymbol{w})$. *Suppose that*

$$\delta \stackrel{\text{def}}{=} \delta_t(\boldsymbol{x},\boldsymbol{w}) \leq \frac{K}{48c_k \log(400m)} \quad \text{and} \quad \Phi_\mu(\boldsymbol{\Psi}(\boldsymbol{x},\boldsymbol{w})) \leq (400m)^2$$

where $\mu = \frac{\varepsilon}{12R} = 2\log(400m)/K$. *Let* $(\boldsymbol{x}^{(new)},\boldsymbol{w}^{(new)}) = \texttt{centeringInexact}(\boldsymbol{x},\boldsymbol{w},K)$, *then*

$$\delta_t(\boldsymbol{x}^{(new)},\boldsymbol{w}^{(new)}) \leq \left(1-\frac{1}{4c_k}\right)\delta \quad \text{and} \quad \Phi_\mu(\boldsymbol{\Psi}(\boldsymbol{x}^{(new)},\boldsymbol{w}^{(new)})) \leq (400m)^2.$$

Also, we have $\|\log(\boldsymbol{g}(\boldsymbol{x}^{(new)})) - \log(\boldsymbol{w})\|_\infty \leq K$.

Proof. By Lemma 6.6.3, inequality (6.16), $c_\delta c_\gamma \leq 1$ and $c_\gamma \leq \frac{5}{4}$ (see Def 6.5.4), we have

$$\begin{aligned}\left\|\log\left(\boldsymbol{g}(\boldsymbol{x}^{(\text{new})})\right) - \log\left(\boldsymbol{g}(\boldsymbol{x})\right)\right\|_{\boldsymbol{g}(\boldsymbol{x})+\infty} &\leq c_\delta c_\gamma \delta(1+4c_\gamma\delta)\\ &\leq c_\delta c_\gamma \delta + 5\delta^2\\ &\leq \left(1-\frac{15}{16c_k}\right)\delta.\end{aligned}$$

Using $K \leq \frac{1}{20c_k}$, we have

$$\left\|\frac{\boldsymbol{w}-\boldsymbol{g}(\boldsymbol{x})}{\boldsymbol{g}(\boldsymbol{x})}\right\|_\infty \leq \|\log(\boldsymbol{w}) - \log(\boldsymbol{g}(\boldsymbol{x}))\|_\infty + \|\log(\boldsymbol{w}) - \log(\boldsymbol{g}(\boldsymbol{x}))\|_\infty^2 \leq \frac{21}{20}K \leq \frac{1}{16c_k}.$$

Hence, we have

$$\begin{aligned}\left\|\log\left(\boldsymbol{g}(\boldsymbol{x}^{(\text{new})})\right) - \log\left(\boldsymbol{g}(\boldsymbol{x})\right)\right\|_{\boldsymbol{w}+\infty} &\leq \left(1+\frac{1}{16c_k}\right)\left(1-\frac{15}{16c_k}\right)\delta\\ &\leq \left(1-\frac{7}{8c_k}\right)\delta.\end{aligned}$$

Therefore, we know that for the Newton step, we have $\boldsymbol{\Psi}(\boldsymbol{x}^{(\text{new})},\boldsymbol{w}) - \boldsymbol{\Psi}(\boldsymbol{x},\boldsymbol{w}) \in U$ where U is the symmetric convex set given by

$$U \stackrel{\text{def}}{=} \{\boldsymbol{x} \in \mathbb{R}^n \mid \|\boldsymbol{x}\|_{\boldsymbol{w}+\infty} \leq C\}$$

where $C = \left(1-\frac{7}{8c_k}\right)\delta$. Note that from our assumption on δ, we have

$$C \leq \delta \leq \frac{K}{48c_k\log(400m)} = R.$$

It ensures that U are contained in some ℓ_∞ ball of radius R. Therefore, we can play the chasing 0 game on $\boldsymbol{\Psi}(\boldsymbol{x},\boldsymbol{w})$ attempting to maintain the invariant that $\|\boldsymbol{\Psi}(\boldsymbol{x},\boldsymbol{w})\|_\infty \leq K$ without taking steps that are more than $1+\varepsilon$ times the size of U where we pick $\varepsilon = \frac{1}{2c_k}$ so to not interfere with our ability to decrease δ_t linearly.

However, to do this with the chasing 0 game, we need to ensure that R satisfying the following

$$\frac{12R}{\varepsilon}\log\left(\frac{12m\tau}{\varepsilon}\right) \leq K$$

where here τ is as defined in Theorem 6.6.6.

To bound τ, we need to lower bound the radius of ℓ_∞ ball it contains. Since by assumption

$\|\boldsymbol{g}(\boldsymbol{x})\|_\infty \leq 2$ and $\|\boldsymbol{\Psi}(\boldsymbol{x},\boldsymbol{w})\|_\infty \leq \frac{1}{8}$, we have that $\|\boldsymbol{w}\|_\infty \leq 3$. Hence, we have

$$\forall u \in \mathbb{R}^m \quad : \quad \|\boldsymbol{u}\|_\infty^2 \geq \frac{1}{3m}\|\boldsymbol{u}\|_{\boldsymbol{w}}^2.$$

Consequently, if $\|\boldsymbol{u}\|_\infty \leq \frac{\delta}{5C_{\text{norm}}\sqrt{m}}$, then $\boldsymbol{u} \in U$. So, we have that U contains a box of radius $\frac{\delta}{5C_{\text{norm}}\sqrt{m}}$ and since U is contained in a box of radius δ, we have that

$$\tau \leq 5C_{\text{norm}}\sqrt{m} \leq 10c_k\sqrt{m}.$$

Using $c_k \leq 24m^{1/4}$, we have

$$\begin{aligned}\frac{12R}{\varepsilon}\log\left(\frac{12m\tau}{\varepsilon}\right) &\leq 24c_kR\log\left(240m^{3/2}c_k^2\right)\\ &\leq 48c_kR\log\left(400m\right) = K.\end{aligned}$$

and

$$\frac{12m\tau}{\varepsilon} \leq 240m^{3/2}c_k^2 \leq (400m)^2.$$

This proves that we meet the conditions of Theorem 6.6.6. Consequently, $\|\boldsymbol{\Psi}(\boldsymbol{x}^{(\text{new})},\boldsymbol{w}^{(\text{new})})\|_\infty \leq K$ and $\Phi_\alpha(\boldsymbol{\Psi}(\boldsymbol{x}^{(\text{new})},\boldsymbol{w}^{(\text{new})})) \leq (400m)^2$.

Since $K \leq \frac{1}{4}$, Lemma 6.6.2 shows that

$$\delta_t(\boldsymbol{x}^{(\text{new})},\boldsymbol{w}) \leq 4\left(\delta_t(\boldsymbol{x},\boldsymbol{w})\right)^2.$$

Our choice of $\boldsymbol{w}^{(\text{new})}$ shows that

$$\begin{aligned}\left\|\log(\boldsymbol{w}) - \log(\boldsymbol{w}^{(\text{new})})\right\|_{\boldsymbol{w}+\infty} &\leq \left(1+\frac{1}{2c_k}\right)\left(1-\frac{7}{8c_k}\right)\delta\\ &\leq \left(1-\frac{3}{8c_k}\right)\delta.\end{aligned}$$

Using $\delta \leq \frac{1}{80c_k}$, the Lemma 6.6.4 shows that

$$\begin{aligned}\delta_t(\boldsymbol{x}^{(\text{new})},\boldsymbol{w}^{(\text{new})}) &\leq \left(1+4\left(1-\frac{3}{8c_k}\right)\delta\right)\left(\delta_t(\boldsymbol{x}^{(\text{new})},\boldsymbol{w}) + \left(1-\frac{3}{8c_k}\right)\delta\right)\\ &\leq (1+4\delta)\left(4\delta^2 + \left(1-\frac{3}{8c_k}\right)\delta\right)\\ &\leq \left(1-\frac{3}{8c_k}\right)\delta + 4\delta^2 + 16\delta^3 + 4\delta^2\\ &\leq \left(1-\frac{1}{4c_k}\right)\delta.\end{aligned}$$

□

6.7 Weight Function

In this section we present the weight function that we use to achieve our $\widetilde{O}(\sqrt{\text{rank}(\boldsymbol{A})}\log(U/\varepsilon))$ iteration linear program solver. This weight function is motivated from the barrier introduced in

Subsection 6.3.5. We define the weight function $\boldsymbol{g} : \Omega \to \mathbb{R}^m_{>0}$ for all $\boldsymbol{x} \in \mathbb{R}^m_{>0}$ as follows

$$\boldsymbol{g}(\boldsymbol{x}) \stackrel{\text{def}}{=} \arg\min_{\boldsymbol{w} \in \mathbb{R}^m_{>0}} \hat{f}(\boldsymbol{x}, \boldsymbol{w}) \quad \text{where} \quad \hat{f}(\boldsymbol{x}, \boldsymbol{w}) \stackrel{\text{def}}{=} \mathbf{1}^T \boldsymbol{w} + \frac{1}{\alpha} \log\det\left(\boldsymbol{A}_x^T \boldsymbol{W}^{-\alpha} \boldsymbol{A}_x\right) - \beta \sum_{i \in [m]} \log w_i. \quad (6.30)$$

where here and in the remainder of the subsection we let $\boldsymbol{A}_x \stackrel{\text{def}}{=} (\boldsymbol{\Phi}''(\boldsymbol{x}))^{-1/2} \boldsymbol{A}$ and the parameters α, β are chosen later such that the following hold

$$\alpha \in [1, 2), \ \beta \in (0, 1), \text{ and } \beta^{1-\alpha} \leq 2. \quad (6.31)$$

Here we choose β small and α just slightly larger than 1.

We start by computing the gradient and Hessian of $\hat{f}(\boldsymbol{x}, \boldsymbol{w})$ with respect to $\boldsymbol{w}$.

Lemma 6.7.1. *For all $\boldsymbol{x} \in \Omega$ and $\boldsymbol{w} \in \mathbb{R}^m_{>0}$, we have*

$$\nabla_w \hat{f}(\boldsymbol{x}, \boldsymbol{w}) = \left(\boldsymbol{I} - \boldsymbol{\Sigma}\boldsymbol{W}^{-1} - \beta \boldsymbol{W}^{-1}\right) \mathbf{1} \quad \text{and} \quad \nabla^2_{ww} \hat{f}(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{W}^{-1} \left(\boldsymbol{\Sigma} + \beta \boldsymbol{I} + \alpha \boldsymbol{\Lambda}\right) \boldsymbol{W}^{-1}$$

where $\boldsymbol{\Sigma} \stackrel{\text{def}}{=} \boldsymbol{\Sigma}_{\boldsymbol{A}_x}(\boldsymbol{w}^{-\alpha})$ and $\boldsymbol{\Lambda} \stackrel{\text{def}}{=} \boldsymbol{\Lambda}_{\boldsymbol{A}_x}(\boldsymbol{w}^{-\alpha})$.

Proof. Using Lemma 6.11.2 and the chain rule we compute the gradient of $\nabla_w \hat{f}(\boldsymbol{x}, \boldsymbol{w})$ as follows

$$\begin{aligned} \nabla_w \hat{f}(\boldsymbol{x}, \boldsymbol{w}) &= \mathbf{1} + \frac{1}{\alpha} \boldsymbol{\Sigma} \boldsymbol{W}^{\alpha} \left(-\alpha \boldsymbol{W}^{-\alpha-1}\right) - \beta \boldsymbol{W}^{-1} \mathbf{1} \\ &= \left(\boldsymbol{I} - \boldsymbol{\Sigma}\boldsymbol{W}^{-1} - \beta \boldsymbol{W}^{-1}\right) \mathbf{1}. \end{aligned}$$

Next, using Lemma 6.11.2 and chain rule, we compute the following for all $i, j \in [m]$

$$\begin{aligned} \frac{\partial (\nabla_w \hat{f}(\boldsymbol{x}, \boldsymbol{w}))_i}{\partial \boldsymbol{w}_j} &= -\frac{\boldsymbol{w}_i \boldsymbol{\Lambda}_{ij} \boldsymbol{w}_j^{\alpha} \left(-\alpha \boldsymbol{w}_j^{-\alpha-1}\right) - \boldsymbol{\Sigma}_{ij} \mathbf{1}_{i=j}}{\boldsymbol{w}_i^2} + \beta \mathbf{1}_{i=j} \left\{\boldsymbol{w}_i^{-2}\right\} \\ &= \frac{\boldsymbol{\Sigma}_{ij}}{\boldsymbol{w}_i \boldsymbol{w}_j} + \alpha \frac{\boldsymbol{\Lambda}_{ij}}{\boldsymbol{w}_i \boldsymbol{w}_j} + \frac{\beta \mathbf{1}_{i=j}}{\boldsymbol{w}_i^2} \quad . \qquad \text{(Using that } \boldsymbol{\Sigma} \text{ is diagonal)} \end{aligned}$$

Consequently, $\nabla^2_{ww} \hat{f}(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{W}^{-1} \left(\boldsymbol{\Sigma} + \beta \boldsymbol{I} + \alpha \boldsymbol{\Lambda}\right) \boldsymbol{W}^{-1}$ as desired. □

Lemma 6.7.2. *For all $\boldsymbol{x} \in \Omega$, the weight function $\boldsymbol{g}(\boldsymbol{x})$ is a well defined with*

$$\beta \leq g_i(\boldsymbol{s}) \leq 1 + \beta \quad \text{and} \quad \|\boldsymbol{g}(\boldsymbol{x})\|_1 = \operatorname{rank}(\boldsymbol{A}) + \beta \cdot m.$$

Furthermore, for all $\boldsymbol{x} \in \Omega$, the weight function obeys the following equations

$$\boldsymbol{G}(\boldsymbol{x}) = (\boldsymbol{\Sigma} + \beta \boldsymbol{I}) \mathbf{1} \quad , \text{ and } \quad \boldsymbol{G}'(\boldsymbol{x}) = -\boldsymbol{G}(\boldsymbol{x}) \left(\boldsymbol{G}(\boldsymbol{x}) + \alpha \boldsymbol{\Lambda}\right)^{-1} \boldsymbol{\Lambda} \left(\boldsymbol{\Phi}''(\boldsymbol{x})\right)^{-1} \boldsymbol{\Phi}'''(\boldsymbol{x})$$

where $\boldsymbol{\Sigma} \stackrel{\text{def}}{=} \boldsymbol{\Sigma}_{\boldsymbol{A}_x}(\boldsymbol{g}^{-\alpha}(\boldsymbol{x}))$, $\boldsymbol{\Lambda} \stackrel{\text{def}}{=} \boldsymbol{\Lambda}_{\boldsymbol{A}_x}(\boldsymbol{g}^{-\alpha}(\boldsymbol{x}))$, and $\boldsymbol{G}'(\boldsymbol{x})$ is the Jacobian matrix of $\boldsymbol{g}$ at $\boldsymbol{x}$.

Proof. By Lemma 6.11.2 we have that $\boldsymbol{\Sigma} \succeq \boldsymbol{\Lambda} \succeq \mathbf{0}$. Therefore, by Lemma 6.7.1, we have that $\nabla^2_{ww} \hat{f}(\boldsymbol{x}, \boldsymbol{w}) \succeq \beta \boldsymbol{W}^{-2}$ and $\hat{f}(\boldsymbol{x}, \boldsymbol{w})$ is convex. Using the formula for the gradient in Lemma 6.7.1, we see that that for all $i \in [m]$ it is the case that

$$\left[\nabla_w \hat{f}(\boldsymbol{x}, \boldsymbol{w})\right]_i = \frac{1}{w_i} \left(w_i - \boldsymbol{\Sigma}_{ii} - \beta\right).$$

Using that $0 \leq \sigma_i \leq 1$ for all i by Lemma 6.11.2 and $\beta \in (0, 1)$ by (6.31), we see that if $\boldsymbol{w}_i \in (0, \beta)$ then $\left[\nabla_w \hat{f}(\boldsymbol{x}, \boldsymbol{w})\right]_i$ is strictly negative and if $\boldsymbol{w}_i \in (1 + \beta, \infty)$ then $\left[\nabla_w \hat{f}(\boldsymbol{x}, \boldsymbol{w})\right]_i$ is strictly positive.

Therefore, for any $\boldsymbol{x} \in \Omega$, the $\boldsymbol{w}$ that minimizes this convex function $\hat{f}(\boldsymbol{x}, \boldsymbol{w})$ lies between the box between β to $1+\beta$. Since $\hat{f}$ is strongly convex in this region, the minimizer is unique.

The formula for $\boldsymbol{G}(\boldsymbol{x})$ follows by setting $\nabla_w \hat{f}(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{0}$ and the size of $g(\boldsymbol{x})$ follows from the fact that $\|\boldsymbol{\sigma}\|_1 = \operatorname{tr}\left(\boldsymbol{P}_{\boldsymbol{A}_x}\left(\boldsymbol{g}^{-\alpha}(\boldsymbol{x})\right)\right)$. Since $\boldsymbol{P}_{\boldsymbol{A}_x}\left(\boldsymbol{g}^{-\alpha}(\boldsymbol{x})\right)$ is a projection onto the image of $\boldsymbol{G}(\boldsymbol{x})^{-\alpha/2}\boldsymbol{A}_x$ and since $\boldsymbol{g}(\boldsymbol{x}) > \boldsymbol{0}$ and $\vec{\phi}''(\boldsymbol{x}) > \boldsymbol{0}$, we have that the dimension of the image of $\boldsymbol{G}(\boldsymbol{x})^{-\alpha/2}\boldsymbol{A}_x$ is the rank of $\boldsymbol{A}$. Hence, we have that $\|\boldsymbol{g}(\boldsymbol{x})\|_1 = \operatorname{rank}(\boldsymbol{A}) + \beta \cdot m$.

By Lemma 6.11.2 and chain rule, we get the following for all $i, j \in [m]$

$$\frac{\partial(\nabla_{\boldsymbol{w}} \hat{f}(\boldsymbol{x}, \boldsymbol{w}))_i}{\partial \boldsymbol{x}_j} = -\boldsymbol{w}_i^{-1}\boldsymbol{\Lambda}_{ij}\vec{\phi}_j''(\boldsymbol{x})\left(-(\vec{\phi}_j''(\boldsymbol{x}))^{-2}\vec{\phi}_j'''(\boldsymbol{x})\right) = \boldsymbol{w}_i^{-1}\boldsymbol{\Lambda}_{ij}(\vec{\phi}_j''(\boldsymbol{x}))^{-1}\vec{\phi}_j'''(\boldsymbol{x}).$$

Consequently, $\boldsymbol{J}_{\boldsymbol{x}}(\nabla_{\boldsymbol{w}} \hat{f}(\boldsymbol{x}, \boldsymbol{w})) = \boldsymbol{W}^{-1}\boldsymbol{\Lambda}\left(\boldsymbol{\Phi}''(\boldsymbol{x})\right)^{-1}\boldsymbol{\Phi}'''(\boldsymbol{x})$ where $\boldsymbol{J}_{\boldsymbol{x}}$ denotes the Jacobian matrix of the function $\nabla_{\boldsymbol{w}} \hat{f}(\boldsymbol{x}, \boldsymbol{w})$ with respect to $\boldsymbol{x}$. Since we have already know that $\boldsymbol{J}_{\boldsymbol{w}}(\nabla_{\boldsymbol{w}} \hat{f}(\boldsymbol{x}, \boldsymbol{w})) = \nabla^2_{\boldsymbol{ww}} f_t(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{W}^{-1}(\boldsymbol{\Sigma} + \beta\boldsymbol{I} + \alpha\boldsymbol{\Lambda})\boldsymbol{W}^{-1}$ is positive definite (and hence invertible), by applying the implicit function theorem (Lemma 2.3.9) to the specification of $\boldsymbol{g}(\boldsymbol{x})$ as the solution to $\nabla_{\boldsymbol{w}} \hat{f}(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{0}$, we have

$$\boldsymbol{G}'(\boldsymbol{x}) = -\left(\boldsymbol{J}_{\boldsymbol{w}}(\nabla_w \hat{f}(\boldsymbol{x}, \boldsymbol{w}))\right)^{-1}\left(\boldsymbol{J}_{\boldsymbol{x}}(\nabla_{\boldsymbol{w}} \hat{f}(\boldsymbol{x}, \boldsymbol{w}))\right) = -\boldsymbol{G}(\boldsymbol{x})\left(\boldsymbol{G}(\boldsymbol{x}) + \alpha\boldsymbol{\Lambda}\right)^{-1}\boldsymbol{\Lambda}\left(\boldsymbol{\Phi}''(\boldsymbol{x})\right)^{-1}\boldsymbol{\Phi}'''(\boldsymbol{x}).$$

□

Now we show the step consistency of $\boldsymbol{g}$.

Lemma 6.7.3 (Step Consistency). *For all $\boldsymbol{x} \in \Omega$ and $\boldsymbol{y} \in \mathbb{R}^m$, and*

$$\boldsymbol{B} \stackrel{\text{def}}{=} \boldsymbol{G}(\boldsymbol{x})^{-1}\boldsymbol{G}'(\boldsymbol{x})(\boldsymbol{\phi}(\boldsymbol{x})'')^{-1/2},$$

we have

$$\|\boldsymbol{B}\boldsymbol{y}\|_{\boldsymbol{G}(\boldsymbol{x})} \le \frac{2}{1+\alpha}\|\boldsymbol{y}\|_{\boldsymbol{G}(\boldsymbol{x})} \quad \text{and} \quad \|\boldsymbol{B}\boldsymbol{y}\|_\infty \le \frac{2}{1+\alpha}\left(\|\boldsymbol{y}\|_\infty + \frac{1+2\alpha}{1+\alpha}\|\boldsymbol{y}\|_{\boldsymbol{G}(\boldsymbol{x})}\right).$$

Therefore

$$\|\boldsymbol{B}\|_{\boldsymbol{g}+\infty} \le \frac{2}{1+\alpha}\left(1 + \frac{2}{C_{norm}}\right).$$

Proof. Fix an arbitrary $\boldsymbol{x} \in \Omega$ and let $\boldsymbol{g} \stackrel{\text{def}}{=} \boldsymbol{g}(\boldsymbol{x})$, $\boldsymbol{\sigma} \stackrel{\text{def}}{=} \boldsymbol{\sigma}_{\boldsymbol{A}_x}\left(\boldsymbol{g}^{-\alpha}(\boldsymbol{x})\right)$, $\boldsymbol{\Sigma} \stackrel{\text{def}}{=} \boldsymbol{\Sigma}_{\boldsymbol{A}_x}\left(\boldsymbol{g}^{-\alpha}(\boldsymbol{x})\right)$, $\boldsymbol{P} \stackrel{\text{def}}{=} \boldsymbol{P}_{\boldsymbol{A}_x}\left(\boldsymbol{g}^{-\alpha}(\boldsymbol{x})\right)$, $\boldsymbol{\Lambda} \stackrel{\text{def}}{=} \boldsymbol{\Lambda}_{\boldsymbol{A}_x}\left(\boldsymbol{g}^{-\alpha}(\boldsymbol{x})\right)$. Also, fix an arbitrary $\boldsymbol{y} \in \mathbb{R}^m$ and let $\boldsymbol{z} \stackrel{\text{def}}{=} \boldsymbol{B}\boldsymbol{y}$.

By Lemma 6.7.2, $\boldsymbol{G}' = -\boldsymbol{G}\left(\boldsymbol{G} + \alpha\boldsymbol{\Lambda}\right)^{-1}\boldsymbol{\Lambda}\left(\boldsymbol{\Phi}''\right)^{-1}\boldsymbol{\Phi}'''$ and therefore

$$\begin{aligned}\boldsymbol{B} &= -\boldsymbol{G}^{-1}\left(\boldsymbol{G}\left(\boldsymbol{G} + \alpha\boldsymbol{\Lambda}\right)^{-1}\boldsymbol{\Lambda}\left(\boldsymbol{\Phi}''\right)^{-1}\boldsymbol{\Phi}'''\right)\left(\boldsymbol{\Phi}''\right)^{-1/2} \\ &= \left(\boldsymbol{G} + \alpha\boldsymbol{\Lambda}\right)^{-1}(2\boldsymbol{\Lambda})\,\boldsymbol{Diag}\left(\frac{-\boldsymbol{\phi}'''}{2(\boldsymbol{\phi}'')^{3/2}}\right).\end{aligned}$$

Let $\boldsymbol{C} \stackrel{\text{def}}{=} \left(\boldsymbol{G} + \alpha\boldsymbol{\Lambda}\right)^{-1}(2\boldsymbol{\Lambda})$ and let $\boldsymbol{y}' \stackrel{\text{def}}{=} \boldsymbol{Diag}\left(\frac{-\boldsymbol{\phi}'''}{2(\boldsymbol{\phi}'')^{3/2}}\right)\boldsymbol{y}$. By the self concordance of $\boldsymbol{\phi}$ (6.11), we know that $\|\boldsymbol{y}'\| \le \|\boldsymbol{y}\|$ for both $\|\cdot\|_{\boldsymbol{G}}$ and $\|\cdot\|_\infty$. Since $\boldsymbol{z} = \boldsymbol{B}\boldsymbol{y} = \boldsymbol{C}\boldsymbol{y}'$, it suffices to bound $\|\boldsymbol{C}\boldsymbol{y}'\|$ in terms of $\|\boldsymbol{y}'\|$ for the necessary norms.

Letting $\bar{\boldsymbol{\Lambda}} \stackrel{\text{def}}{=} \boldsymbol{G}^{-1/2}\boldsymbol{\Lambda}\boldsymbol{G}^{-1/2}$, we simplify the equation further and note that

$$\|\boldsymbol{C}\|_{\boldsymbol{G}} = \|\boldsymbol{G}^{1/2}\left(\boldsymbol{G} + \alpha\boldsymbol{\Lambda}\right)^{-1}(2\boldsymbol{\Lambda})\,\boldsymbol{G}^{-1/2}\|_2 = \|\left(\boldsymbol{I} + \alpha\bar{\boldsymbol{\Lambda}}\right)^{-1}\left(2\bar{\boldsymbol{\Lambda}}\right)\|_2.$$

Now, for any eigenvector, $\boldsymbol{v}$, of $\bar{\boldsymbol{\Lambda}}$ with eigenvalue λ, we see that $\boldsymbol{v}$ is an eigenvector of $(\boldsymbol{I}+\alpha\bar{\boldsymbol{\Lambda}})^{-1}(2\bar{\boldsymbol{\Lambda}})$ with eigenvalue $2\lambda/(1+\alpha\lambda)$. Furthermore, since $\mathbf{0} \preceq \bar{\boldsymbol{\Lambda}} \preceq \boldsymbol{I}$, we have that $\|\boldsymbol{C}\|_{\boldsymbol{G}} \leq 2/(1+\alpha)$ and hence $\|\boldsymbol{z}\|_{\boldsymbol{G}} \leq 2(1+\alpha)^{-1}\|\boldsymbol{y}'\|_{\boldsymbol{G}} \leq 2(1+\alpha)^{-1}\|\boldsymbol{y}\|_{\boldsymbol{G}}$ as desired.

To bound $\|\boldsymbol{z}\|_\infty$, we use that $(\boldsymbol{G}+\alpha\boldsymbol{\Lambda})\,\boldsymbol{z} = 2\boldsymbol{\Lambda}\boldsymbol{y}'$, $\boldsymbol{\Lambda} = \boldsymbol{\Sigma} - \boldsymbol{P}^{(2)}$, and $\boldsymbol{G} = \boldsymbol{\Sigma} + \beta\boldsymbol{I}$ to derive

$$(1+\alpha)\,\boldsymbol{\Sigma}\boldsymbol{z} + \beta\boldsymbol{z} - \alpha\boldsymbol{P}^{(2)}\boldsymbol{z} = 2\boldsymbol{\Sigma}\boldsymbol{y}' - 2\boldsymbol{P}^{(2)}\boldsymbol{y}'.$$

Looking at the i^{th} coordinate of both sides and using that $\boldsymbol{\sigma}_i \geq 0$, we have

$$\begin{aligned}
&((1+\alpha)\boldsymbol{\sigma}_i + \beta)\,|\boldsymbol{z}_i| \\
\leq&\alpha\left|[\boldsymbol{P}^{(2)}\boldsymbol{z}]_i\right| + 2\boldsymbol{\sigma}_i\|\boldsymbol{y}'\|_\infty + 2\left|[\boldsymbol{P}^{(2)}\boldsymbol{y}']_i\right| \\
\leq&\alpha\boldsymbol{\sigma}_i\|\boldsymbol{z}\|_{\boldsymbol{\Sigma}} + 2\boldsymbol{\sigma}_i\|\boldsymbol{y}'\|_\infty + 2\boldsymbol{\sigma}_i\|\boldsymbol{y}'\|_{\boldsymbol{\Sigma}} && \text{(Lemma 6.11.2)} \\
\leq&2\boldsymbol{\sigma}_i\|\boldsymbol{y}'\|_\infty + \boldsymbol{\sigma}_i\left(\frac{2\alpha}{1+\alpha}+2\right)\|\boldsymbol{y}'\|_{\boldsymbol{G}} && (\boldsymbol{\Sigma}\preceq\boldsymbol{G}\text{and } \|\boldsymbol{z}\|_{\boldsymbol{G}} \leq 2(1+\alpha)^{-1}\|\boldsymbol{y}'\|_{\boldsymbol{G}})
\end{aligned}$$

Hence, we have

$$\begin{aligned}
|\boldsymbol{z}_i| &\leq \frac{2}{1+\alpha}\|\boldsymbol{y}'\|_\infty + \frac{1}{1+\alpha}\left(\frac{2\alpha}{1+\alpha}+2\right)\|\boldsymbol{y}'\|_{\boldsymbol{G}} \\
&\leq \frac{2}{1+\alpha}\left[\|\boldsymbol{y}'\|_\infty + 2\|\boldsymbol{y}'\|_{\boldsymbol{G}}\right].
\end{aligned}$$

Therefore, $\|\boldsymbol{B}\boldsymbol{y}\|_\infty = \|\boldsymbol{z}\|_\infty \leq 2(1+\alpha)^{-1}(\|\boldsymbol{y}'\|_\infty + 2\|\boldsymbol{y}'\|_{\boldsymbol{G}})$. Finally, we note that

$$\begin{aligned}
\|\boldsymbol{B}\boldsymbol{y}\|_{\boldsymbol{g}+\infty} &= \|\boldsymbol{B}\boldsymbol{y}\|_\infty + C_{\text{norm}}\|\boldsymbol{B}\boldsymbol{y}\|_{\boldsymbol{G}} && \text{(Definition)} \\
&\leq \frac{2}{1+\alpha}\|\boldsymbol{y}\|_\infty + \frac{2}{1+\alpha}\cdot 2\|\boldsymbol{y}\|_{\boldsymbol{G}} + \frac{2}{1+\alpha}C_{\text{norm}}\|\boldsymbol{y}\|_{\boldsymbol{G}} \\
&\leq \frac{2}{1+\alpha}\left(1+\frac{2}{C_{\text{norm}}}\right)\|\boldsymbol{y}\|_{\boldsymbol{g}+\infty}.
\end{aligned}$$

□

Theorem 6.7.4. *Choosing parameters*

$$\alpha = 1 + \frac{1}{\log_2\left(\frac{2m}{\text{rank}(\boldsymbol{A})}\right)} \quad , \quad \beta = \frac{\text{rank}(\boldsymbol{A})}{2m} \quad , \textit{ and } \quad C_{norm} = 18\log_2\left(\frac{2m}{\text{rank}(\boldsymbol{A})}\right)$$

yields

$$c_1(\boldsymbol{g}) = 2\text{rank}(\boldsymbol{A}) \quad , \quad c_\gamma(\boldsymbol{g}) = 1 + \frac{1}{9\log_2\left(\frac{2m}{\text{rank}(\boldsymbol{A})}\right)} \quad , \textit{ and } \quad c_\delta(\boldsymbol{g}) = 1 - \frac{2}{9\log_2\left(\frac{2m}{\text{rank}(\boldsymbol{A})}\right)}.$$

In particular, we have

$$c_\gamma(\boldsymbol{g})c_\delta(\boldsymbol{g}) \leq 1 - \frac{1}{9\log_2\left(\frac{2m}{\text{rank}(\boldsymbol{A})}\right)}.$$

Proof. The bounds on $c_1(\boldsymbol{g})$ and $c_\delta(\boldsymbol{g})$ follow immediately from Lemma 6.7.2 and Lemma 6.7.3. Now, we estimate the $c_\gamma(\boldsymbol{g})$ and let $\frac{4}{5}\boldsymbol{g} \leq \boldsymbol{w} \leq \frac{5}{4}\boldsymbol{g}$. Fix an arbitrary $\boldsymbol{x}\in\Omega$ and let $\boldsymbol{g} \stackrel{\text{def}}{=} \boldsymbol{g}(\boldsymbol{x})$. Recall that by Lemma 6.7.2, we have $\boldsymbol{g} \geq \beta$. Furthermore, since $\boldsymbol{g}^{-1} = \boldsymbol{g}^{\alpha-1}\boldsymbol{g}^{-\alpha}$ and $\beta^{\alpha-1} \geq \frac{1}{2}$, the following holds

$$\frac{4}{10}\boldsymbol{g}_i^{-\alpha} \leq \frac{4}{5}\beta^{\alpha-1}\boldsymbol{g}_i^{-\alpha} \leq \frac{4}{10}\boldsymbol{g}_i^{-1} \leq \boldsymbol{w}_i^{-1} \tag{6.32}$$

for all i. Applying this and using the definition of $\boldsymbol{P}_{\boldsymbol{A}_x}$ yields

$$\boldsymbol{A}_x(\boldsymbol{A}_x^T\boldsymbol{W}^{-1}\boldsymbol{A}_x)^{-1}\boldsymbol{A}_x^T \preceq \frac{10}{4}\boldsymbol{A}_x(\boldsymbol{A}_x^T\boldsymbol{G}^{-\alpha}\boldsymbol{A}_x)^{-1}\boldsymbol{A}_x^T = \frac{10}{4}\boldsymbol{G}^{\alpha/2}\boldsymbol{P}_{\boldsymbol{A}_x}(\boldsymbol{g}^{-\alpha})\boldsymbol{G}^{\alpha/2} \quad . \tag{6.33}$$

Hence, we have

$$\begin{aligned}
\frac{\boldsymbol{\sigma}_i\left(\frac{1}{\boldsymbol{w}\vec{\phi}''}\right)}{\boldsymbol{w}_i} &= \frac{\mathbf{1}_i^T\boldsymbol{A}_x(\boldsymbol{A}_x^T\boldsymbol{W}^{-1}\boldsymbol{A}_x)^{-1}\boldsymbol{A}_x^T\mathbf{1}_i}{\boldsymbol{w}_i^2} \\
&\leq \frac{10}{4}\frac{\mathbf{1}_i^T\boldsymbol{G}^{\alpha/2}\boldsymbol{P}_{\boldsymbol{A}_x}(\boldsymbol{g}^{-\alpha})\boldsymbol{G}^{\alpha/2}\mathbf{1}_i}{\boldsymbol{w}_i^2} \\
&\leq \frac{10}{4}\left(\frac{5}{4}\right)^2\frac{\boldsymbol{\sigma}_i\left(\frac{1}{\boldsymbol{g}^{\alpha}\vec{\phi}''}\right)}{\boldsymbol{g}_i^{-2\alpha}} < 4.
\end{aligned}$$

Since $\boldsymbol{P}_{\boldsymbol{x},\boldsymbol{w}}$ is an orthogonal projection in $\|\cdot\|_{\boldsymbol{w}}$, we have $\|\boldsymbol{P}_{\boldsymbol{x},\boldsymbol{w}}\|_{\boldsymbol{w}\to\boldsymbol{w}} = 1$. Let $\overline{\boldsymbol{P}}_{\boldsymbol{x},\boldsymbol{w}} \stackrel{\text{def}}{=} \boldsymbol{I} - \boldsymbol{P}_{\boldsymbol{x},\boldsymbol{w}}$, we have

$$\begin{aligned}
\left\|\overline{\boldsymbol{P}}_{\boldsymbol{x},\boldsymbol{w}}\right\|_{\boldsymbol{w}\to\infty} &= \max_{i\in[m]}\max_{\|\boldsymbol{y}\|_{\boldsymbol{w}}\leq 1}\mathbf{1}_i^T\overline{\boldsymbol{P}}_{\boldsymbol{x},\boldsymbol{w}}\boldsymbol{y} \\
&\leq \max_{i\in[m]}\left\|(\boldsymbol{w})^{-1/2}\,\overline{\boldsymbol{P}}_{\boldsymbol{x},\boldsymbol{w}}^T\mathbf{1}_i\right\|^2 \\
&= \max_{i\in[m]}\sqrt{\mathbf{1}_i\boldsymbol{W}^{-1}\boldsymbol{A}_x\left(\boldsymbol{A}_x^T\boldsymbol{W}^{-1}\boldsymbol{A}_x\right)^{-1}\boldsymbol{A}_x^T\boldsymbol{W}^{-1}\mathbf{1}_i}. \\
&= \max_{i\in[m]}\sqrt{\frac{\sigma_i\left(\frac{1}{\boldsymbol{w}\vec{\phi}''}\right)}{w_i}} \leq 2.
\end{aligned}$$

For any $\boldsymbol{y}$, we have

$$\begin{aligned}
\|\boldsymbol{P}_{\boldsymbol{x},\boldsymbol{w}}\boldsymbol{y}\|_{\boldsymbol{w}+\infty} &\leq \|\boldsymbol{P}_{\boldsymbol{x},\boldsymbol{w}}\boldsymbol{y}\|_\infty + C_{\text{norm}}\|\boldsymbol{P}_{\boldsymbol{x},\boldsymbol{w}}\boldsymbol{y}\|_{\boldsymbol{w}} \\
&\leq \|\boldsymbol{y}\|_\infty + \left\|\overline{\boldsymbol{P}}_{\boldsymbol{x},\boldsymbol{w}}\boldsymbol{y}\right\|_\infty + C_{\text{norm}}\|\boldsymbol{y}\|_{\boldsymbol{w}} \\
&\leq \|\boldsymbol{y}\|_\infty + (2 + C_{\text{norm}})\|\boldsymbol{y}\|_{\boldsymbol{w}} \\
&\leq \frac{C_{\text{norm}}+2}{C_{\text{norm}}}\|\boldsymbol{y}\|_{\boldsymbol{w}+\infty}\,.
\end{aligned}$$

Hence, we have $c_\gamma \leq \frac{C_{\text{norm}}+2}{C_{\text{norm}}}$. Thus, we have picked $C_{\text{norm}} = \frac{18}{\alpha-1}$ and have $c_\gamma \leq 1 + \frac{\alpha-1}{9}$. □

6.7.1 Weight computation

Here, we describe how to efficiently compute approximations to the weight function $\boldsymbol{g} : \Omega \to \mathbb{R}^m_{>0}$ as given by (6.30). The two main technical tools we use towards this end are the *gradient descent method*, Theorem 2.3.12, a standard result in convex optimization, and fast numerical methods for estimating leverage scores using the Johnson-Lindenstrauss Lemma, Theorem 2.3.4, a powerful tool in randomized numerical linear algebra.

Since the weight function, $\boldsymbol{g}$, is defined as the minimizer of a convex optimization problem, we could use the gradient descent method directly to minimize $\hat{f}$ and hence compute $\boldsymbol{g}$. Indeed, in Lemma 6.7.6 we show how applying the gradient descent method in a carefully scaled space allows us to compute $\boldsymbol{g}(\boldsymbol{x})$ to high accuracy in $\widetilde{O}(1)$ iterations. Unfortunately, this result makes two assumptions to compute $\boldsymbol{g}(\boldsymbol{x})$: (1) we are given a weight $\boldsymbol{w} \in \Omega$ that is not too far from $\boldsymbol{g}(\boldsymbol{x})$ and (2) we compute the gradient

of $\hat{f}$ exactly.

Assumption (1) is not an issue as we always ensure that $\boldsymbol{g}$ does not change too much between calls to compute $\boldsymbol{g}$ and therefore can always use our previous weights as the approximation to $\boldsymbol{g}(\boldsymbol{x})$. However, naively computing the gradient of $\hat{f}$ is computationally expensive and hence assumption (2) is problematic. To deal with this issue we use the fact that by careful application of Johnson-Lindenstrauss one can compute a multiplicative approximation to the gradient efficiently and in Theorem 6.7.7 we show that this suffices to compute an approximation to $\boldsymbol{g}$ that suffices to use in our weighted path following scheme.

Recall from Theorem 2.3.12 that if we take repeated projected gradient steps then we can achieve linear convergence up to bounds on how much the hessian of the function changes over the domain of interest. To apply this Theorem 2.3.12 to compute $\boldsymbol{g}(\boldsymbol{s})$ we first need to show that there is a region around the optimal point $\boldsymbol{g}(\boldsymbol{s})$ such that the Hessian of $\hat{f}$ does not change too much.

Lemma 6.7.5 (Hessian Approximation). *For $\|\boldsymbol{W}_{(0)}^{-1}(\boldsymbol{g}(\boldsymbol{x})-\boldsymbol{w}^{(0)})\|_\infty \leq \frac{1}{24}$ we have*

$$0.84\boldsymbol{W}^{-1} \preceq \nabla^2_{\boldsymbol{ww}} f_t(\boldsymbol{x},\boldsymbol{w}) \preceq 3.6\boldsymbol{W}^{-1}.$$

Proof. From Lemma 6.7.1, we know that

$$\nabla^2_{\boldsymbol{ww}} f_t(\boldsymbol{x},\boldsymbol{w}) = \boldsymbol{W}^{-1}\left(\boldsymbol{\Sigma}+\beta\boldsymbol{I}+\alpha\boldsymbol{\Lambda}\right)\boldsymbol{W}^{-1}$$

where $\boldsymbol{\Sigma}=\boldsymbol{\Sigma}_{\boldsymbol{A}_x}(\boldsymbol{w}^{-\alpha})$ and $\boldsymbol{\Lambda}=\boldsymbol{\Lambda}_{\boldsymbol{A}_x}(\boldsymbol{w}^{-\alpha})$. Using $\boldsymbol{0}\preceq\boldsymbol{\Lambda}\preceq\boldsymbol{\Sigma}$, we have

$$\boldsymbol{W}^{-1}\left(\boldsymbol{\Sigma}+\beta\boldsymbol{I}\right)\boldsymbol{W}^{-1} \preceq \nabla^2_{\boldsymbol{ww}} f_t(\boldsymbol{x},\boldsymbol{w}) \preceq 3\boldsymbol{W}^{-1}\left(\boldsymbol{\Sigma}+\beta\boldsymbol{I}\right)\boldsymbol{W}^{-1}$$

Using that $\|\boldsymbol{W}_{(0)}^{-1}(\boldsymbol{g}(\boldsymbol{x})-\boldsymbol{w}^{(0)})\|_\infty \leq \frac{1}{24}$, we have

$$\boldsymbol{\Sigma}+\beta\boldsymbol{I} \preceq \left(1-\frac{1}{24}\right)^{-4}\boldsymbol{\Sigma}_{\boldsymbol{A}_x}(\boldsymbol{g}^{-\alpha})+\beta\boldsymbol{I} \preceq \left(1-\frac{1}{24}\right)^{-4}\boldsymbol{G} \preceq 1.19\boldsymbol{W}$$

and

$$\boldsymbol{\Sigma}+\beta\boldsymbol{I} \succeq \left(1-\frac{1}{24}\right)^{4}\boldsymbol{\Sigma}_{\boldsymbol{A}_x}(\boldsymbol{g}^{-\alpha})+\beta\boldsymbol{I} \succeq \left(1-\frac{1}{24}\right)^{4}\boldsymbol{G} \succeq 0.84\boldsymbol{W}.$$

□

Combining Theorem 2.3.12 and Lemma 6.7.5, we get the following algorithm to compute the weight function using the exact computation of the gradient of $\hat{f}$. Note that this algorithm applies Theorem 2.3.12 multiple times as in each iteration we are taking a gradient step with respect to a different norm.

Lemma 6.7.6 (Exact Weight Computation). *Given $\boldsymbol{w}^{(0)}\in\mathbb{R}^m_{>0}$ such that $\|\boldsymbol{W}_{(0)}^{-1}(\boldsymbol{g}(\boldsymbol{x})-\boldsymbol{w}^{(0)})\|_\infty \leq \frac{1}{48}$. Let*

$$Q=\left\{\boldsymbol{w}\in\mathbb{R}^m \mid \|\boldsymbol{W}_{(0)}^{-1}(\boldsymbol{g}(\boldsymbol{x})-\boldsymbol{w}^{(0)})\|_\infty \leq \frac{1}{48}\right\}.$$

For all $k\geq 0$ let

$$\boldsymbol{w}^{(k+1)}=\operatorname*{arg\,min}_{\boldsymbol{w}\in Q}\left\|\boldsymbol{w}-\frac{1}{2}\left(\boldsymbol{w}^{(k)}+\boldsymbol{\sigma}_{\boldsymbol{A}_x}\left(\left(\boldsymbol{w}^{(k)}\right)^{-\alpha}\right)+\beta\right)\right\|^2_{\boldsymbol{W}_{(k)}^{-1}}$$

This implies that for all k ,

$$\left\|\boldsymbol{g}(\boldsymbol{x})^{-1}(\boldsymbol{g}(\boldsymbol{x})-\boldsymbol{w}^{(k)})\right\|^2_\infty \leq 4m^2\left(1-\frac{1}{10}\right)^k.$$

Proof. Note that iterations of Theorem 2.3.12 can be rewritten as

$$\begin{aligned}\boldsymbol{w}^{(k+1)} &= \arg\min_{\boldsymbol{w}\in Q}\left\langle\left(\boldsymbol{I}-\boldsymbol{\Sigma}_{\boldsymbol{A}_x}\left(\left(\boldsymbol{w}^{(k)}\right)^{-\alpha}\right)\boldsymbol{W}_{(k)}^{-1}-\beta\boldsymbol{W}_{(k)}^{-1}\right)\mathbf{1},\boldsymbol{w}\right\rangle+\frac{4}{2}\left\|\boldsymbol{w}-\boldsymbol{w}^{(k)}\right\|_{\boldsymbol{W}_{(k)}^{-1}}^2\\ &= \arg\min_{\boldsymbol{w}\in Q}\left\|\boldsymbol{w}-\frac{1}{4}\left(\boldsymbol{w}^{(k)}+\boldsymbol{\sigma}_{\boldsymbol{A}_x}\left(\left(\boldsymbol{w}^{(k)}\right)^{-\alpha}\right)+\beta\right)\right\|_{\boldsymbol{W}_{(k)}^{-1}}^2\end{aligned}$$

which is the same as in the statement of this lemma. To apply Theorem 2.3.12 we note that for any $\boldsymbol{w}\in Q$ the definition of Q and the fact that $\alpha\in(0,1)$ implies that $(1-\frac{1}{48})\boldsymbol{W}_{(0)}\preceq\boldsymbol{W}\preceq(1+\frac{1}{48})\boldsymbol{W}_{(0)}$. Therefore Lemma 6.7.5 shows that for all $\boldsymbol{w}^{(k)}\in Q$,

$$\frac{4}{5}\boldsymbol{W}_{(k)}^{-1}\preceq 0.84\boldsymbol{W}_{(0)}^{-1}\preceq\nabla_{\boldsymbol{ww}}^2 f_t(\boldsymbol{x},\boldsymbol{w})\preceq 3.6\boldsymbol{W}_{(0)}^{-1}\preceq 4\boldsymbol{W}_{(k)}^{-1}. \tag{6.34}$$

Hence, Theorem 2.3.12 and inequality (6.34) shows that

$$\left\|\boldsymbol{w}^{(k+1)}-\boldsymbol{g}\right\|_{\boldsymbol{W}_{(k)}^{-1}}^2\leq\left(1-\frac{1}{5}\right)\left\|\boldsymbol{w}^{(k)}-\boldsymbol{g}\right\|_{\boldsymbol{W}_{(k)}^{-1}}^2.$$

Since $\left\|\boldsymbol{W}_{(0)}^{-1}(\boldsymbol{g}(\boldsymbol{s})-\boldsymbol{w}^{(0)})\right\|_\infty\leq\frac{1}{48}$ and $\boldsymbol{w}^{(k)}\in Q$ we know that $\boldsymbol{G}\succeq\left(1-\frac{1}{48}\right)^2\boldsymbol{W}_{(k)}$. Hence, we have

$$\begin{aligned}\|\boldsymbol{w}^{(k)}-\boldsymbol{g}\|_{\boldsymbol{G}^{-1}}^2 &\leq \left(1-\frac{1}{48}\right)^{-2}\left(1-\frac{1}{5}\right)\|\boldsymbol{w}^{(k-1)}-\boldsymbol{g}\|_{\boldsymbol{G}^{-1}}^2\\ &\leq \left(1-\frac{1}{10}\right)\|\boldsymbol{w}^{(k-1)}-\boldsymbol{g}\|_{\boldsymbol{G}^{-1}}^2\\ &\leq \left(1-\frac{1}{10}\right)^k\|\boldsymbol{w}^{(0)}-\boldsymbol{g}\|_{\boldsymbol{G}^{-1}}^2\end{aligned}$$

The result follows from the facts that

$$\|\boldsymbol{w}^{(0)}-\boldsymbol{g}\|_{\boldsymbol{G}^{-1}}^2\leq m\|\boldsymbol{g}\|_\infty\|\boldsymbol{g}^{-1}(\boldsymbol{g}-\boldsymbol{w}^{(0)})\|_\infty^2\leq\frac{m(1+\beta)}{\left(1-\frac{1}{48}\right)^2}\|\boldsymbol{w}_{(0)}^{-1}(\boldsymbol{g}-\boldsymbol{w}^{(0)})\|_\infty^2$$

and $\|\boldsymbol{g}^{-1}(\boldsymbol{w}^{(k)}-\boldsymbol{g})\|_\infty^2\leq\beta^{-1}\|\boldsymbol{w}^{(k)}-\boldsymbol{g}\|_{\boldsymbol{G}^{-1}}^2$ where $\beta=\frac{\operatorname{rank}(\boldsymbol{A})}{2m}$. □

Recall that Theorem 2.3.4 showed that we can compute leverage scores, $\boldsymbol{\sigma}_{\boldsymbol{A}_x}$, approximately by solving only $\widetilde{O}(1)$ many regression problems. Now, we show that we can modify Lemma 6.7.6 and use Lemma 2.3.4. Our weight computation and the analysis is as follows.

Algorithm 9: $\boldsymbol{w}=\texttt{computeWeight}(\boldsymbol{x},\boldsymbol{w}^{(0)},K)$

Let $\alpha=1+\frac{1}{\log_2\left(\frac{2m}{\operatorname{rank}(\boldsymbol{A})}\right)}$, $\beta=\frac{\operatorname{rank}(\boldsymbol{A})}{2m}$, $\varepsilon=\frac{K}{40c_r\log\left(\frac{2m}{K}\right)}$.

Let $Q=\left\{\boldsymbol{w}\in\mathbb{R}^m\mid\left\|\boldsymbol{W}_{(0)}^{-1}(\boldsymbol{w}-\boldsymbol{w}^{(0)})\right\|_\infty\leq\frac{1}{48}\right\}$.

for $j=1$ *to* k *where* $k=\lceil 20\log\left(\frac{4m}{K}\right)\rceil$ **do**

  Let $\boldsymbol{\sigma}^{(j)}$ be a $(1+\varepsilon)$ approximation of $\boldsymbol{\sigma}_{\boldsymbol{A}}\left(\left(\boldsymbol{w}^{(j)}\right)^{-\alpha}\right)$. (Theorem 2.3.4)

  Set $\boldsymbol{w}^{(j)}=\arg\min_{\boldsymbol{w}\in Q}\left\|\boldsymbol{w}-\frac{1}{4}\left(\boldsymbol{w}^{(j-1)}+\boldsymbol{\sigma}^{(j)}+\beta\mathbf{1}\right)\right\|_{\boldsymbol{w}_{(j-1)}^{-1}}^2$

end

Output $\boldsymbol{w}^{(j)}$.

Note that the convex set Q is aligned with standard basis and hence the step on $\boldsymbol{w}^{(j)}$ can be

computed by explicit formula (6.36).

Theorem 6.7.7 (Approximate Weight Computation). *Let* $\boldsymbol{x} \in \Omega$, $\|\boldsymbol{W}_{(0)}^{-1}(\boldsymbol{g}(\boldsymbol{x}) - \boldsymbol{w}^{(0)})\|_\infty \leq \frac{1}{48}$ *and* $K \in (0,1)$. *The algorithm* `computeWeight`$(\boldsymbol{x}, \boldsymbol{w}^{(0)}, K)$ *returns* $\boldsymbol{w}$ *such that*

$$\|\boldsymbol{g}(\boldsymbol{x})^{-1}(\boldsymbol{g}(\boldsymbol{x}) - \boldsymbol{w})\|_\infty \leq K$$

with probability $1 - \frac{\tilde{O}(1)}{m}$.

The running time is dominated by the time needed to solve $\tilde{O}(\log^3(1/K)/K^2)$ *linear systems.*

Proof. Consider an execution of `computeWeight`$(\boldsymbol{x}, \boldsymbol{w}^{(0)}, K)$ where $\boldsymbol{\sigma}^{(j)} = \boldsymbol{\sigma}_{\boldsymbol{A}_s}\left((\boldsymbol{w}^{(j)})^{-\alpha}\right)$ exactly, i.e. $\boldsymbol{\sigma}^{(j)} = \boldsymbol{\sigma}_{\boldsymbol{A}_s}((\boldsymbol{w})^{-\alpha})$, and let $\boldsymbol{v}^{(j)}$denote the $\boldsymbol{w}$ computed during this idealized execution of `computeWeight`.

Now suppose that for all $i \in [m]$ we have

$$(1-\varepsilon)^M \boldsymbol{v}_i^{(j)} \leq \boldsymbol{w}_i^{(j)} \leq (1+\varepsilon)^M \boldsymbol{v}_i^{(j)} \tag{6.35}$$

for some $M \geq 0$ and $j \in [k-1]$. Since the objective function and the constraint Q are axis-aligned we can compute $\boldsymbol{w}^{(j)}$ coordinate-wise and we see that

$$\boldsymbol{w}^{(j+1)} = \texttt{median}\left(\left(1 - \frac{1}{48}\right)\boldsymbol{w}^{(0)}, \frac{1}{4}\left(\boldsymbol{\sigma}_{\boldsymbol{A}_s}\left(\left(\boldsymbol{w}^{(j)}\right)^{\alpha}\right) + \beta\right), \left(1 + \frac{1}{48}\right)\boldsymbol{w}^{(0)}\right) \tag{6.36}$$

where $[\texttt{median}\,(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})]_i$ is equal to the median of x_i, y_i and z_i for all $i \in [m]$. By (6.35), (6.36), and the fact that $(1-\varepsilon)\,\sigma_{\boldsymbol{A}_x}\left(\left(\boldsymbol{w}^{(j+1)}\right)^{\alpha}\right)_i \leq \boldsymbol{\sigma}_i^{(j+1)} \leq (1+\varepsilon)\,\sigma_{\boldsymbol{A}_x}\left(\left(\boldsymbol{w}^{(j+1)}\right)^{\alpha}\right)_i$ for all $i \in [m]$, we have that

$$(1-\varepsilon)^{M+1}\boldsymbol{v}_i^{(j+1)} \leq \boldsymbol{w}_i^{(j+1)} \leq (1+\varepsilon)^{M+1}\boldsymbol{v}_i^{(j+1)}.$$

Since $\boldsymbol{v}^{(0)} = \boldsymbol{w}^{(0)}$ and since $j \in [k-1]$ was arbitrary we can apply induction and we have that for all $j \in [k]$

$$(1-\varepsilon)^j \boldsymbol{v}_i^{(j)} \leq \boldsymbol{w}_i^{(j)} \leq (1+\varepsilon)^j \boldsymbol{v}_i^{(j)}.$$

Note that $k\varepsilon \leq \frac{1}{8}$ and therefore by Taylor series expansion we have $\|\boldsymbol{V}_{(k)}^{-1}\left(\boldsymbol{w}^{(k)} - \boldsymbol{v}^{(k)}\right)\|_\infty \leq \frac{9}{8}\varepsilon k$. Furthermore since $\boldsymbol{v}^{(k)} \in Q$ we know that $\boldsymbol{G}(\boldsymbol{s}) \succeq \left(1 - \frac{1}{24}\right)\boldsymbol{V}_{(k)}$. Putting these together, applying Lemma 6.7.6, and recalling that $k = \lceil 20 \log\left(\frac{4m}{K}\right)\rceil$ we have

$$\begin{aligned}
\|\boldsymbol{g}(\boldsymbol{x})^{-1}(\boldsymbol{g}(\boldsymbol{x}) - \boldsymbol{w}^{(k)})\|_\infty &\leq \|\boldsymbol{g}(\boldsymbol{x})^{-1}(\boldsymbol{g}(\boldsymbol{x}) - \boldsymbol{v}^{(k)})\|_\infty + \|\boldsymbol{g}(\boldsymbol{x})^{-1}\left(\boldsymbol{v}^{(k)} - \boldsymbol{w}^{(k)}\right)\|_\infty \\
&\leq 2m\left(1 - \frac{1}{10}\right)^{\frac{k}{2}} + \left(1 - \frac{1}{24}\right)^{-1}\|\boldsymbol{V}_{(k)}^{-1}(\boldsymbol{v}^{(k)} - \boldsymbol{w}^{(k)})\|_\infty \\
&\leq 2m \cdot \exp\left(-\frac{k}{20}\right) + 1.5k\varepsilon \\
&\leq \frac{K}{2} + 1.5\varepsilon\lceil 20 \log\left(\frac{4m}{K}\right)\rceil \leq K
\end{aligned}$$

□

Finally, we show how to compute an initial weight without having an approximate weight to help the computation. The algorithm `computeInitialWeight`$(\boldsymbol{x}, K)$ computes an initial weight in $\tilde{O}\left(\sqrt{\text{rank}\boldsymbol{A}}\right)$ iterations of `computeWeight` by computing $\boldsymbol{g}$ for a large enough value of β that the

calculation is trivial and then decreasing β gradually.

Algorithm 10: $\boldsymbol{w} = \texttt{computeInitialWeight}(\boldsymbol{s}, K)$

Let $\alpha = 1 + \frac{1}{\log_2\left(\frac{2m}{\text{rank}(\boldsymbol{A})}\right)}$, $\beta = 48$ and $\boldsymbol{w} = \beta\mathbf{1}$.

while $\beta > \frac{\text{rank}(\boldsymbol{A})}{2m}$ **do**

  Set $\boldsymbol{w} = \texttt{computeWeight}(\boldsymbol{x}, \boldsymbol{w}, \frac{1}{100})$.

  Set $\beta = \max\left\{\left(1 - \frac{1}{1000\sqrt{\text{rank}(\boldsymbol{A})}}\right)\beta, \frac{\text{rank}(\boldsymbol{A})}{2m}\right\}$.

end

Output $\texttt{computeWeight}(\boldsymbol{x}, \boldsymbol{w}, K)$.

Theorem 6.7.8 (Computating Initial Weights). *For $\boldsymbol{x} \in \Omega$ and $K > 0$, with constant probability the algorithm $\texttt{computeInitialWeight}(\boldsymbol{x}, K)$ returns $\boldsymbol{w} \in \mathbb{R}^m_{>0}$ such that*

$$\|\boldsymbol{g}(\boldsymbol{x})^{-1}(\boldsymbol{g}(\boldsymbol{x}) - \boldsymbol{w})\|_\infty \leq K.$$

The total running time of $\texttt{computeInitialWeight}(\boldsymbol{x}, K)$ is dominated by the time needed to solve $\tilde{O}(\sqrt{\text{rank}\,(\boldsymbol{A})}\log^3(1/K)/K^2)$ linear systems.

Proof. Fix $\boldsymbol{x} \in \Omega$. Let $\boldsymbol{g} : \Omega \to \mathbb{R}^m$ be defined by[7]

$$\boldsymbol{g}(\beta) \stackrel{\text{def}}{=} \arg\min_{\boldsymbol{w} \in \mathbb{R}^m_{>0}} \mathbf{1}^T\boldsymbol{w} + \frac{1}{\alpha}\log\det(\boldsymbol{A}_x^T\boldsymbol{W}^{-\alpha}\boldsymbol{A}_x) - \beta\sum_{i\in[m]}\log w_i.$$

We first prove by induction that at the beginning of each while iteration, we have that

$$\|\boldsymbol{W}^{-1}(\boldsymbol{g}(\beta) - \boldsymbol{w})\|_\infty \leq \frac{1}{48}. \tag{6.37}$$

First, we prove that (6.37) holds at the first iteration. Since $\boldsymbol{g}(\beta) = \boldsymbol{\sigma}(\beta) + \beta$ where $\boldsymbol{\sigma}(\beta) \stackrel{\text{def}}{=} \boldsymbol{\sigma}_{\boldsymbol{A}_x}(\boldsymbol{g}^{-\alpha}(\beta))$, we have that for all $i \in [m]$

$$\beta \leq g(\beta)_i \leq 1 + \beta.$$

Therefore, the initial weight, $\boldsymbol{w} = \beta\mathbf{1} \in \mathbb{R}^m_{>0}$ satisfies the invariant (6.37).

Now, we assume (6.37) holds at the beginning of a certain iteration. After the step of `computeWeight`, by Theorem 6.7.7 we have

$$\|\boldsymbol{G}(\beta)^{-1}(\boldsymbol{g}(\beta) - \boldsymbol{w})\|_\infty \leq \frac{1}{100}. \tag{6.38}$$

Therefore, it suffices to prove that $\boldsymbol{g}(\beta)$ is close to $\boldsymbol{g}(\beta - \theta)$ for small θ.

To bound how much $\boldsymbol{g}(\beta)$ changes for small changes in β we proceed similarly to Lemma 6.7.2. First by the implicit function theorem (Lemma 2.3.9) and direct calculation we know that

$$\frac{d\boldsymbol{g}}{d\beta} = -\left(\boldsymbol{J}_{\boldsymbol{w}}(\nabla_w \hat{f}(\boldsymbol{x}, \boldsymbol{w}))\right)^{-1}\left(\boldsymbol{J}_\beta(\nabla_{\boldsymbol{w}}\hat{f}(\boldsymbol{x}, \boldsymbol{w}))\right) = \boldsymbol{G}(\beta)\left(\boldsymbol{G}(\beta) + \alpha\boldsymbol{\Lambda}_g\right)^{-1}\mathbf{1} \tag{6.39}$$

where $\boldsymbol{\Lambda}_g \stackrel{\text{def}}{=} \boldsymbol{\Lambda}_{\boldsymbol{A}_x}(\boldsymbol{G}(\beta)^{-\alpha}\mathbf{1})$. Next to estimate how fast $\boldsymbol{g}$ can change as a function of β we estimate (6.39) in a similar manner to Lemma 6.7.3. Note that

$$\boldsymbol{G}(\beta) + \alpha\boldsymbol{\Lambda}_g \succeq \boldsymbol{\Sigma}(\beta)$$

[7] Note that early we assumed that $\beta < 1$ and here we use much larger values of β. However, this bound on β was primarily to assist in bounding c_1 and does not affect this proof.

where $\boldsymbol{\Sigma}(\beta) \stackrel{\text{def}}{=} \boldsymbol{\Sigma}_{\boldsymbol{A}_s}(\boldsymbol{g}^{\alpha}(\beta))$. Consequently,

$$\begin{aligned}\left\|\boldsymbol{G}(\beta)^{-1}\frac{d\boldsymbol{g}}{d\beta}\right\|_{\boldsymbol{\Sigma}(\beta)}^2 &\leq \left\|\left(\boldsymbol{G}(\beta)+\alpha\boldsymbol{\Lambda}_g\right)^{-1}\mathbf{1}\right\|_{\boldsymbol{\Sigma}(\beta)}^2 \\ &\leq \|\mathbf{1}\|_{\boldsymbol{\Sigma}(\beta)}^2 = \operatorname{rank}(\boldsymbol{A}). \end{aligned} \tag{6.40}$$

Using this estimate of how much $\boldsymbol{g}$ changes in the $\boldsymbol{\Sigma}(\beta)$ norm, we now estimate how much $\boldsymbol{g}$ changes in the ℓ_∞ norm. Let $\boldsymbol{z} \stackrel{\text{def}}{=} \left(\boldsymbol{G}(\beta)+\alpha\boldsymbol{\Lambda}_g\right)^{-1}\mathbf{1}$. Then, we have

$$\begin{aligned}\left((1+\alpha)\,\boldsymbol{\sigma}_i(\beta)+\beta\right)|\boldsymbol{z}_i| &\leq \left|\alpha\mathbf{1}_i^T\boldsymbol{P}^{(2)}\boldsymbol{z}\right|+1 \\ &\leq \alpha\boldsymbol{\sigma}_i(\beta)\|\boldsymbol{z}\|_{\boldsymbol{\Sigma}(\beta)}+1.\end{aligned}$$

Using (6.40) and $\alpha<1$, we have

$$\left\|\frac{d\ln\boldsymbol{g}}{d\beta}\right\|_\infty = \|\boldsymbol{z}\|_\infty \leq \max\left(\frac{\alpha\|\boldsymbol{z}\|_{\boldsymbol{\Sigma}(\beta)}}{1+\alpha},\frac{1}{\beta}\right) \leq \max\left(\sqrt{\operatorname{rank}(\boldsymbol{A})},\frac{1}{\beta}\right).$$

Using (6.38), we have that

$$\|\boldsymbol{G}(\beta-\theta)^{-1}(\boldsymbol{g}(\beta-\theta)-\boldsymbol{w})\|_\infty \leq \frac{1}{48}$$

for $\theta \leq \frac{\beta}{1000\sqrt{\operatorname{rank}(\boldsymbol{A})}}$.

Hence, this proves the while loop preserves the invariant (6.37). Therefore, the assumptions needed for Theorem 6.7.7 throughout and `computeWeight` works as desired. Since each iteration β decreased by $\tilde{O}\left(1/\sqrt{\operatorname{rank}(\boldsymbol{A})}\right)$ portion and the initial β is $\tilde{O}(1)$ we see that the algorithm requires only $\tilde{O}\left(\sqrt{\operatorname{rank}(\boldsymbol{A})}\right)$ iterations. Using Theorem 6.7.7 to bound the total number of linear systems solved then yields the result. □

6.8 The Algorithm

Here we show how to use the results of previous sections to solve (6.9) using exact linear system solver. In the next section we will discuss how to relax this assumption. The central goal of this section is to develop an algorithm, `LPSolve`, for which we can prove the following theorem

Theorem 6.8.1. *Suppose we have an interior point $\boldsymbol{x}_0 \in \Omega$ for the linear program (6.9). Then, the algorithm `LPSolve` outputs $\boldsymbol{x}$ such that $\boldsymbol{c}^T\boldsymbol{x} \leq OPT+\varepsilon$ in $\tilde{O}\left(\sqrt{\operatorname{rank}(\boldsymbol{A})}\left(\mathcal{T}_w+\operatorname{nnz}(\boldsymbol{A})\right)\log\left(U/\varepsilon\right)\right)$ work and $\tilde{O}\left(\sqrt{\operatorname{rank}(\boldsymbol{A})}\mathcal{T}_d\log\left(U/\varepsilon\right)\right)$ depth where $U = \max\left(\left\|\frac{\boldsymbol{u}-\boldsymbol{l}}{\boldsymbol{u}-\boldsymbol{x}_0}\right\|_\infty, \left\|\frac{\boldsymbol{u}-\boldsymbol{l}}{\boldsymbol{x}_0-\boldsymbol{l}}\right\|_\infty, \|\boldsymbol{u}-\boldsymbol{l}\|_\infty, \|\boldsymbol{c}\|_\infty\right)$ and $\mathcal{T}_w$ and $\mathcal{T}_d$ is the work and depth needed to compute $\left(\boldsymbol{A}^T\boldsymbol{D}\boldsymbol{A}\right)^{-1}\boldsymbol{q}$ for input positive definite diagonal matrix $\boldsymbol{D}$ and vector $\boldsymbol{q}$.*

We break this proof into several parts. First we provide Lemma 6.8.2, and adaptation of a proof from [206, Thm 4.2.7] that allows us to reason about the effects of making progress along the weighted central path. Then we provide Lemma 6.8.3 that we use to bound the distance to the weighted central path in terms of centrality. After that in Lemma 6.8.3, we analyze a subroutine, `pathFollowing`, for following the weighted central path. Using these lemmas we conclude by describing our `LPSolve` algorithm and proving Theorem 6.8.1.

Lemma 6.8.2 ([206, Theorem 4.2.7]). *Let $x^* \in \mathbb{R}^m$ denote an optimal solution to (6.9) and $\boldsymbol{x}_t = \arg\min f_t(\boldsymbol{x}, \boldsymbol{w})$ for some $t > 0$ and $\boldsymbol{w} \in \mathbb{R}^m_{>0}$. Then the following holds*

$$\boldsymbol{c}^T \boldsymbol{x}_t(\boldsymbol{w}) - \boldsymbol{c}^T \boldsymbol{x}^* \leq \frac{\|\boldsymbol{w}\|_1}{t}.$$

Proof. By the optimality conditions of (6.9) we know that $\nabla_x f_t(\boldsymbol{x}_t(\boldsymbol{w})) = t \cdot \boldsymbol{c} + \boldsymbol{w}\vec{\phi}'(\boldsymbol{x}_t(\boldsymbol{w}))$ is orthogonal to the kernel of $\boldsymbol{A}^T$. Furthermore since $\boldsymbol{x}_t(\boldsymbol{w}) - \boldsymbol{x}^* \in \ker(\boldsymbol{A}^T)$ we have

$$\left(t \cdot \boldsymbol{c} + \boldsymbol{w}\vec{\phi}'(\boldsymbol{x}_t(\boldsymbol{w}))\right)^T (\boldsymbol{x}_t(\boldsymbol{w}) - \boldsymbol{x}^*) = 0.$$

Using that $\phi_i'(x_t(\boldsymbol{w})_i) \cdot (x_i^* - x_t(\boldsymbol{w})_i) \leq 1$ by Lemma 6.5.2 then yields

$$\boldsymbol{c}^T(\boldsymbol{x}_t(\boldsymbol{w}) - \boldsymbol{x}^*) = \frac{1}{t} \sum_{i \in [m]} w_i \cdot \phi_i'(x_t(\boldsymbol{w})_i) \cdot (x_i^* - x_t(\boldsymbol{w})_i) \leq \frac{\|\boldsymbol{w}\|_1}{t}.$$

□

Lemma 6.8.3. *For $\delta_t(\boldsymbol{x}^{(1)}, \boldsymbol{g}(\boldsymbol{x}^{(1)})) \leq \frac{1}{960 c_k^2 \log(400m)}$ and $\boldsymbol{x}_t \stackrel{\text{def}}{=} \arg\min f_t(\boldsymbol{x}, \boldsymbol{w})$ we have*

$$\left\| \sqrt{\vec{\phi}''(\boldsymbol{x}_t)} \left(\boldsymbol{x}^{(1)} - \boldsymbol{x}_t\right) \right\|_\infty \leq 16 c_\gamma c_k \delta_t(\boldsymbol{x}^{(1)}, \boldsymbol{g}(\boldsymbol{x}^{(1)})).$$

Proof. We use Theorem 6.6.8 with exact weight computation and start with $\boldsymbol{x}^{(1)}$ and $\boldsymbol{w}^{(1)} = \boldsymbol{g}(\boldsymbol{x}^{(1)})$. In each iteration, δ_t is decreased by a factor of $\left(1 - \frac{1}{4c_k}\right)$. (6.16) shows that

$$\| \sqrt{\vec{\phi}''(\boldsymbol{x}^{(k)})} \left(\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}\right) \|_\infty \leq c_\gamma \delta_t(\boldsymbol{x}^{(k)}, \boldsymbol{w}^{(k)}).$$

The Lemma 6.5.1 shows that

$$\begin{aligned} \left\| \log\left(\boldsymbol{\phi}''(\boldsymbol{x}^{(k)})\right) - \log\left(\boldsymbol{\phi}''(\boldsymbol{x}^{(k+1)})\right) \right\|_\infty &\leq \left(1 - 2c_\gamma \delta_t(\boldsymbol{x}^{(k)}, \boldsymbol{w}^{(k)})\right)^{-1} \\ &\leq e^{4c_\gamma \delta_t(\boldsymbol{x}^{(k)}, \boldsymbol{w}^{(k)})}. \end{aligned}$$

Therefore, for any k, we have

$$\left\| \log\left(\boldsymbol{\phi}''(\boldsymbol{x}^{(1)})\right) - \log\left(\boldsymbol{\phi}''(\boldsymbol{x}^{(k)})\right) \right\|_\infty \leq e^{4c_\gamma \sum \delta_t(\boldsymbol{x}^{(k)}, \boldsymbol{w}^{(k)})} \leq e^{32 c_k c_\gamma \delta_t(\boldsymbol{x}^{(1)}, \boldsymbol{g}(\boldsymbol{x}^{(1)}))} \leq 2.$$

Hence, for any k, we have

$$\begin{aligned} \left\| \sqrt{\vec{\phi}''(\boldsymbol{x}_t)} \left(\boldsymbol{x}^{(1)} - \boldsymbol{x}^{(k)}\right) \right\|_\infty &\leq \sum 2c_\gamma \delta_t(\boldsymbol{x}^{(k)}, \boldsymbol{w}^{(k)}) \\ &\leq 16 c_\gamma c_k \delta_t(\boldsymbol{x}^{(1)}, \boldsymbol{g}(\boldsymbol{x}^{(1)})). \end{aligned}$$

It is clear now $\boldsymbol{x}^{(k)}$ forms a Cauchy sequence and converges to $\boldsymbol{x}_t$ because δ_t continuous and $\boldsymbol{x}_t$ is the unique point such that $\delta_t = 0$. □

Next, we put together the results of Section 6.6 and analyze the following algorithm for following

the weighted central path.

Algorithm 11: $(\boldsymbol{x}^{(\mathbf{final})}, \boldsymbol{w}^{(\mathbf{new})}) = \texttt{pathFollowing}(\boldsymbol{x}, \boldsymbol{w}, t_{\mathbf{start}}, t_{\mathbf{end}}, \varepsilon)$

Let $c_k = 9\log_2\left(\frac{2m}{\mathrm{rank}(\boldsymbol{A})}\right), t = t_{\mathrm{start}}, K = \frac{1}{20c_k}$.

while *($t < t_{end}$ if $t_{start} < t_{end}$) or ($t > t_{end}$ if $t_{start} > t_{end}$)* **do**

Set $(\boldsymbol{x}^{(\mathrm{new})}, \boldsymbol{w}^{(\mathrm{new})}) = \texttt{centeringInexact}(\boldsymbol{x}, \boldsymbol{w}, K)$ where it uses the function `computeWeight` to find the approximation of $\boldsymbol{g}(\boldsymbol{x})$.

Set $t \leftarrow t\left(1 \pm \frac{1}{10^5 c_k^4 \log(400m)\sqrt{\mathrm{rank}(\boldsymbol{A})}}\right)$ where the sign of $\pm$ is the sign of $t_{\mathrm{end}} - t_{\mathrm{start}}$.

Set $\boldsymbol{x} \leftarrow \boldsymbol{x}^{(\mathrm{final})}$ and $\boldsymbol{w} \leftarrow \boldsymbol{w}^{(\mathrm{new})}$.

end

for $i = 1, \cdots, 4c_k \log(1/\varepsilon)$ **do**

Set $(\boldsymbol{x}, \boldsymbol{w}) = \texttt{centeringInexact}(\boldsymbol{x}, \boldsymbol{w}, K)$ where it uses the function `computeWeight` to find the approximation of $\boldsymbol{g}(\boldsymbol{x})$.

end

Output $(\boldsymbol{x}, \boldsymbol{w})$.

Theorem 6.8.4. *Suppose that*

$$\delta_{t_{start}}(\boldsymbol{x}, \boldsymbol{w}) \leq \frac{1}{960c_k^2 \log(400m)} \quad \text{and} \quad \Phi_\mu(\boldsymbol{\Psi}(\boldsymbol{x}, \boldsymbol{w})) \leq (400m)^2.$$

where $\mu = 2\log(400m)/K$. *Let* $(\boldsymbol{x}^{(final)}, \boldsymbol{w}^{(new)}) = \texttt{pathFollowing}(\boldsymbol{x}, \boldsymbol{w}, t_{start}, t_{end})$, *then*

$$\delta_{t_{end}}(\boldsymbol{x}^{(final)}, \boldsymbol{w}^{(new)}) \leq \varepsilon \quad \text{and} \quad \Phi_\mu(\boldsymbol{\Psi}(\boldsymbol{x}^{(final)}, \boldsymbol{w}^{(new)})) \leq (400m)^2.$$

Furthermore, $\texttt{pathFollowing}(\boldsymbol{x}, \boldsymbol{w}, t_{start}, t_{end})$ *takes time*

$$\tilde{O}\left(\sqrt{\mathrm{rank}(\boldsymbol{A})}\left(\left|\log\left(\frac{t_{end}}{t_{start}}\right)\right| + \log(1/\varepsilon)\right)(\mathcal{T} + m)\right)$$

where $\mathcal{T}$ *is the time needed to solve on linear system.*

Proof. We use induction to prove the invariant that $\delta_t(\boldsymbol{x}, \boldsymbol{w}) \leq \frac{1}{960c_k^2 \log(400m)}$ and $\Phi_\alpha(\boldsymbol{\Psi}(\boldsymbol{x}, \boldsymbol{w})) \leq (400m)^2$ on the beginning of while iteration. The initial condition holds by assumption.

Now, assume the invariant holds at a certain iteration. Theorem 6.6.8 shows that

$$\|\log(\boldsymbol{g}(\boldsymbol{x}^{(\mathrm{new})})) - \log(\boldsymbol{w})\|_\infty \leq K \leq \frac{1}{20c_k}. \tag{6.41}$$

Thus, the weight satisfies the condition of Theorem 6.7.7 and the algorithm `centeringInexact` can use the function `computeWeight` to find the approximation of $\boldsymbol{g}(\boldsymbol{x}^{(\mathrm{new})})$. Consequently,

$$\delta_t(\boldsymbol{x}^{(\mathrm{final})}, \boldsymbol{w}^{(\mathrm{new})}) \leq \left(1 - \frac{1}{4c_k}\right)\delta_t \quad \text{and} \quad \Phi_\alpha(\boldsymbol{\Psi}(\boldsymbol{x}^{(\mathrm{final})}, \boldsymbol{w}^{(\mathrm{new})})) \leq (400m)^2.$$

Using Lemma 6.6.1, (6.41) and Theorem 6.7.4,we have

$$\delta_t(\boldsymbol{x}^{(\mathrm{final})}, \boldsymbol{w}^{(\mathrm{new})}) \leq \frac{1}{960c_k^2 \log(400m)}.$$

Hence, we proved that the invariant holds after one iteration. The $\delta_t < \varepsilon$ bounds follows from the last loop. □

Algorithm 12: $\boldsymbol{x}^{(\mathrm{final})} = \mathtt{LPSolve}(\boldsymbol{x}, \varepsilon)$

Let $\beta = \frac{\mathrm{rank}(\boldsymbol{A})}{2m}$, $\boldsymbol{w} = \mathtt{computeInitialWeight}(\boldsymbol{x}, \frac{1}{10^5 \log^5(400m)})$, $d = -\boldsymbol{w}_i \phi_i'(\boldsymbol{x})$.
Let $t_1 = (10^{10} U^2 m^3)^{-1}$, $t_2 = 3m/\varepsilon$, $\varepsilon_1 = \frac{1}{2000 c_k^2 \log(400m)}$, $\varepsilon_2 = \frac{\varepsilon}{100^3 m^3 U^2}$.
Set $(\boldsymbol{x}^{(\mathrm{new})}, \boldsymbol{w}^{(\mathrm{new})}) = \mathtt{pathFollowing}(\boldsymbol{x}, \boldsymbol{w}, 1, t_1, \varepsilon_1)$ with cost vector $\boldsymbol{d}$.
Set $(\boldsymbol{x}^{(\mathrm{final})}, \boldsymbol{w}^{(\mathrm{final})}) = \mathtt{pathFollowing}(\boldsymbol{x}^{(\mathrm{new})}, \boldsymbol{w}^{(\mathrm{new})}, t_1, t_2, \varepsilon_2)$ with cost vector $\boldsymbol{c}$.
Output $\boldsymbol{x}^{(\mathrm{final})}$.

Proof of Theorem 6.8.1. By Theorem 6.7.8, we know the initial weight satisfies

$$\|\boldsymbol{G}(\boldsymbol{x})^{-1}(\boldsymbol{g}(\boldsymbol{x}) - \boldsymbol{w})\|_\infty \le \frac{1}{10^5 \log^5(400m)}.$$

By the definition of $\boldsymbol{d}$, we have $\boldsymbol{x}$ is the minimum of $\min \boldsymbol{d}^T \boldsymbol{x} - \sum \boldsymbol{w}_i \phi_i(\boldsymbol{x})$ given $\boldsymbol{A}^T \boldsymbol{x} = \boldsymbol{b}$. Therefore, $(\boldsymbol{x}, \boldsymbol{w})$ satisfies the assumption of Theorem 6.8.4 because $\delta_t = 0$ and Φ_α is small enough. Hence, we have

$$\delta_{t_1}(\boldsymbol{x}^{(\mathrm{new})}, \boldsymbol{w}^{(\mathrm{new})}) \le \frac{1}{2000 c_k^2 \log(400m)} \quad \text{and} \quad \Phi_\alpha(\boldsymbol{\Psi}(\boldsymbol{x}^{(\mathrm{new})}, \boldsymbol{w}^{(\mathrm{new})})) \le (400m)^2.$$

Lemma 6.5.2 shows that $\|\phi_i'(\boldsymbol{x})\|_\infty \le U$ and hence $\|\boldsymbol{c} - \boldsymbol{d}\|_\infty \le 2U$. Also, Lemma 6.5.1 shows that $\min_{\boldsymbol{y}} \sqrt{\vec{\phi}''(\boldsymbol{y})} \ge \frac{1}{U}$. Therefore, we have

$$\begin{aligned}
\delta_{t_1}^{\boldsymbol{c}}(\boldsymbol{x}^{(\mathrm{new})}, \boldsymbol{w}^{(\mathrm{new})}) &= \min_{\boldsymbol{\eta} \in \mathbb{R}^n} \left\| \frac{t_1 \boldsymbol{c} + \boldsymbol{w} \vec{\phi}'(\boldsymbol{x}^{(\mathrm{new})}) - \boldsymbol{A}\boldsymbol{\eta}}{\boldsymbol{w}^{(\mathrm{new})} \sqrt{\vec{\phi}''(\boldsymbol{x}^{(\mathrm{new})})}} \right\|_{\boldsymbol{w}^{(\mathrm{new})} + \infty} \\
&\le \min_{\boldsymbol{\eta} \in \mathbb{R}^n} \left\| \frac{t_1 \boldsymbol{d} + \boldsymbol{w} \vec{\phi}'(\boldsymbol{x}^{(\mathrm{new})}) - \boldsymbol{A}\boldsymbol{\eta}}{\boldsymbol{w}^{(\mathrm{new})} \sqrt{\vec{\phi}''(\boldsymbol{x}^{(\mathrm{new})})}} \right\|_{\boldsymbol{w}^{(\mathrm{new})} + \infty} + t_1 \left\| \frac{\boldsymbol{c} - \boldsymbol{d}}{\boldsymbol{w}^{(\mathrm{new})} \sqrt{\vec{\phi}''(\boldsymbol{x}^{(\mathrm{new})})}} \right\|_{\boldsymbol{w}^{(\mathrm{new})} + \infty} \\
&\le \delta_{t_1}^{\boldsymbol{d}}(\boldsymbol{x}^{(\mathrm{new})}, \boldsymbol{w}^{(\mathrm{new})}) + 4U^2 t_1 \|\boldsymbol{1}\|_{\boldsymbol{w} + \infty} \\
&= \delta_{t_1}^{\boldsymbol{d}}(\boldsymbol{x}^{(\mathrm{new})}, \boldsymbol{w}^{(\mathrm{new})}) + 100 m U^2 t_1.
\end{aligned}$$

Since we have chosen t_1 small enough, we have $\delta_{t_1}^{\boldsymbol{c}}(\boldsymbol{x}^{(\mathrm{new})}, \boldsymbol{w}^{(\mathrm{new})})$ is small enough to satisfy the assumption of Theorem 6.8.4. So, we only need to prove how large t_2 should be and how small ε_2 should be in order to get $\boldsymbol{x}$ such that $\boldsymbol{c}^T \boldsymbol{x} \le \mathrm{OPT} + \varepsilon$. By Lemma 6.8.2 and $\|\boldsymbol{w}^{(\mathrm{final})}\| \le 3m$, we have

$$\boldsymbol{c}^T \boldsymbol{x}_{t_2} \le \mathrm{OPT} + \frac{3m}{t_2}.$$

Also, Lemma 6.8.3 shows that we have

$$\left\| \sqrt{\vec{\phi}''(\boldsymbol{x}_{t_2})} \left(\boldsymbol{x}^{(\mathrm{final})} - \boldsymbol{x}_{t_2} \right) \right\|_\infty \le 32 \varepsilon_2 c_k.$$

Using $\min_{\boldsymbol{y}} \sqrt{\vec{\phi}''(\boldsymbol{y})} \ge \frac{1}{U}$, we have $\left\| \boldsymbol{x}^{(\mathrm{final})} - \boldsymbol{x}_{t_2} \right\|_\infty \le 32 \varepsilon_2 c_k U$ and hence our choice of t_2 and ε_2 gives the result

$$\boldsymbol{c}^T \boldsymbol{x}^{(\mathrm{final})} \le \mathrm{OPT} + \frac{3m}{t_2} + 32 \varepsilon_2 c_k U^2 \le \mathrm{OPT} + \varepsilon.$$

□

6.9 Generalized Minimum Cost Flow

In this section we show how to use the interior point method in Section 6.8 to solve the maximum flow problem in time $\widetilde{O}(m\sqrt{n}\log^{O(1)}(U))$, to solve the minimum cost flow problem in time, $\widetilde{O}(m\sqrt{n}\log^{O(1)}(U))$, and to compute ε-approximate solutions to the lossy generalized minimum cost flow problem in time $\widetilde{O}(m\sqrt{n}\log^{O(1)}(U/\varepsilon))$. Our algorithm for the generalized minimum cost flow problem is essentially the same as our algorithm for the simpler specific case of minimum cost flow and maximum flow and therefore, we present the algorithm for the generalized minimum cost flow problem directly.

The generalized minimum cost flow problem [63] is as follows. Let $G = (V, E)$ be a connected directed graph where each edge e has capacity $c_e > 0$ and multiplier $1 \geq \gamma_e > 0$. For each edge e, there can be only at most c_e units of flow on that edge and the flow on that edge must be non-negative. Also, for each unit of flow entering edge e, there are only γ_e units of flow going out. The generalized maximum flow problem is to compute how much flow can be sent into t given a unlimited source s. The generalized minimum cost flow is to ask what is the minimum cost of sending the maximum flow given the cost of each edge is q_e. The maximum flow and the minimum cost flow are the case with $\gamma_e = 1$ for all edges e.

Since the generalized minimum cost flow includes all of these cases, we focus on this general formulation. The problem can be written as the following linear program

$$\min_{\mathbf{0} \leq \boldsymbol{x} \leq \boldsymbol{c}} \boldsymbol{q}^T \boldsymbol{x} \text{ such that } \boldsymbol{A}\boldsymbol{x} = F\mathbf{1}_t$$

where F is the generalized maximum flow value, $\mathbf{1}_t$ is a indicator vector of size $(n-1)$ that is non-zero at vertices t and $\boldsymbol{A}$ is a $|V \backslash \{s\}| \times |E|$ matrix such that for each edge e, we have

$$\begin{aligned} \boldsymbol{A}(e_{head}, e) &= \gamma(e), \\ \boldsymbol{A}(e_{tail}, e) &= -1. \end{aligned}$$

In order words, the constraint $\boldsymbol{A}x = F\mathbf{1}_t$ requires the flow to satisfies the flow conversation at all vertices except s and t and requires it flows F unit of flow to t. We assume c_e are integer and γ_e is a rational number. Let U be the maximum of c_e, q_e, the numerator of γ_e and the denominator of γ_e. For the generalized flow problems, getting an efficient exact algorithm is difficult and we aim for approximation algorithms only.

Definition 6.9.1. We call a flow an $\varepsilon-$approximate generalized maximum flow if it is a flow satisfies the flow conservation and the flow value is larger than maximum flow value minus ε. We call a flow is an $\varepsilon-$approximate generalized minimum cost maximum flow if it is an ε-approximate maximum flow and has cost not greater than the minimum cost maximum flow value.

Note that $\text{rank}\,(\boldsymbol{A}) = n - 1$ because the graph is connected and hence our algorithm takes only $\tilde{O}(\sqrt{n}L)$ iterations. Therefore, the problems remaining are to compute L and bound how much time is required to solve the linear systems involved. However, L is large in the most general setting and hence we cannot use the standard theory to say how to get the initial point, how to round to the vertex. Furthermore, the condition number of $\boldsymbol{A}^T\boldsymbol{A}$ can be very bad.

In [63], they used dual path following to solve the generalized minimum cost flow problem with the caveats that the dual polytope is not bounded, the problem of getting the initial flow, the problem of rounding it to the a feasible flow. We use there analysis to formulate the problem in a manner more amenable to our algorithms. Since we are doing the primal path following, we will state a reformulation of the LP slightly different.

Theorem 6.9.2 ([63]). *Given a directed graph G. We can find a new directed graph $\tilde{G}$ with $O(m)$ edges and $O(n)$ vertices in $\tilde{O}(m)$ time such that the modified linear program*

$$\min_{0\leq x_i\leq c_i, 0\leq y_i\leq 4mU^2, 0\leq z_i\leq 4mU^2} \boldsymbol{q}^T\boldsymbol{x} + \frac{256m^5U^5}{\varepsilon^2}\left(\mathbf{1}^T\boldsymbol{y} + \mathbf{1}^T\boldsymbol{z}\right) \text{ such that } \boldsymbol{A}\boldsymbol{x} + \boldsymbol{y} - \boldsymbol{z} = F\mathbf{1}_t$$

satisfies the following conditions:

1. $\boldsymbol{x} = \frac{c}{2}\mathbf{1}$, $\boldsymbol{y} = 2mU^2\mathbf{1} - (\boldsymbol{A}\frac{c}{2}\mathbf{1})^- + F\vec{1}_t$, $z = 2mU^2\mathbf{1} + (\boldsymbol{A}\frac{c}{2}\mathbf{1})^+$ *is an interior point of the linear program.*

2. *Given any $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ such that $\|\boldsymbol{A}\boldsymbol{x} + \boldsymbol{y} - \boldsymbol{z}\|_2 \leq \frac{\varepsilon^2}{128m^2n^2U^3}$ and with cost value within $\frac{\varepsilon^2}{128m^2n^2U^3}$ of the optimum. Then, one can compute an ε-approximate minimum cost maximum flow for graph G in time $\tilde{O}(m)$.*

3. *The linear system of the linear program is well-conditioned, i.e., the condition number of*

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{I} & -\boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{A}^T \\ \boldsymbol{I} \\ -\boldsymbol{I} \end{bmatrix}$$

   is $O(mU)$.

4. *The linear system of the linear program can be solve in nearly linear time, i.e. for any diagonal matrix $\boldsymbol{S}$ with condition number κ and vector b, it takes $\tilde{O}\left(m\log\left(\frac{\kappa U}{\delta}\right)\right)$ time to find x such that*

$$\|x - \boldsymbol{L}^{-1}b\|_{\boldsymbol{L}} \leq \delta\|x\|_{\boldsymbol{L}}$$

   where $\boldsymbol{L} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{I} & -\boldsymbol{I} \end{bmatrix} \boldsymbol{S} \begin{bmatrix} \boldsymbol{A}^T \\ \boldsymbol{I} \\ -\boldsymbol{I} \end{bmatrix}$.

The main difference between what stated in [63] and here is that

1. Our linear program solver can support constraint $l_i \leq x_i \leq u_i$ and hence we do not need to split the flow variable to positive part and negative part.

2. Our linear program solver is primal path following and hence we add the constraint $y_i \leq 4mU^2$ and $z_i \leq 4mU^2$. Since the maximum flow value is at most mU^2, it does not affect the optimal solution of the linear program.

3. We remove the variable $\mathbf{x}_3$ in [63] because the purpose of that is to make the dual polytope is bounded and we do not need it here.

Using the reduction mentioned above, one can obtain the promised generalized minimum cost flow algorithm.

Theorem 6.9.3. *There is a randomized algorithm to compute an $\varepsilon-$approximate generalized minimum cost maximum flow in $\tilde{O}(\sqrt{n}\log^{O(1)}(U/\varepsilon))$ depth $\widetilde{O}(m\sqrt{n}\log^{O(1)}(U/\varepsilon))$ total work (see Definition 6.9.1). Furthermore, there is an algorithm to compute an exact standard minimum cost maximum flow in $\tilde{O}(\sqrt{n}\log^{O(1)}(U))$ depth and $\widetilde{O}(m\sqrt{n}\log^{O(1)}(U))$ total work.*

Proof. Using the reduction above and Theorem 6.8.1, we get an algorithm of generalized minimum cost flow by solving $\widetilde{O}(\sqrt{n})$ linear systems to $\tilde{\mathcal{O}}(1)$ bit accuracy and the condition number of those systems

are poly(mU/ε). In [63], they showed that the linear system involved can be reduced to $\tilde{O}(\log(U/\varepsilon))$ many Laplacian systems and hence we can use a recent nearly linear work polylogarithmic depth Laplacian system solver of Spielman and Peng [223]. In total, it takes $\widetilde{O}(m \log^{O(1)}(\frac{U}{\varepsilon}))$ time to solve each systems.

For the standard minimum cost maximum flow problem, it is known that the solution set is a convex polytope with integer coordinates and we can use Isolation lemma to make sure there is unique minimum. Hence, we only need to take $\varepsilon = \text{poly}(1/mU)$ and round the solution to the closest integer. See Section 3.5 in [63] for details. □

6.10 Glossary

Here we summarize various linear programming specific notation that we use throughout the chapter. For many quantities we included the typical order of magnitude as they appear during our algorithms.

- Linear program related: constraint matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, cost vector $\boldsymbol{c} \in \mathbb{R}^m$, constraint vector $\boldsymbol{b} \in \mathbb{R}^n$, solution $\boldsymbol{x} \in \mathbb{R}^m$, weights of constraints $\boldsymbol{w} \in \mathbb{R}^m$ where m is the number of variables and n is the number of constraints.
- Matrix version of variables: $\boldsymbol{S}$ is the diagonal matrix corresponds to $\boldsymbol{s}$, $\boldsymbol{W}$ corresponds to $\boldsymbol{w}$, $\boldsymbol{\Phi}$ corresponds to ϕ.
- Penalized objective function (6.12): $f_t(\boldsymbol{x}, \boldsymbol{w}) = t \cdot \boldsymbol{c}^T \boldsymbol{x} + \sum_{i \in [m]} \boldsymbol{w}_i \phi_i(\boldsymbol{x}_i)$.
- Barrier functions (Sec 6.5.1): For $[l, \infty)$, we use $\phi(x) = -\log(x - l)$. For $(-\infty, u]$, we use $\phi(x) = -\log(u - x)$. For $[l, u]$, we use $\phi(x) = -\log(ax + b)$ where $a = \frac{\pi}{u-l}$ and $b = -\frac{\pi}{2}\frac{u+l}{u-l}$.
- The projection matrix $\boldsymbol{P}_{\boldsymbol{x},\boldsymbol{w}}$ (6.14): $\boldsymbol{P}_{\boldsymbol{x},\boldsymbol{w}} = \boldsymbol{I} - \boldsymbol{W}^{-1}\boldsymbol{A}_x \left(\boldsymbol{A}_x^T \boldsymbol{W}^{-1} \boldsymbol{A}_x\right)^{-1} \boldsymbol{A}_x^T$ where $\boldsymbol{A}_x \stackrel{\text{def}}{=} \boldsymbol{\Phi}''(\boldsymbol{x})^{-1/2}\boldsymbol{A}$.
- Newton step (6.13): $\boldsymbol{h}_t(\boldsymbol{x}, \boldsymbol{w}) = -\boldsymbol{\Phi}''(\boldsymbol{x})^{-1/2} \boldsymbol{P}_{\boldsymbol{x},\boldsymbol{w}} \boldsymbol{W}^{-1} \boldsymbol{\Phi}''(\boldsymbol{x})^{-1/2} \nabla_x f_t(\boldsymbol{x}, \boldsymbol{w})$.
- The mixed norm (6.15): $\|\boldsymbol{y}\|_{\boldsymbol{w}+\infty} = \|\boldsymbol{y}\|_\infty + C_{\text{norm}} \|\boldsymbol{y}\|_{\boldsymbol{W}}$ where $C_{\text{norm}} \approx \text{polylog}(m)$.
- Centrality (6.17): $\delta_t(\boldsymbol{x}, \boldsymbol{w}) = \min_{\boldsymbol{\eta} \in \mathbb{R}^n} \left\| \frac{\nabla_x f_t(\boldsymbol{x},\boldsymbol{w}) - \boldsymbol{A}\boldsymbol{\eta}}{\boldsymbol{w}\sqrt{\phi''(\boldsymbol{x})}} \right\|_{\boldsymbol{w}+\infty} \approx \frac{1}{\text{polylog}(m)}$.
- Properties of weight function (Def 6.5.4): **size** $c_1(\boldsymbol{g}) = \|\boldsymbol{g}(\boldsymbol{x})\|_1 \approx \text{rank}\,(\boldsymbol{A})$, **slack sensitivity** $c_\gamma(\boldsymbol{g}) = \|\boldsymbol{P}_{\boldsymbol{x},\boldsymbol{w}}\|_{\boldsymbol{w}+\infty} \approx 1 + \frac{1}{\text{polylog}(m)}$, **step consistency** $c_\delta(\boldsymbol{g}) \approx 1 - \frac{1}{\text{polylog}(m)}$.
- Difference between $\boldsymbol{g}$ and $\boldsymbol{w}$ (6.29): $\boldsymbol{\Psi}(\boldsymbol{x}, \boldsymbol{w}) = \log(\boldsymbol{g}(\boldsymbol{x})) - \log(\boldsymbol{w})$.
- Potential function for tracing 0 (Thm 6.6.6): $\Phi_\mu(\boldsymbol{x}) = e^{\mu x} + e^{-\mu x} \approx \text{poly}(m)$.
- The weight function proposed (6.30):

$$\boldsymbol{g}(\boldsymbol{x}) = \arg\min_{\boldsymbol{w} \in \mathbb{R}^m_{>0}} \hat{f}(\boldsymbol{x}, \boldsymbol{w}) \quad \text{where} \quad \hat{f}(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{1}^T \boldsymbol{w} + \frac{1}{\alpha} \log\det\left(\boldsymbol{A}_x^T \boldsymbol{W}^{-\alpha} \boldsymbol{A}_x\right) - \beta \sum_i \log w_i$$

where $\boldsymbol{A}_x = (\boldsymbol{\Phi}''(\boldsymbol{x}))^{-1/2}\boldsymbol{A}$, $\alpha \approx 1 + 1/\log_2\left(\frac{m}{\text{rank}(\boldsymbol{A})}\right)$, $\beta \approx \text{rank}(\boldsymbol{A})/m$.

6.11 Appendix: Technical Lemmas

Lemma 6.11.1. *For any norm* $\|\cdot\|$ *and* $\|\boldsymbol{y}\|_Q \stackrel{\text{def}}{=} \min_{\boldsymbol{\eta}\in\mathbb{R}^n} \left\|\boldsymbol{y} - \frac{\boldsymbol{A}\boldsymbol{\eta}}{\boldsymbol{w}\sqrt{\vec{\phi}''(\boldsymbol{x})}}\right\|$, *we have*

$$\|\boldsymbol{y}\|_Q \leq \|\boldsymbol{P}_{\boldsymbol{x},\boldsymbol{w}}\boldsymbol{y}\| \leq \|\boldsymbol{P}_{\boldsymbol{x},\boldsymbol{w}}\| \cdot \|\boldsymbol{y}\|_Q \,.$$

Proof. By definition $\boldsymbol{P}_{\boldsymbol{x},\boldsymbol{w}}\boldsymbol{y} = \boldsymbol{y} - \frac{\boldsymbol{A}\boldsymbol{\eta}_y}{\boldsymbol{w}\sqrt{\vec{\phi}''(\boldsymbol{x})}}$ for some $\boldsymbol{\eta}_y \in \mathbb{R}^n$. Consequently,

$$\|\boldsymbol{y}\|_Q \;=\; \min_{\boldsymbol{\eta}\in\mathbb{R}^n} \left\|\boldsymbol{y} - \frac{\boldsymbol{A}\eta}{\boldsymbol{w}\sqrt{\vec{\phi}''(\boldsymbol{x})}}\right\| \leq \|\boldsymbol{P}_{\boldsymbol{x},\boldsymbol{w}}\boldsymbol{y}\| \,.$$

On the other hand, let $\boldsymbol{\eta}_q$ such that $\|\boldsymbol{y}\|_Q = \left\|\boldsymbol{y} - \frac{\boldsymbol{A}\boldsymbol{\eta}_q}{\boldsymbol{w}\sqrt{\vec{\phi}''(\boldsymbol{x})}}\right\|$. Since $\boldsymbol{P}_{\boldsymbol{x},\boldsymbol{w}}\boldsymbol{W}^{-1}(\boldsymbol{\Phi}'')^{-1/2}\boldsymbol{A} = \boldsymbol{0}$, we have

$$\|\boldsymbol{P}_{\boldsymbol{x},\boldsymbol{w}}\boldsymbol{y}\| \;=\; \left\|\boldsymbol{P}_{\boldsymbol{x},\boldsymbol{w}}\left(\boldsymbol{y} - \frac{\boldsymbol{A}\boldsymbol{\eta}_q}{\boldsymbol{w}\sqrt{\vec{\phi}''}}\right)\right\| \leq \|\boldsymbol{P}_{\boldsymbol{x},\boldsymbol{w}}\| \cdot \left\|\boldsymbol{y} - \frac{\boldsymbol{A}\boldsymbol{\eta}_q}{\boldsymbol{w}\sqrt{\vec{\phi}''}}\right\| = \|\boldsymbol{P}_{\boldsymbol{x},\boldsymbol{w}}\| \cdot \|\boldsymbol{y}\|_Q \,.$$

□

Lemma 6.11.2. *For any projection matrix* $\boldsymbol{P} \in \mathbb{R}^{m\times m}$, $\boldsymbol{\Sigma} = \boldsymbol{Diag}(\boldsymbol{P})$, $i,j \in [m]$, $\boldsymbol{x} \in \mathbb{R}^m$, *and* $\boldsymbol{w} \in \mathbb{R}^m_{>0}$ *we have*

1. $\boldsymbol{\Sigma}_{ii} = \sum_{j\in[m]} \boldsymbol{P}^{(2)}_{ij}$,
2. $\boldsymbol{0} \preceq \boldsymbol{P}^{(2)} \preceq \boldsymbol{\Sigma} \preceq \boldsymbol{I}$,
3. $\boldsymbol{P}^{(2)}_{ij} \leq \boldsymbol{\Sigma}_{ii}\boldsymbol{\Sigma}_{jj}$,
4. $|\boldsymbol{1}_i^T\boldsymbol{P}^{(2)}\boldsymbol{x}| \leq \boldsymbol{\Sigma}_{ii}\|\boldsymbol{x}\|_{\boldsymbol{\Sigma}}$.
5. $\nabla_{\boldsymbol{w}} \log\det(\boldsymbol{A}^T\boldsymbol{W}\boldsymbol{A}) = \boldsymbol{\Sigma}_{\boldsymbol{A}}(\boldsymbol{w})\boldsymbol{w}^{-1}$.
6. $\boldsymbol{J}_{\boldsymbol{w}}(\boldsymbol{\sigma}_{\boldsymbol{A}}(\boldsymbol{w})) = \boldsymbol{\Lambda}_{\boldsymbol{A}}(\boldsymbol{w})\boldsymbol{W}^{-1}$.

Proof. To prove (1), we simply note that by definition of a projection matrix $\boldsymbol{P} = \boldsymbol{P}\boldsymbol{P}$ and therefore

$$\boldsymbol{\Sigma}_{ii} = \boldsymbol{P}_{ii} = \boldsymbol{1}_i^T\boldsymbol{P}\boldsymbol{1}_i = \boldsymbol{1}_i^T\boldsymbol{P}\boldsymbol{P}\boldsymbol{1}_i = \sum_{j\in[n]} \boldsymbol{P}^2_{ij} = \sum_{j\in[n]} \boldsymbol{P}^{(2)}_{ij}$$

To prove (2), we observe that since $\boldsymbol{P}$ is a projection matrix, all its eigenvectors are either 0 or 1. Therefore, $\boldsymbol{\Sigma} \preceq \boldsymbol{I}$ and by (1) $\boldsymbol{\Sigma} - \boldsymbol{P}^{(2)}$ is diagonally dominant. Consequently, $\boldsymbol{\Sigma} - \boldsymbol{P}^{(2)} \succeq 0$. Rearranging terms and using the well known fact that the shur product of two positive semi-definite matrices is positive semi-definite yields (2).

To prove (3), we use $\boldsymbol{P} = \boldsymbol{P}\boldsymbol{P}$, Cauchy-Schwarz, and (1) to derive

$$\boldsymbol{P}_{ij} = \sum_{k\in[n]} \boldsymbol{P}_{ik}\boldsymbol{P}_{kj} \leq \sqrt{\left(\sum_{k\in[n]} \boldsymbol{P}^2_{ik}\right)\left(\sum_{k\in[n]} \boldsymbol{P}^2_{kj}\right)} = \sqrt{\boldsymbol{\Sigma}_{ii}\boldsymbol{\Sigma}_{jj}} \quad .$$

Squaring then yields (3).

To prove (4), we note that by the definition of $\boldsymbol{P}^{(2)}$ and Cauchy-Schwarz, we have

$$\left|\mathbf{1}_i^T \boldsymbol{P}^{(2)} \boldsymbol{x}\right| = \left|\sum_{j\in[n]} \boldsymbol{P}_{ij}^{(2)} \vec{x}_j\right| \leq \sqrt{\left(\sum_{j\in[n]} \boldsymbol{\Sigma}_{jj} \vec{x}_j^2\right) \cdot \sum_{j\in[n]} \frac{\boldsymbol{P}_{ij}^{(4)}}{\boldsymbol{\Sigma}_{jj}}} \tag{6.42}$$

Now, by (1) and (3), we know that

$$\sum_{j\in[n]} \frac{\boldsymbol{P}_{ij}^4}{\boldsymbol{\Sigma}_{jj}} \leq \sum_{j\in[n]} \frac{\boldsymbol{P}_{ij}^2 \boldsymbol{\Sigma}_{ii} \boldsymbol{\Sigma}_{jj}}{\boldsymbol{\Sigma}_{jj}} = \boldsymbol{\Sigma}_{ii} \sum_{j\in[n]} \boldsymbol{P}_{ij}^2 = \boldsymbol{\Sigma}_{ii}^2 \tag{6.43}$$

Since $\|\boldsymbol{x}\|_{\boldsymbol{\Sigma}} \stackrel{\text{def}}{=} \sqrt{\sum_{j\in[n]} \boldsymbol{\Sigma}_{jj} \boldsymbol{x}_j^2}$, combining (6.42) and (6.43) yields $\left|\mathbf{1}_i^T \boldsymbol{P}^{(2)} \boldsymbol{x}\right| \leq \boldsymbol{\Sigma}_{ii} \|\boldsymbol{x}\|_{\boldsymbol{\Sigma}}$ as desired.

To prove (5), we let $f(\boldsymbol{w}) = \log\det(\boldsymbol{A}^T \boldsymbol{W} \boldsymbol{A})$. Lemma 2.3.8 shows that

$$\begin{aligned} \frac{\partial}{\partial w_i} f(\boldsymbol{w}) &= \operatorname{Tr}\left(\left(\boldsymbol{A}^T \boldsymbol{W} \boldsymbol{A}\right)^{-1} \boldsymbol{a}_i \boldsymbol{a}_i^T\right) \\ &= \boldsymbol{\Sigma}_{\boldsymbol{A}}(\boldsymbol{w})_{ii} w_i^{-1}. \end{aligned}$$

To prove (6), we let $\boldsymbol{R}_{\boldsymbol{A}}(\boldsymbol{w})_{ij} \stackrel{\text{def}}{=} \boldsymbol{a}_i^T (\boldsymbol{A}^T \boldsymbol{W} \boldsymbol{A})^{-1} \boldsymbol{a}_j$. Lemma 2.3.8 shows that

$$\frac{\partial}{\partial w_i} (\boldsymbol{A}^T \boldsymbol{W} \boldsymbol{A})^{-1} = -(\boldsymbol{A}^T \boldsymbol{W} \boldsymbol{A})^{-1} \boldsymbol{a}_i \boldsymbol{a}_i^T (\boldsymbol{A}^T \boldsymbol{W} \boldsymbol{A})^{-1}$$

and hence $\frac{\partial}{\partial \boldsymbol{w}_k} [\boldsymbol{R}_{\boldsymbol{A}}(\boldsymbol{w})]_{ij} = -\boldsymbol{R}_{\boldsymbol{A}}(\boldsymbol{w})_{ik} \boldsymbol{R}_{\boldsymbol{A}}(\boldsymbol{w})_{kj}$. Since by definition $\boldsymbol{\sigma}_{\boldsymbol{A}}(\boldsymbol{w})_i = \boldsymbol{w}_i \boldsymbol{R}_{\boldsymbol{A}}(\boldsymbol{w})_{ii}$ by the previous lemma, we have that

$$\frac{\partial}{\partial w_j} \boldsymbol{\sigma}_{\boldsymbol{A}}(\boldsymbol{w})_i = \mathbf{1}_{i=j} \boldsymbol{R}_{\boldsymbol{A}}(\boldsymbol{w})_{ii} - w_i \boldsymbol{R}_{\boldsymbol{A}}(\boldsymbol{w})_{ij}^{(2)}.$$

Writing this in matrix form and recalling the definition of the Jacobian then yields

$$\boldsymbol{J}_{\boldsymbol{w}}(\boldsymbol{\sigma}_{\boldsymbol{A}}(\boldsymbol{w})) = \boldsymbol{Diag}(\boldsymbol{R}_{\boldsymbol{A}}(\boldsymbol{w})) - \boldsymbol{W} \boldsymbol{R}_{\boldsymbol{A}}(\boldsymbol{w})^{(2)}.$$

Right multiplying by $\boldsymbol{I} = \boldsymbol{W} \boldsymbol{W}^{-1}$ and recalling the definition of $\boldsymbol{\Lambda}_{\boldsymbol{A}}$ then yields the result. □

Lemma 6.11.3. *For any x, ε and $\lambda > 0$, we have*

$$\frac{\pi}{2} |x| - \pi\varepsilon - \frac{1}{\lambda} \leq x \tan^{-1}(\lambda(x+\varepsilon)) \leq \frac{\pi}{2} |x|.$$

Proof. We first consider the case $\varepsilon = 0$. Note that

$$x \tan^{-1}(\lambda x) \leq \frac{\pi}{2} |x|.$$

Also, we note that

$$x \tan^{-1}(\lambda x) \geq |x| \left(\frac{\pi}{2} - \frac{1}{\lambda |x|}\right)$$

because

$$\left|\tan\left(\frac{\pi}{2} - \frac{1}{\lambda |x|}\right)\right| = \left|\frac{\cos(\frac{1}{\lambda|x|})}{\sin(\frac{1}{\lambda|x|})}\right| \leq \lambda |x|.$$

Hence, we have

$$\frac{\pi}{2}|x| - \frac{1}{\lambda} \leq x\tan^{-1}(\lambda x) \leq \frac{\pi}{2}|x|.$$

For $\varepsilon \neq 0$, we have

$$\frac{\pi}{2}|x+\varepsilon| - \frac{1}{\lambda} \leq (x+\varepsilon)\tan^{-1}(\lambda(x+\varepsilon)) \leq \frac{\pi}{2}|x+\varepsilon|.$$

Thus, we have

$$\frac{\pi}{2}|x| - \pi\varepsilon - \frac{1}{\lambda} \leq x\tan^{-1}(\lambda(x+\varepsilon)) \leq \frac{\pi}{2}|x|.$$

□

■ 6.12 Appendix: Projection on Mixed Norm Ball

In this section, we give an algorithm to solve the problem

$$\max_{\|\boldsymbol{x}\|_2+\|\boldsymbol{l}^{-1}\boldsymbol{x}\|_\infty \leq 1} \langle \boldsymbol{a}, \boldsymbol{x}\rangle \tag{6.44}$$

for some given vector $\boldsymbol{l}$ and $\boldsymbol{a}$. This is used in Section 6.7.1 to compute the weights. We note that

$$\begin{aligned}
\max_{\|\boldsymbol{x}\|_2+\|\boldsymbol{l}^{-1}\boldsymbol{x}\|_\infty \leq 1} \langle \boldsymbol{a}, \boldsymbol{x}\rangle &= \max_{0\leq t\leq 1} \max_{\|\boldsymbol{x}\|_2\leq 1-t \text{ and } -tl_i\leq x_i\leq tl_i} \langle \boldsymbol{a}, \boldsymbol{x}\rangle \\
&= \max_{0\leq t\leq 1} (1-t) \max_{\|\boldsymbol{x}\|_2\leq 1 \text{ and } -\frac{t}{1-t}l_i\leq x_i\leq \frac{t}{1-t}l_i} \langle \boldsymbol{a}, \boldsymbol{x}\rangle \\
&= \max_{0\leq t\leq 1} (1-t) f(\frac{t}{1-t}) \qquad (6.45)
\end{aligned}$$

where

$$f(\frac{t}{1-t}) = \max_{\|\boldsymbol{x}\|_2\leq 1, -tl_i\leq x_i\leq tl_i} \langle \boldsymbol{a}, \boldsymbol{x}\rangle.$$

In [165], we showed that the maximizer $\boldsymbol{x}$ in the problem f must be of the form

$$\boldsymbol{x}^{i_t} = \begin{cases} \frac{t}{1-t}\text{sign}\,(a_j)\,l_j & \text{if } j \in \{1,2,\cdots,i_t\} \\ \sqrt{\frac{1-\left(\frac{t}{1-t}\right)^2\sum_{k=0}^{i_t} l_k^2}{1-\sum_{k=0}^{i_t} a_k^2}}\boldsymbol{a}_j & \text{otherwise} \end{cases}. \tag{6.46}$$

where i_t be the first coordinate such that

$$\frac{1-\left(\frac{t}{1-t}\right)^2\sum_{k=0}^{i} l_k^2}{1-\sum_{k=0}^{i} a_k^2} \leq \frac{\left(\frac{t}{1-t}\right)^2 l_i^2}{a_i^2}.$$

Note that $i_t \geq i_s$ if $t \leq s$. Therefore, we have that the set of t such that $i_t = j$ is simply an interval given by[8]

$$\frac{|a_j|}{\sqrt{l_j^2\left(1-\sum_{k=0}^{j} a_k^2\right) + a_j^2\sum_{k=0}^{j} l_k^2}} \leq \frac{t}{1-t} < \frac{|a_{j-1}|}{\sqrt{l_{j-1}^2\left(1-\sum_{k=0}^{j-1} a_k^2\right) + a_{j-1}^2\sum_{k=0}^{j-1} l_k^2}}. \tag{6.47}$$

[8] There are some boundary cases we ignored for simplicity.

Therefore, we know that

$$\begin{aligned} f(\frac{t}{1-t}) &= \left\langle \boldsymbol{a}, \boldsymbol{x}^{(i_t)} \right\rangle \\ &= \frac{t}{1-t} \sum_{j=1}^{i_t} |a_j| |l_j| + \sqrt{1 - \left(\frac{t}{1-t}\right)^2 \sum_{k=0}^{i_t} l_k^2} \sqrt{1 - \sum_{k=0}^{i_t} a_k^2}. \end{aligned}$$

Putting this into 6.45, we have that

$$\max_{\|\boldsymbol{x}\|_2 + \|\boldsymbol{l}^{-1}\boldsymbol{x}\|_\infty \leq 1} \langle \boldsymbol{a}, \boldsymbol{x} \rangle = \max_{0 \leq t \leq 1} t \sum_{j=1}^{i_t} |a_j| |l_j| + \sqrt{(1-t)^2 - t^2 \sum_{k=0}^{i_t} l_k^2} \sqrt{1 - \sum_{k=0}^{i_t} a_k^2}.$$

Note that the function $t \sum_{j=1}^{i} |a_j| |l_j| + \sqrt{(1-t)^2 - t^2 \sum_{k=0}^{i} l_k^2} \sqrt{1 - \sum_{k=0}^{i} a_k^2}$ is concave and the solution has a close form. Therefore, one can compute the maximum value for each interval of t (6.47) and find which is the best. Hence, we get the following algorithm.

$\boldsymbol{x} =$ `projectOntoMixedNormBallParallel`$(\boldsymbol{a}, \boldsymbol{l})$
1. Set $\boldsymbol{a} = \boldsymbol{a}/\|\boldsymbol{a}\|_2$.
2. Sort the coordinate such that $\|a_i\| / l_i$ is in descending order.
3. Precompute $\sum_{k=0}^{i} l_k^2$, $\sum_{k=0}^{i} a_k^2$ and $\sum_{j=1}^{i} \|a_j\| \|l_j\|$ for all i.
4. Let $g_i(t) = t \sum_{j=1}^{i} \|a_j\| \|l_j\| + \sqrt{(1-t)^2 - t^2 \sum_{k=0}^{i} l_k^2} \sqrt{1 - \sum_{k=0}^{i} a_k^2}$.
5. For each $j \in \{1, \cdots, n\}$, Find $t_j = \arg\max_{i_t = j} g_j(t)$ using (6.47)
6. Find $i = \arg\max_i g_i(t_i)$.
7. Output $(1 - t_i)\boldsymbol{x}^{(i)}$ defined by 6.46.

The discussion above leads to the following theorem.

Theorem 6.12.1. *The algorithm* `projectOntoMixedNormBallParallel` outputs a solution to

$$\max_{\|\boldsymbol{x}\|_2 + \|\boldsymbol{l}^{-1}\boldsymbol{x}\|_\infty \leq 1} \langle \boldsymbol{a}, \boldsymbol{x} \rangle$$

in total work $\tilde{O}(m)$ and depth $\tilde{O}(1)$.

6.13 Appendix: A $\tilde{O}(n)$-Self-Concordant Barrier Function

In this section, we proved the barrier function introduced in Subsection 6.3.5 is an $\tilde{O}(n)$-self-concordant barrier. We separate the proof of Theorem 6.3.5 into two lemma, Lemma 6.13.3 and Lemma 6.13.8. The proof is pretty straightforward, but it requires some familiarity with matrix calculus.

6.13.1 Notation

Here, we define some notations that is used throughout in this section. They different from previous sections because this section our domain is $\Omega \stackrel{\text{def}}{=} \{\boldsymbol{A}\boldsymbol{x} > \boldsymbol{b}\}$ instead of $\boldsymbol{A}^\top \boldsymbol{x} = \boldsymbol{b}$.

1. Let $f(x, w) = \ln\det(\boldsymbol{A}^T \boldsymbol{S}^{-1} \boldsymbol{W} \boldsymbol{S}^{-1} \boldsymbol{A}) - \frac{n}{m} \sum_{i=1}^{m} w_i \ln w_i - \frac{n}{m} \sum_{i=1}^{m} \ln s_i$.
2. Let $\boldsymbol{S} = \boldsymbol{Diag}\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}$ and $\boldsymbol{A}_x = \boldsymbol{S}^{-1}\boldsymbol{A}$.

3. Let $\boldsymbol{P}(x,w) = \sqrt{\boldsymbol{W}}\boldsymbol{A}_x \left(\boldsymbol{A}_x^T \boldsymbol{W} \boldsymbol{A}_x\right)^{-1} \boldsymbol{A}_x^T \sqrt{\boldsymbol{W}}$, $\sigma(x,w) = \boldsymbol{diag}\left(\boldsymbol{P}(x,w)\right)$.

4. Let $\boldsymbol{P}^{(2)}(x,w)$ be a $m \times m$ matrix such that $\left(\boldsymbol{P}^{(2)}(x,w)\right)_{ij} = (\boldsymbol{P}(x,w))_{ij}^2$.

5. Let $\boldsymbol{\Lambda}(x,w) = \boldsymbol{\Sigma}(x,w) - \boldsymbol{P}^{(2)}(x,w)$.

6. Let w_x be the unique maximizer of $f(x,w)$, i.e. $w_x \stackrel{\text{def}}{=} \arg\max_{w_i \geq 0} f(x,w)$.

7. Let $\boldsymbol{P}_x = \boldsymbol{P}(x,w)$, $\sigma_x = \sigma(x,w_x)$, $\boldsymbol{\Sigma}_x = \boldsymbol{\Sigma}(x,w_x)$, $\boldsymbol{P}_x^{(2)} = \boldsymbol{P}^{(2)}(x,w_x)$ and $\boldsymbol{\Lambda}_x = \boldsymbol{\Lambda}(x,w_x)$.

Similar to lemma 6.11.2, we have $\boldsymbol{0} \preceq \boldsymbol{P}^{(2)}(x,w) \preceq \boldsymbol{\Sigma}(x,w) \preceq \boldsymbol{I}$, $\boldsymbol{0} \preceq \boldsymbol{\Lambda}(x,w)$ and $\sum_i \sigma_i(x,w) = n$.

6.13.2 Basic properties of f and p

First, we compute the gradient and Hessian of f and p.

Lemma 6.13.1. *For any interior point $x \in \Omega$ and any positive weight $w > 0$, we have*

$$\begin{aligned}
\nabla_x f(x,w) &= -2\boldsymbol{A}_x^T \sigma(x,w) - \frac{n}{m}\boldsymbol{A}_x^T 1, \\
\nabla_w f(x,w) &= \boldsymbol{W}^{-1}\sigma(x,w) - \frac{n}{m}\ln(ew), \\
\nabla_{xx}^2 f(x,w) &= \boldsymbol{A}_x^T \left(6\boldsymbol{\Sigma}(x,w) - 4\boldsymbol{P}^{(2)}(x,w) + \frac{n}{m}\boldsymbol{I}\right)\boldsymbol{A}_x, \\
\nabla_{ww}^2 f(x,w) &= -\boldsymbol{W}^{-1}\boldsymbol{P}^{(2)}(x,w)\boldsymbol{W}^{-1} - \frac{n}{m}\boldsymbol{W}^{-1}, \\
\nabla_{xw}^2 f(x,w) &= -2\boldsymbol{A}_x^T \left(\boldsymbol{\Sigma}(x,w) - \boldsymbol{P}^{(2)}(x,w)\right)\boldsymbol{W}^{-1}.
\end{aligned} \tag{6.48}$$

For $x \in \Omega$, w_x uniquely exist in the region $\left\{\frac{1}{e} \leq w_i \leq \frac{m}{n}\right\}$ and p satisfies

$$\begin{aligned}
\nabla p(x) &= -2\boldsymbol{A}_x^T \sigma_x - \frac{n}{m}\boldsymbol{A}_x^T 1. \\
\nabla^2 p(x) &= \boldsymbol{A}_x^T \left(6\boldsymbol{\Sigma}_x - 4\boldsymbol{P}_x^{(2)} + \frac{n}{m}I\right)\boldsymbol{A}_x + 4\boldsymbol{A}_x^T \boldsymbol{\Lambda}_x \left(\boldsymbol{P}_x^{(2)} + \frac{n}{m}\boldsymbol{W}_x\right)^{-1} \boldsymbol{\Lambda}_x \boldsymbol{A}_x.
\end{aligned} \tag{6.49}$$

Also, we have

$$\boldsymbol{J}_x w = -2\boldsymbol{W}\left(\boldsymbol{P}^{(2)}(x,w) + \frac{n}{m}\boldsymbol{W}\right)^{-1}\left(\boldsymbol{\Sigma}(x,w) - \boldsymbol{P}^{(2)}(x,w)\right)\boldsymbol{A}_x. \tag{6.50}$$

Proof. For $\nabla_x f$, we have

$$\begin{aligned}
\frac{\partial f}{\partial s_i} &= \mathrm{Tr}\left(\left(\boldsymbol{A}^T\boldsymbol{S}^{-1}\boldsymbol{W}\boldsymbol{S}^{-1}\boldsymbol{A}\right)^{-1}\left(\frac{\partial}{\partial s_i}\right)\left(\boldsymbol{A}^T\boldsymbol{S}^{-1}\boldsymbol{W}\boldsymbol{S}^{-1}\boldsymbol{A}\right)\right) - \frac{n}{m}\frac{1}{s_i} \\
&= -2\mathrm{Tr}\left(\boldsymbol{A}\left(\boldsymbol{A}^T\boldsymbol{S}^{-1}\boldsymbol{W}\boldsymbol{S}^{-1}\boldsymbol{A}\right)^{-1}\boldsymbol{A}^T\frac{w_i}{s_i^3}1_i 1_i^T\right) - \frac{n}{m}\frac{1}{s_i} \\
&= -2\frac{\sigma_i(x,w)}{s_i} - \frac{n}{m}\frac{1}{s_i}
\end{aligned}$$

By Chain rule, we have

$$\nabla_x f(x,w) = -2\boldsymbol{A}_x^T\sigma(x,w) - \frac{n}{m}\boldsymbol{A}_x^T 1.$$

For $\nabla_w f$, we have

$$\begin{aligned}\frac{\partial f}{\partial w_i} &= \mathrm{Tr}\left(\left(\boldsymbol{A}^T\boldsymbol{S}^{-1}\boldsymbol{W}\boldsymbol{S}^{-1}\boldsymbol{A}\right)^{-1}\left(\frac{\partial}{\partial w_i}\right)\left(\boldsymbol{A}^T\boldsymbol{S}^{-1}\boldsymbol{W}\boldsymbol{S}^{-1}\boldsymbol{A}\right)\right) - \frac{n}{m}\ln w_i - \frac{n}{m} \\ &= \frac{\sigma_i(x,w)}{w_i} - \frac{n}{m}\ln(ew_i).\end{aligned}$$

Hence, we have

$$\nabla_w f(x,w) = \boldsymbol{W}^{-1}\sigma - \frac{n}{m}\ln(ew).$$

For $\nabla_{xx} f$, taking partial derivative on $\frac{\partial f}{\partial s_i}$, we have

$$\begin{aligned}\frac{\partial^2 f}{\partial s_i \partial s_j} &= \frac{\partial}{\partial s_j}\left(-2\mathrm{Tr}\left(\boldsymbol{A}\left(\boldsymbol{A}^T\boldsymbol{S}^{-1}\boldsymbol{W}\boldsymbol{S}^{-1}\boldsymbol{A}\right)^{-1}\boldsymbol{A}^T\frac{w_i}{s_i^3}1_i 1_i^T\right) - \frac{n}{m}\frac{1}{s_i}\right) \\ &= -4\mathrm{Tr}\left(\boldsymbol{A}\left(\boldsymbol{A}_x^T\boldsymbol{W}\boldsymbol{A}_x\right)^{-1}\boldsymbol{A}^T\frac{w_j}{s_j^3}1_j 1_j^T\boldsymbol{A}\left(\boldsymbol{A}_x^T\boldsymbol{W}_x\boldsymbol{A}\right)^{-1}\boldsymbol{A}^T\frac{w_i}{s_i^3}1_i 1_i^T\right) \\ &\quad 6\mathrm{Tr}\left(\boldsymbol{A}\left(\boldsymbol{A}_x^T\boldsymbol{W}\boldsymbol{A}_x\right)^{-1}\boldsymbol{A}^T\frac{w_i}{s_i^4}1_i 1_i^T 1_{i=j}\right) + \frac{n}{m}\frac{1}{s_i^2}1_{i=j} \\ &= 6\frac{\sigma_i(x,w)}{s_i^2}1_{i=j} - 4\frac{(\boldsymbol{P}(x,w))_{ij}^2}{s_i s_j} + \frac{n}{m}\frac{1}{s_i^2}1_{i=j}.\end{aligned}$$

By Chain rule, we have

$$\nabla_{xx} f(x,w) = \boldsymbol{A}_x^T\left(6\boldsymbol{\Sigma}(x,w) - 4\boldsymbol{P}^{(2)}(x,w) + \frac{n}{m}\boldsymbol{I}\right)\boldsymbol{A}_x.$$

For $\nabla_{ww} f$, taking partial derivative on $\frac{\partial f}{\partial w_i}$, we have

$$\begin{aligned}\frac{\partial^2 f}{\partial w_i \partial w_j} &= \frac{\partial}{\partial w_j}\left(\mathrm{Tr}\left(\boldsymbol{A}_x\left(\boldsymbol{A}_x^T\boldsymbol{W}\boldsymbol{A}_x\right)^{-1}\boldsymbol{A}_x^T e_i e_i^T\right) - \frac{n}{m}\ln w_i - \frac{n}{m}\right) \\ &= -\mathrm{Tr}\left(\boldsymbol{A}_x\left(\boldsymbol{A}_x^T\boldsymbol{W}\boldsymbol{A}_x\right)^{-1}\boldsymbol{A}_x^T e_j e_j^T\boldsymbol{A}_x\left(\boldsymbol{A}_x^T\boldsymbol{W}\boldsymbol{A}_x\right)^{-1}\boldsymbol{A}_x^T e_i e_i^T\right) - \frac{n}{m}\frac{1}{w_j}1_{i=j} \\ &= -\frac{(\boldsymbol{P}(x,w))_{ij}^2}{w_i w_j} - \frac{n}{m}\frac{1}{w_j}1_{i=j}.\end{aligned}$$

Hence, we have

$$\nabla_{ww} f(x,w) = -\boldsymbol{W}^{-1}\boldsymbol{P}^{(2)}(x,w)\boldsymbol{W}^{-1} - \frac{n}{m}\boldsymbol{W}^{-1}.$$

For $\nabla_{xw} f$, taking partial derivative on $\frac{\partial f}{\partial w_i}$, we have

$$\begin{aligned}\frac{\partial^2 f}{\partial s_j \partial w_i} &= \frac{\partial}{\partial s_j}\left(\mathrm{Tr}\left(\boldsymbol{A}\left(\boldsymbol{A}_x^T\boldsymbol{W}\boldsymbol{A}_x\right)^{-1}\boldsymbol{A}^T\frac{e_i e_i^T}{s_i^2}\right) - \frac{n}{m}\ln w_i - \frac{n}{m}\right) \\ &= 2\mathrm{Tr}\left(\boldsymbol{A}\left(\boldsymbol{A}_x^T\boldsymbol{W}\boldsymbol{A}_x\right)^{-1}\boldsymbol{A}^T\frac{w_j e_j e_j^T}{s_j^3}\boldsymbol{A}\left(\boldsymbol{A}_x^T\boldsymbol{W}\boldsymbol{A}_x\right)^{-1}\boldsymbol{A}^T\frac{e_i e_i^T}{s_i^2}\right) \\ &\quad -2\mathrm{Tr}\left(\boldsymbol{A}\left(\boldsymbol{A}_x^T\boldsymbol{W}\boldsymbol{A}_x\right)^{-1}\boldsymbol{A}^T\frac{e_i e_i^T}{s_i^3}1_{i=j}\right) \\ &= 2\frac{(\boldsymbol{P}(x,w))_{ij}}{s_j w_i} - 2\frac{\sigma_i(x,w)}{s_j w_i}1_{i=j}.\end{aligned}$$

By Chain rule, we have

$$\nabla_{xw} f(x,w) = -2\boldsymbol{A}_x^T \left(\boldsymbol{\Sigma}(x,w) - \boldsymbol{P}^{(2)}(x,w)\right) \boldsymbol{W}^{-1}.$$

Now, we prove that w_x uniquely exists in the region $\{\frac{1}{e} \le w_i \le \frac{m}{n}\}$. For any w such that $w_i > \frac{m}{n}$ for some i, we have

$$\begin{aligned}\frac{\partial f}{\partial w_i} &= \frac{\sigma_i(x,w)}{w_i} - \frac{n}{m}\ln(ew_i)\\ &\le \frac{1}{w_i} - \frac{n}{m}\ln(ew_i)\\ &\le \frac{n}{m} - \frac{n}{m}\ln(e\frac{m}{n}) < 0.\end{aligned}$$

Hence, we have $\sup_{w_i \ge 0} f(x,w) = \sup_{\frac{m}{n} \ge w_i \ge 0} f(x,w)$. Since the region $0 \le w_i \le \frac{m}{n}$ is a compact set and f is a continuous function, there is w_x such that attains $\sup_{w_i \ge 0} f(x,w)$. For the uniqueness, we note that $\nabla_{ww} f(x,w) \prec \mathbf{0}$ for all w and hence f is strictly concave and the maximizer is unique. Now, we note that for any w such that $w_i < \frac{1}{e}$ for some i, we have

$$\frac{\partial f}{\partial w_i} = \frac{\sigma_i(x,w)}{w_i} - \frac{n}{m}\ln(ew_i) > 0.$$

Therefore, the minimizer is in the region $\{\frac{1}{e} \le w_i \le \frac{m}{n}\}$.

To compute p, we note that $f(x,w)$ is concave in w and the maximizer is in the interior of the set $\{w_i \ge 0\}$ and therefore the optimality conditions imply that

$$\nabla_w f(x, w_x) = 0. \tag{6.51}$$

Since $\nabla^2_{ww} f(x,w_x)$ is invertible, implicit function theorem (Lemma 2.3.9) shows that w_x is differentiable and satisfies

$$\boldsymbol{J}_x w = -\left(\nabla^2_{ww} f(x,w_x)\right)^{-1}\left(\nabla^2_{wx} f(x,w_x)\right).$$

This gives (6.50).

Using that $p(x) = f(x,w_x)$ and applying the chain rule, we have

$$\begin{aligned}\nabla p(x) &= \nabla_x f(x,w_x) + \nabla_w f(x,w_x)\boldsymbol{J}_x w_x\\ &= \nabla_x f(x,w_x)\end{aligned}$$

because the optimality condition. Taking derivative again, we have

$$\begin{aligned}\nabla^2 p(x) &= \nabla^2_{xx} f(x,w_x) + \nabla^2_{xw} f(x,w_x)\boldsymbol{J}_x w_x\\ &= \nabla^2_{xx} f(x,w_x) - \nabla^2_{xw} f(x,w_x)\left(\nabla^2_{ww} f(x,w_x)\right)^{-1}\left(\nabla^2_{wx} f(x,w_x)\right)\end{aligned}$$

Substituting in the computed values for $\nabla^2_{xx} f(x,w_x)$, $\nabla^2_{xw} f(x,w_x)$, and $\nabla^2_{wx} f(x,w_x)$ then yields the result. □

Now, we discuss properties of the optimal weight w_x.

Lemma 6.13.2. *For $x \in \Omega$, we have $\sigma_x = \frac{n}{m} w_x \ln(ew_x)$. Also, we have $\max_i \frac{\sigma_{x,i}}{w_{x,i}} \le \frac{n}{m}\ln(\frac{me}{n})$.*

Proof. The optimality conditions (6.51) implies

$$\sigma_x = \frac{n}{m} w_x \ln(ew_x).$$

Lemma 6.13.1 shows that $\frac{1}{e} \leq w_x \leq \frac{m}{n}$ and hence

$$\frac{\sigma_{x,i}}{w_{x,i}} \leq \frac{n}{m} \ln(\frac{me}{n}).$$

□

Using these, we can bound $(\nabla p)^T \left(\nabla^2 p\right)^{-1} (\nabla p)$.

Lemma 6.13.3. *For $x \in \Omega$, we have*

$$(\nabla p(x))^T \left(\nabla^2 p(x)\right)^{-1} (\nabla p(x)) \leq 6n.$$

Proof. Using $\nabla p(x) = -2\boldsymbol{A}_x^T \sigma_x - \frac{n}{m} \boldsymbol{A}_x^T 1$, we have

$$\begin{aligned} & (\nabla p(x))^T \left(\nabla^2 p(x)\right)^{-1} (\nabla p(x)) \\ \leq & \ 8\sigma_x^T \boldsymbol{A}_x \left(\nabla^2 p(x)\right)^{-1} \boldsymbol{A}_x^T \sigma_x + 2\left(\frac{n}{m}\right)^2 1^T \boldsymbol{A}_x \left(\nabla^2 p(x)\right)^{-1} \boldsymbol{A}_x^T 1. \end{aligned}$$

For the first term, we use $\boldsymbol{\Lambda}_x \succeq 0$ and $\boldsymbol{P}_x^{(2)} \succeq 0$ and get

$$\begin{aligned} & \sigma_x^T \boldsymbol{A}_x \left(\nabla^2 p(x)\right)^{-1} \boldsymbol{A}_x^T \sigma_x \\ \leq & \ \sigma_x^T \boldsymbol{A}_x \left(\boldsymbol{A}_x^T \left(6\boldsymbol{\Sigma}_x - 4\boldsymbol{P}_x^{(2)} + \frac{n}{m} \boldsymbol{I}_m\right) \boldsymbol{A}_x\right)^{-1} \boldsymbol{A}_x^T \sigma_x \\ \leq & \ \frac{1}{2} \sigma_x^T \boldsymbol{A}_x \left(\boldsymbol{A}_x^T \boldsymbol{\Sigma}_x \boldsymbol{A}_x\right)^{-1} \boldsymbol{A}_x^T \sigma_x \\ \leq & \ \frac{1}{2} \sum_{i=1}^m \sigma_{x,i} \\ = & \ \frac{n}{2}. \end{aligned}$$

For the second term, we have

$$\begin{aligned} & 1_m^T \boldsymbol{A}_x \left(\nabla^2 p(x)\right)^{-1} \boldsymbol{A}_x^T 1_m \\ \leq & \ 1_m^T \boldsymbol{A}_x \left(\boldsymbol{A}_x^T \left(6\boldsymbol{\Sigma}_x - 4\boldsymbol{P}_x^{(2)} + \frac{n}{m} \boldsymbol{I}_m\right) \boldsymbol{A}_x\right)^{-1} \boldsymbol{A}_x^T 1_m \\ \leq & \ \frac{m}{n} 1_m^T \boldsymbol{A}_x \left(\boldsymbol{A}_x^T \boldsymbol{A}_x\right)^{-1} \boldsymbol{A}_x^T 1_m \\ \leq & \ \frac{m^2}{n}. \end{aligned}$$

Combining both terms, we have the result. □

6.13.3 w_x, $\boldsymbol{\Sigma}_x$ and $\boldsymbol{P}_x^{(2)}$ are stable

In this subsection, we write $x_t = x + th$, $\boldsymbol{A}_t = \boldsymbol{A}_{x_t}$, $s_t = \boldsymbol{A}_t x_t - b$, $\boldsymbol{S}_t = \boldsymbol{Diag}(s_t)$, $w_t = w_{x_t}$, $\boldsymbol{W}_t = \boldsymbol{Diag}(w_t)$, $\boldsymbol{P}_t = \boldsymbol{P}(x_t, w_t)$, $\sigma_t = \sigma(x_t, w_t)$, $\boldsymbol{\Sigma}_t = \boldsymbol{\Sigma}(x_t, w_t)$, $\boldsymbol{P}_t^{(2)} = \boldsymbol{P}^{(2)}(x_t, w_t)$ and $\boldsymbol{\Lambda}_t = \boldsymbol{\Lambda}(x_t, w_t)$. First of all, we show that $\boldsymbol{A}_x h$ is small.

Lemma 6.13.4. *For any $x \in \Omega$ and $h \in \mathbb{R}^n$, we have*

$$\|\boldsymbol{A}_x h\|_\infty \leq \frac{1}{\sqrt{2}} \ln\left(\frac{me}{n}\right) \|h\|_{\nabla^2 p(x)}.$$

Therefore, we have

$$\|\boldsymbol{S}_t^{-1}\frac{d}{dt}s_t\|_\infty \le \frac{1}{\sqrt{2}}\ln\left(\frac{me}{n}\right)\|h\|_{\nabla^2 p(x_t)}.$$

Proof. Note that

$$\begin{aligned}\|\boldsymbol{A}_x h\|_\infty &= \max_i \langle e_i, \boldsymbol{A}_x h\rangle \\ &\le \max_i \left\langle (\boldsymbol{A}_x^T\boldsymbol{W}\boldsymbol{A}_x)^{-1/2}\boldsymbol{A}_x^T 1_i, (\boldsymbol{A}_x^T\boldsymbol{W}\boldsymbol{A}_x)^{1/2}h\right\rangle \\ &\le \left(\max_i \frac{\sigma_{x,i}}{w_{x,i}}\right)\|h\|_{\boldsymbol{A}_x^T\boldsymbol{W}\boldsymbol{A}_x}.\end{aligned}$$

Lemma 6.13.2 shows that $\max_i \frac{\sigma_{x,i}}{w_{x,i}} \le \frac{n}{m}\ln\left(\frac{me}{n}\right)$ and hence

$$\|\boldsymbol{A}_x h\|_\infty \le \frac{n}{m}\ln\left(\frac{me}{n}\right)\|h\|_{\boldsymbol{A}_x^T\boldsymbol{W}\boldsymbol{A}_x}.$$

On the other hand, we have

$$\begin{aligned}\nabla^2 p &= \boldsymbol{A}_x^T\left(6\boldsymbol{\Sigma}_x - 4\boldsymbol{P}_x^{(2)} + \frac{n}{m}\boldsymbol{I}\right)\boldsymbol{A}_x + 4\boldsymbol{A}_x^T\boldsymbol{\Lambda}_x\left(\frac{n}{m}\boldsymbol{W}_x + \boldsymbol{P}_x^{(2)}\right)^{-1}\boldsymbol{\Lambda}_x\boldsymbol{A}_x \\ &\succeq \boldsymbol{A}_x^T\left(2\boldsymbol{\Sigma}_x + \frac{n}{m}\boldsymbol{I}\right)\boldsymbol{A}_x\end{aligned}$$

Lemma 6.13.2 shows that $\sigma_x = \frac{n}{m}w\ln(ew)$ and hence

$$2\sigma + \frac{n}{m} = \frac{n}{m}\left(2w\ln(ew) + 1\right).$$

Using $\frac{2w\ln(ew)+1}{w} \ge 4 - \ln(4) \ge 2$ for all $w > 0$, we have

$$\nabla^2 p(x) \succeq 2\boldsymbol{A}_x^T\boldsymbol{W}_x\boldsymbol{A}_x$$

Hence, we have

$$\|\boldsymbol{A}_x h\|_\infty \le \frac{1}{\sqrt{2}}\ln\left(\frac{me}{n}\right)\|h\|_{\nabla^2 p(x)}.$$

The last claim follows from the equality

$$\boldsymbol{S}_t^{-1}\frac{d}{dt}s_t = \boldsymbol{A}_t h.$$

□

In the next lemma, we show that w_t is stable both $\boldsymbol{P}^{(2)}$ norm, $\boldsymbol{\Sigma}$ norm and ℓ^∞ norm.

Lemma 6.13.5. *We have*

$$\begin{aligned}\|\boldsymbol{W}_t^{-1}\frac{d}{dt}w_t\|_{\boldsymbol{P}_t^{(2)}} &\le \frac{1}{2}\|h\|_{\nabla^2 p(x_t)}, \\ \|\boldsymbol{W}_t^{-1}\frac{d}{dt}w_t\|_{\boldsymbol{\Sigma}_t} &\le \sqrt{\ln\left(\frac{m}{n}e\right)}\|h\|_{\nabla^2 p(x_t)}, \\ \|\boldsymbol{W}_t^{-1}\frac{d}{dt}w_t\|_\infty &\le \ln(\frac{me}{n})\left(1 + \sqrt{2}\ln\left(\frac{me^2}{n}\right)\right)\|h\|_{\nabla^2 p(x_t)}.\end{aligned}$$

Proof. Using (6.50), we have

$$\frac{d}{dt}w_t = -2\boldsymbol{W}_t\left(\frac{n}{m}\boldsymbol{W}_t + \boldsymbol{P}_t^{(2)}\right)^{-1}\boldsymbol{\Lambda}_t\boldsymbol{A}_t h. \tag{6.52}$$

Let $z = \left(\frac{n}{m}\boldsymbol{W}_t + \boldsymbol{P}_t^{(2)}\right)^{-1}\boldsymbol{\Lambda}_t\boldsymbol{A}_t h$. First, we note that

$$\|z\|^2_{\boldsymbol{P}_t^{(2)}} = h^T\boldsymbol{A}_t^T\boldsymbol{\Lambda}_t\left(\frac{n}{m}\boldsymbol{W}_t + \boldsymbol{P}_t^{(2)}\right)^{-1}\boldsymbol{P}_t^{(2)}\left(\frac{n}{m}\boldsymbol{W}_t + \boldsymbol{P}_t^{(2)}\right)^{-1}\boldsymbol{\Lambda}_t\boldsymbol{A}_t h.$$

Since $\boldsymbol{P}_t^{(2)} \preceq \frac{n}{m}\boldsymbol{W}_t + \boldsymbol{P}_t^{(2)}$ and $\nabla^2 p \succeq 4\boldsymbol{A}_t^T\boldsymbol{\Lambda}_t\left(\frac{n}{m}\boldsymbol{W}_t + \boldsymbol{P}_t^{(2)}\right)^{-1}\boldsymbol{\Lambda}_t\boldsymbol{A}_t$ (6.49), we have

$$\|z\|^2_{\boldsymbol{P}_t^{(2)}} \leq h^T\boldsymbol{A}_t^T\boldsymbol{\Lambda}_t\left(\frac{n}{m}\boldsymbol{W}_t + \boldsymbol{P}_t^{(2)}\right)^{-1}\boldsymbol{\Lambda}_t\boldsymbol{A}_t h \leq \frac{1}{4}\|h\|^2_{\nabla^2 p(x_t)}.$$

For the $\boldsymbol{\Sigma}_t$ bound, note that

$$\|z\|^2_{\boldsymbol{\Sigma}_t} = h^T\boldsymbol{A}_t^T\boldsymbol{\Lambda}_t\left(\frac{n}{m}\boldsymbol{W}_t + \boldsymbol{P}_t^{(2)}\right)^{-1}\boldsymbol{\Sigma}_t\left(\frac{n}{m}\boldsymbol{W}_t + \boldsymbol{P}_t^{(2)}\right)^{-1}\boldsymbol{\Lambda}_t\boldsymbol{A}_t h.$$

Using Lemma 6.13.2, we have $\max_i \frac{\sigma_{t,i}}{w_{t,i}} \leq \frac{n}{m}\ln(\frac{me}{n})$ and hence

$$\begin{aligned}\|z\|^2_{\boldsymbol{\Sigma}_t} &\leq \ln\left(\frac{me}{n}\right)h^T\boldsymbol{A}_t^T\boldsymbol{\Lambda}_t\left(\frac{n}{m}\boldsymbol{W}_t + \boldsymbol{P}_t^{(2)}\right)^{-1}\left(\frac{n}{m}\boldsymbol{W}_t\right)\left(\frac{n}{m}\boldsymbol{W}_t + \boldsymbol{P}_t^{(2)}\right)^{-1}\boldsymbol{\Lambda}_t\boldsymbol{A}_t h \\ &\leq \ln\left(\frac{me}{n}\right)h^T\boldsymbol{A}_t^T\boldsymbol{\Lambda}_t\left(\frac{n}{m}\boldsymbol{W}_t + \boldsymbol{P}_t^{(2)}\right)^{-1}\boldsymbol{\Lambda}_t\boldsymbol{A}_t h \leq \frac{\ln\left(\frac{me}{n}\right)}{4}\|h\|^2_{\nabla^2 p(x_t)}.\end{aligned}$$

For the ℓ^∞ bound, note that

$$\left(\frac{n}{m}\boldsymbol{W}_t + \boldsymbol{P}_t^{(2)}\right)z = \boldsymbol{\Sigma}_t\boldsymbol{A}_t h - \boldsymbol{P}_t^{(2)}\boldsymbol{A}_t h.$$

Hence, we have

$$\frac{n}{m}w_{t,i}\left|z_i\right| \leq \left|\boldsymbol{P}_t^{(2)}z\right|_i + \sigma_{t,i}\left|\boldsymbol{A}_t h\right|_i + \left|\boldsymbol{P}_t^{(2)}\boldsymbol{A}_t h\right|_i. \tag{6.53}$$

For the term $\left|\boldsymbol{P}_t^{(2)}z\right|_i$, note that

$$\left|\boldsymbol{P}_t^{(2)}z\right|_i = \sqrt{1_i^T\boldsymbol{P}_t^{(2)}1_i}\sqrt{\|z\|^2_{\boldsymbol{P}_t^{(2)}}} = \sigma_{t,i}\|z\|_{\boldsymbol{P}_t^{(2)}} \leq \frac{\sigma_{t,i}}{2}\|h\|_{\nabla^2 p(x_t)}.$$

Therefore, $\left|\boldsymbol{P}_t^{(2)}z\right|_i \leq \frac{\sigma_{t,i}}{2}\|h\|_{\nabla^2 p(x_t)}$.

For the term $\left|\boldsymbol{P}_t^{(2)}\boldsymbol{A}_t h\right|_i$, using $\nabla^2 p(x_t) \succeq 2\boldsymbol{A}_t^t\boldsymbol{\Sigma}_t\boldsymbol{A}_t$ and $\boldsymbol{\Sigma}_t \succeq \boldsymbol{P}_t^{(2)}$, we have

$$\begin{aligned}\left|\boldsymbol{P}_t^{(2)}\boldsymbol{A}_t h\right|_i &= \sqrt{1_i^T\boldsymbol{P}_t^{(2)}1_i}\sqrt{\|\boldsymbol{A}_t h\|^2_{\boldsymbol{P}_t^{(2)}}} \\ &\leq \sigma_{t,i}\|\boldsymbol{A}_t h\|_{\boldsymbol{\Sigma}_t} \\ &\leq \frac{\sigma_{t,i}}{\sqrt{2}}\|h\|_{\nabla^2 p(x_t)}.\end{aligned}$$

For the term $\left|\boldsymbol{A}_t h\right|_i$, Lemma 6.13.4 shows that $\left|\boldsymbol{A}_t h\right|_i \leq \frac{1}{\sqrt{2}}\ln\left(\frac{me}{n}\right)\|h\|_{\nabla^2 p(x_t)}$.

Combining these terms into (6.53), we have

$$\frac{n}{m}w_{t,i}\left|z_i\right| \leq \sigma_{t,i}\left(\frac{1}{2} + \frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}\ln\left(\frac{me}{n}\right)\right)\|h\|_{\nabla^2 p(x_t)}.$$

Using Lemma 6.13.2, we have $\max_i \frac{\sigma_{t,i}}{w_{t,i}} \leq \frac{n}{m}\ln(\frac{me}{n})$ and hence

$$|z_i| \leq \ln(\frac{me}{n})\left(\frac{1}{2} + \frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}\ln\left(\frac{me}{n}\right)\right)\|h\|_{\nabla^2 p(x_t)}.$$

Using the formula (6.52) for w_t, we have the result. □

In the next lemma, we show that $\boldsymbol{\Sigma}_t$ is stable.

Lemma 6.13.6. *We have*

$$\left|\frac{d}{dt}\boldsymbol{\Sigma}_t\right| \preceq 2\ln(\frac{me}{n})\left(1 + \sqrt{2}\ln\left(\frac{me^2}{n}\right)\right)\|h\|_{\nabla^2 p(x_t)}\boldsymbol{\Sigma}_t.$$

Proof. Fix any $y \in \mathbb{R}^n$. Let $\alpha(t) = y^T \boldsymbol{A}_t^T \boldsymbol{\Sigma}_t \boldsymbol{A}_t y$. The optimality condition (6.48) gives

$$\sigma_t = \frac{n}{m} w_t \ln(e w_t).$$

Taking derivative on both side, we have

$$\begin{aligned}\frac{d}{dt}\sigma_t &= \frac{n}{m}\left(\frac{d}{dt}w_t\right)\ln(ew_t) + \frac{n}{m}\left(\frac{d}{dt}w_t\right)\\ &= \frac{n}{m}\left(\frac{d}{dt}w_t\right)\ln(e^2 w_t)\end{aligned}$$

Using Lemma 6.13.5, we have

$$\|\boldsymbol{W}_t^{-1}\frac{d}{dt}w_t\|_\infty \leq \ln(\frac{me}{n})\left(1 + \sqrt{2}\ln\left(\frac{me^2}{n}\right)\right)\|h\|_{\nabla^2 p(x_t)}.$$

Therefore, for all i, we have

$$\begin{aligned}\left|\frac{d}{dt}\sigma_{t,i}\right| &\leq \ln(\frac{me}{n})\left(1 + \sqrt{2}\ln\left(\frac{me^2}{n}\right)\right)\frac{n}{m}w_{t,i}\ln(e^2 w_t)\\ &\leq 2\ln(\frac{me}{n})\left(1 + \sqrt{2}\ln\left(\frac{me^2}{n}\right)\right)\sigma_{t,i}\|h\|_{\nabla^2 p(x_t)}.\end{aligned}$$

This gives the result. □

Now, we want to prove $\boldsymbol{P}_t^{(2)}$ is stable.

Lemma 6.13.7. *We have*

$$\left|\frac{d}{dt}y^T\boldsymbol{P}_t^{(2)}y\right| \preceq \left(\ln(\frac{me}{n})\left(1 + \sqrt{2}\ln\left(\frac{me^3}{n}\right)\right) + 2\sqrt{\ln\left(\frac{m}{n}e\right)} + 4\right)\|y\|_{\boldsymbol{\Sigma}_t}^2\|h\|_{\nabla^2 p(x_t)}.$$

Proof. Fix any $y \in \mathbb{R}^n$. Let $\alpha(t) = y^T\boldsymbol{P}_t^{(2)}y$. Then, we have

$$\frac{d\alpha}{dt} = \frac{d}{dt}\sum_{i,j}(\boldsymbol{P}_t)_{i,j}^2 y_i y_j = 2\sum_{i,j}(\boldsymbol{P}_t)_{i,j}\frac{d}{dt}(\boldsymbol{P}_t)_{i,j}y_i y_j.$$

Let $z_t = \frac{d}{dt}\ln\left(w_t/s_t^2\right)$. Note that

$$\begin{aligned}\frac{d}{dt}\left(\boldsymbol{P}_t\right)_{i,j} &= \frac{d}{dt}1_i^T\sqrt{\boldsymbol{W}_t}\boldsymbol{A}_t\left(\boldsymbol{A}_t^T\boldsymbol{W}_t\boldsymbol{A}_t\right)^{-1}\boldsymbol{A}_t^T\sqrt{\boldsymbol{W}_t}1_j \\ &= \frac{1}{2}\frac{dz_{t,i}}{dt}\left(\boldsymbol{P}_t\right)_{i,j} + \frac{1}{2}\frac{dz_{t,j}}{dt}\left(\boldsymbol{P}_t\right)_{i,j} - 1_i^T\cdot\boldsymbol{P}_t\cdot\boldsymbol{Diag}\frac{dz_{t,k}}{dt}\cdot\boldsymbol{P}_t\cdot 1_j.\end{aligned}$$

Hence, we have

$$\begin{aligned}\frac{d\alpha}{dt} &= \sum_{i,j}\left(\boldsymbol{P}_t\right)_{i,j}\frac{dz_{t,i}}{dt}\left(\boldsymbol{P}_t\right)_{i,j}y_iy_j + \sum_{i,j}\left(\boldsymbol{P}_t\right)_{i,j}\frac{dz_{t,j}}{dt}\left(\boldsymbol{P}_t\right)_{i,j}y_iy_j \\ &\quad -2\sum_{i,j}\left(\boldsymbol{P}_t\right)_{i,j}1_i^T\cdot\boldsymbol{P}_t\cdot\boldsymbol{Diag}\left(\frac{dz_{t,k}}{dt}\right)\cdot\boldsymbol{P}_t\cdot 1_jy_iy_j \\ &= 2y^T\boldsymbol{P}_t^{(2)}\boldsymbol{Diag}\left(\frac{dz_{t,j}}{dt}\right)y - 2\sum_{i,j,k}\left(\boldsymbol{P}_t\right)_{i,j}\left(\boldsymbol{P}_t\right)_{i,k}\left(\boldsymbol{P}_t\right)_{j,k}y_iy_j\frac{dz_{t,k}}{dt}. \qquad (6.54)\end{aligned}$$

For the first term $y^T\boldsymbol{P}_t^{(2)}\boldsymbol{diag}\left(\frac{dz_{t,j}}{dt}\right)y$, using $\boldsymbol{P}_t^{(2)}\preceq\boldsymbol{\Sigma}_t$, we have

$$\left|y^T\boldsymbol{P}_t^{(2)}\boldsymbol{Diag}\left(\frac{dz_{t,j}}{dt}\right)y\right| \leq \|y\|_{\boldsymbol{P}_t^{(2)}}\|\boldsymbol{Diag}\left(\frac{dz_{t,j}}{dt}\right)y\|_{\boldsymbol{P}_t^{(2)}} \leq \|y\|_{\boldsymbol{\Sigma}_t}^2\|\frac{dz_t}{dt}\|_\infty.$$

Using Lemma 6.13.4 and 6.13.5, we have

$$\begin{aligned}\|\frac{dz_t}{dt}\|_\infty &\leq \|\frac{d}{dt}\ln(w_t)\|_\infty + 2\|\frac{d}{dt}\ln(s_t)\|_\infty \\ &\leq \ln(\frac{me}{n})\left(1+\sqrt{2}\ln\left(\frac{me^3}{n}\right)\right)\|h\|_{\nabla^2 p(x_t)}.\end{aligned}$$

Hence, we have

$$\left|y^T\boldsymbol{P}_t^{(2)}\boldsymbol{Diag}\left(\frac{dz_{t,j}}{dt}\right)y\right| \leq \left(\ln(\frac{me}{n})\left(1+\sqrt{2}\ln\left(\frac{me^3}{n}\right)\right)\right)\|y\|_{\boldsymbol{\Sigma}_t}^2\|h\|_{\nabla^2 p(x_t)}.$$

For the second term $\sum_{i,j,k}\left(\boldsymbol{P}_t\right)_{i,j}\left(\boldsymbol{P}_t\right)_{i,k}\left(\boldsymbol{P}_t\right)_{j,k}y_iy_j\frac{d}{dt}z_{t,k}$, we let $a_{i,j} = \left(\boldsymbol{P}_t\right)_{i,j}\sqrt{|y_iy_j|}$, $b_{i,k} = \left(\boldsymbol{P}_t\right)_{i,k}\sqrt{\left|y_i\frac{d}{dt}z_{t,k}\right|}$ and $c_{j,k} = \left(\boldsymbol{P}_t\right)_{j,k}\sqrt{\left|y_j\frac{d}{dt}z_{t,k}\right|}$ and get

$$\left|\sum_{i,j,k}\left(\boldsymbol{P}_t\right)_{i,j}\left(\boldsymbol{P}_t\right)_{i,k}\left(\boldsymbol{P}_t\right)_{j,k}y_iy_j\frac{d}{dt}z_{t,k}\right| \leq \sum_{i,j,k}\left|a_{i,j}b_{i,k}c_{j,k}\right|.$$

Using Cauchy Schwarz Inequality twice and $\boldsymbol{P}_t^{(2)}\preceq\boldsymbol{\Sigma}_t$, we have

$$\begin{aligned}\left|\sum_{i,j,k}\left(\boldsymbol{P}_t\right)_{i,j}\left(\boldsymbol{P}_t\right)_{i,k}\left(\boldsymbol{P}_t\right)_{j,k}y_iy_j\frac{d}{dt}z_{t,k}\right| &\leq \sqrt{\sum_{i,j}a_{i,j}^2}\sqrt{\sum_{i,k}b_{i,k}^2}\sqrt{\sum_{j,k}c_{j,k}^2} \\ &= \|\,|y|\,\|_{\boldsymbol{P}_t^{(2)}}\left\langle|y|,\left|\frac{d}{dt}z_t\right|\right\rangle_{\boldsymbol{P}_t^{(2)}}^2 \\ &\leq \|y\|_{\boldsymbol{\Sigma}_t}^2\|\frac{d}{dt}z_t\|_{\boldsymbol{\Sigma}_t}.\end{aligned}$$

Using Lemma 6.13.5, we have

$$\left|\sum_{i,j,k}(\boldsymbol{P}_t)_{i,j}(\boldsymbol{P}_t)_{i,k}(\boldsymbol{P}_t)_{j,k}y_iy_j\frac{d}{dt}z_{t,k}\right| \leq \left(\sqrt{\ln\left(\frac{m}{n}e\right)}+2\right)\|y\|_{\boldsymbol{\Sigma}_t}^2\|h\|_{\nabla^2 p(x_t)}.$$

Combining two cases into (6.54), we have

$$\left|\frac{d\alpha}{dt}\right| \leq \left(\ln(\frac{me}{n})\left(1+\sqrt{2}\ln\left(\frac{me^3}{n}\right)\right)+2\sqrt{\ln\left(\frac{m}{n}e\right)}+4\right)\|y\|_{\boldsymbol{\Sigma}_t}^2\|h\|_{\nabla^2 p(x_t)}.$$

□

■ 6.13.4 $\nabla^2 p$ is stable

Now, we can combine all lemmas before to prove the wanted statement.

Lemma 6.13.8. *We have*

$$\left|\frac{d}{dt}\nabla^2 p\right| \preceq O\left(\ln^3\left(\frac{me}{n}\right)\right)\|h\|_{\nabla^2 p(x_t)}\nabla^2 p.$$

Proof. Note that

$$\nabla^2 p(x_t) = \boldsymbol{A}_t^T\left(6\boldsymbol{\Sigma}_t - 4\boldsymbol{P}_t^{(2)}+\frac{n}{m}I\right)\boldsymbol{A}_t + 4\boldsymbol{A}_t^T\boldsymbol{\Lambda}_t\left(\boldsymbol{P}_t^{(2)}+\frac{n}{m}\boldsymbol{W}_t\right)^{-1}\boldsymbol{\Lambda}_t\boldsymbol{A}_t.$$

□

Fix any $y\in\mathbb{R}^n$. Let $\alpha(t)=y^T\nabla^2 p(x_t)y$. Then, we have

$$\begin{aligned}
\left|\frac{d\alpha}{dt}\right| \leq\ & 6\left|\frac{d}{dt}y^T\boldsymbol{A}_t^T\boldsymbol{\Sigma}_t\boldsymbol{A}_t y\right| + 4\left|\frac{d}{dt}y^T\boldsymbol{A}_t^T\boldsymbol{P}_t^{(2)}\boldsymbol{A}_t y\right| \\
& +4\left|\frac{d}{dt}\boldsymbol{A}_t^T\boldsymbol{\Lambda}_t\left(\boldsymbol{P}_t^{(2)}+\frac{n}{m}\boldsymbol{W}_t\right)^{-1}\boldsymbol{\Lambda}_t\boldsymbol{A}_t\right| \\
\leq\ & 12\left|y^T\boldsymbol{A}_t^T\boldsymbol{\Sigma}_t\left(\frac{d}{dt}\boldsymbol{A}_t\right)y\right| + 6\left|y^T\boldsymbol{A}_t^T\left(\frac{d}{dt}\boldsymbol{\Sigma}_t\right)\boldsymbol{A}_t y\right| \\
& +8\left|y^T\boldsymbol{A}_t^T\boldsymbol{P}_t^{(2)}\left(\frac{d}{dt}\boldsymbol{A}_t\right)y\right| + 4\left|y^T\boldsymbol{A}_t^T\left(\frac{d}{dt}\boldsymbol{P}_t^{(2)}\right)\boldsymbol{A}_t y\right| \\
& +8\left|y^T\boldsymbol{A}_t^T\boldsymbol{\Lambda}_t\left(\boldsymbol{P}_t^{(2)}+\frac{n}{m}\boldsymbol{W}_t\right)^{-1}\boldsymbol{\Lambda}_t\left(\frac{d}{dt}\boldsymbol{A}_t\right)y\right| \\
& +8\left|y^T\boldsymbol{A}_t^T\boldsymbol{\Lambda}_t\left(\boldsymbol{P}_t^{(2)}+\frac{n}{m}\boldsymbol{W}_t\right)^{-1}\left(\frac{d}{dt}\boldsymbol{P}_t^{(2)}\right)\boldsymbol{A}_t y\right| \\
& +8\left|y^T\boldsymbol{A}_t^T\boldsymbol{\Lambda}_t\left(\boldsymbol{P}_t^{(2)}+\frac{n}{m}\boldsymbol{W}_t\right)^{-1}\left(\frac{d}{dt}\boldsymbol{\Sigma}_t\right)\boldsymbol{A}_t y\right| \\
& +4\left|y^T\boldsymbol{A}_t^T\boldsymbol{\Lambda}_t\left(\boldsymbol{P}_t^{(2)}+\frac{n}{m}\boldsymbol{W}_t\right)^{-1}\left(\frac{d}{dt}\boldsymbol{P}_t^{(2)}\right)\left(\boldsymbol{P}_t^{(2)}+\frac{n}{m}\boldsymbol{W}_t\right)^{-1}\boldsymbol{\Lambda}_t\boldsymbol{A}_t y\right| \\
& +4\left|y^T\boldsymbol{A}_t^T\boldsymbol{\Lambda}_t\left(\boldsymbol{P}_t^{(2)}+\frac{n}{m}\boldsymbol{W}_t\right)^{-1}\left(\frac{d}{dt}\frac{n}{m}\boldsymbol{W}_t\right)\left(\boldsymbol{P}_t^{(2)}+\frac{n}{m}\boldsymbol{W}_t\right)^{-1}\boldsymbol{\Lambda}_t\boldsymbol{A}_t y\right|.
\end{aligned}$$

Now, we need to bound the terms above one by one. Using Lemma 6.13.4, we have

$$\begin{aligned}\left|y^T \boldsymbol{A}_t^T \boldsymbol{\Sigma}_t \left(\frac{d}{dt}\boldsymbol{A}_t\right) y\right| &\leq O\left(\ln\left(\frac{me}{n}\right)\right) \|h\|_{\nabla^2 p(x_t)} \left|y^T \boldsymbol{A}_t^T \boldsymbol{\Sigma}_t \boldsymbol{A}_t y\right| \\ &= O\left(\ln\left(\frac{me}{n}\right)\right) \|h\|_{\nabla^2 p(x_t)} \|y\|^2_{\nabla^2 p(x_t)}.\end{aligned}$$

Using Lemma 6.13.6, we have

$$\begin{aligned}\left|y^T \boldsymbol{A}_t^T \left(\frac{d}{dt}\boldsymbol{\Sigma}_t\right) \boldsymbol{A}_t y\right| &\leq O\left(\ln^2\left(\frac{me}{n}\right)\right) \|h\|_{\nabla^2 p(x_t)} \left|y^T \boldsymbol{A}_t^T \boldsymbol{\Sigma}_t \boldsymbol{A}_t y\right| \\ &\leq O\left(\ln^2\left(\frac{me}{n}\right)\right) \|h\|_{\nabla^2 p(x_t)} \|y\|^2_{\nabla^2 p(x_t)}.\end{aligned}$$

Using Lemma 6.13.4, we have

$$\begin{aligned}\left|y^T \boldsymbol{A}_t^T \boldsymbol{P}_t^{(2)} \left(\frac{d}{dt}\boldsymbol{A}_t\right) y\right| &\leq \sqrt{y^T \boldsymbol{A}_t^T \boldsymbol{P}_t^{(2)} \boldsymbol{A}_t y} \sqrt{y^T \left(\frac{d}{dt}\boldsymbol{A}_t\right) \boldsymbol{P}_t^{(2)} \left(\frac{d}{dt}\boldsymbol{A}_t\right) y} \\ &\leq O(1)\sqrt{\|y\|^2_{\nabla^2 p(x_t)}} \sqrt{\left(\ln\left(\frac{me}{n}\right) \|h\|_{\nabla^2 p(x_t)}\right)^2 \|y\|^2_{\nabla^2 p(x_t)}} \\ &= O\left(\ln\left(\frac{me}{n}\right)\right) \|h\|_{\nabla^2 p(x_t)} \|y\|^2_{\nabla^2 p(x_t)}.\end{aligned}$$

Using Lemma 6.13.7, we have

$$\begin{aligned}\left|y^T \boldsymbol{A}_t^T \left(\frac{d}{dt}\boldsymbol{P}_t^{(2)}\right) \boldsymbol{A}_t y\right| &\leq O\left(\ln^2\left(\frac{me}{n}\right)\right) \|\boldsymbol{A}_t y\|^2_{\boldsymbol{\Sigma}_t} \|h\|_{\nabla^2 p(x_t)} \\ &= O\left(\ln^2\left(\frac{me}{n}\right)\right) \|h\|_{\nabla^2 p(x_t)} \|y\|^2_{\nabla^2 p(x_t)}.\end{aligned}$$

Using Lemma 6.13.4, we have

$$\begin{aligned}&\left|y^T \boldsymbol{A}_t^T \boldsymbol{\Lambda}_t \left(\boldsymbol{P}_t^{(2)} + \frac{n}{m}\boldsymbol{W}_t\right)^{-1} \boldsymbol{\Lambda}_t \left(\frac{d}{dt}\boldsymbol{A}_t\right) y\right| \\ \leq\ & \sqrt{y^T \boldsymbol{A}_t^T \boldsymbol{\Lambda}_t \left(\boldsymbol{P}_t^{(2)} + \frac{n}{m}\boldsymbol{W}_t\right)^{-1} \boldsymbol{\Lambda}_t \left(\boldsymbol{P}_t^{(2)} + \frac{n}{m}\boldsymbol{W}_t\right)^{-1} \boldsymbol{A}_t^T y} \\ & \times \sqrt{y^T \left(\frac{d}{dt}\boldsymbol{A}_t\right)^T \boldsymbol{\Lambda}_t \left(\frac{d}{dt}\boldsymbol{A}_t\right) y} \\ =\ & O(1)\sqrt{\ln(\frac{me}{n})\|y\|^2_{\nabla^2 p(x_t)}} \sqrt{y^T \left(\frac{d}{dt}\boldsymbol{A}_t\right)^T \Sigma_t \left(\frac{d}{dt}\boldsymbol{A}_t\right) y} \\ =\ & O\left(\ln^{3/2}\left(\frac{me}{n}\right)\right) \|h\|_{\nabla^2 p(x_t)} \|y\|^2_{\nabla^2 p(x_t)}\end{aligned}$$

Using Lemma 6.13.6, we have

$$\begin{aligned}
&\left|y^T \boldsymbol{A}_t^T \boldsymbol{\Lambda}_t \left(\boldsymbol{P}_t^{(2)} + \frac{n}{m}\boldsymbol{W}_t\right)^{-1} \left(\frac{d}{dt}\boldsymbol{\Sigma}_t\right) \boldsymbol{A}_t y\right| \\
\leq\ & \sqrt{y^T \boldsymbol{A}_t^T \boldsymbol{\Lambda}_t \left(\boldsymbol{P}_t^{(2)} + \frac{n}{m}\boldsymbol{W}_t\right)^{-1} \left(\frac{d}{dt}\boldsymbol{\Sigma}_t\right) \left(\boldsymbol{P}_t^{(2)} + \frac{n}{m}\boldsymbol{W}_t\right)^{-1} \boldsymbol{A}_t^T y} \\
& \times \sqrt{y^T \boldsymbol{A}_t^T \left(\frac{d}{dt}\boldsymbol{\Sigma}_t\right) \boldsymbol{A}_t y} \\
\leq\ & O\left(\ln^{3/2}\left(\frac{me}{n}\right)\right) \|h\|_{\nabla^2 p(x_t)} \|y\|^2_{\nabla^2 p(x_t)}
\end{aligned}$$

Let $z = \left(\boldsymbol{P}_t^{(2)} + \frac{n}{m}\boldsymbol{W}_t\right)^{-1} \boldsymbol{\Lambda}_t \boldsymbol{A}_t y$, using Lemma 6.13.7, we have

$$\begin{aligned}
&\left|y^T \boldsymbol{A}_t^T \boldsymbol{\Lambda}_t \left(\boldsymbol{P}_t^{(2)} + \frac{n}{m}\boldsymbol{W}_t\right)^{-1} \left(\frac{d}{dt}\boldsymbol{P}_t^{(2)}\right) \left(\boldsymbol{P}_t^{(2)} + \frac{n}{m}\boldsymbol{W}_t\right)^{-1} \boldsymbol{\Lambda}_t \boldsymbol{A}_t y\right| \\
\leq\ & O\left(\ln^2\left(\frac{me}{n}\right)\right) \|z\|^2_{\boldsymbol{\Sigma}_t} \|h\|_{\nabla^2 p(x_t)}.
\end{aligned}$$

Note that

$$\begin{aligned}
\|z\|^2_{\boldsymbol{\Sigma}_t} &= y^T \boldsymbol{A}_t^T \boldsymbol{\Lambda}_t \left(\boldsymbol{P}_t^{(2)} + \frac{n}{m}\boldsymbol{W}_t\right)^{-1} \boldsymbol{\Sigma}_t \left(\boldsymbol{P}_t^{(2)} + \frac{n}{m}\boldsymbol{W}_t\right)^{-1} \boldsymbol{\Lambda}_t \boldsymbol{A}_t y \\
&\leq \ln(\frac{me}{n}) y^T \boldsymbol{A}_t^T \boldsymbol{\Lambda}_t \left(\boldsymbol{P}_t^{(2)} + \frac{n}{m}\boldsymbol{W}_t\right)^{-1} \boldsymbol{\Lambda}_t \boldsymbol{A}_t y \leq O\left(\ln(\frac{me}{n})\right) \|y\|^2_{\nabla^2 p(x_t)}.
\end{aligned}$$

Hence, we have

$$\begin{aligned}
&\left|y^T \boldsymbol{A}_t^T \boldsymbol{\Lambda}_t \left(\boldsymbol{P}_t^{(2)} + \frac{n}{m}\boldsymbol{W}_t\right)^{-1} \left(\frac{d}{dt}\boldsymbol{P}_t^{(2)}\right) \left(\boldsymbol{P}_t^{(2)} + \frac{n}{m}\boldsymbol{W}_t\right)^{-1} \boldsymbol{\Lambda}_t \boldsymbol{A}_t y\right| \\
\leq\ & O\left(\ln^3(\frac{me}{n})\right) \|y\|^2_{\nabla^2 p(x_t)} \|h\|_{\nabla^2 p(x_t)}.
\end{aligned}$$

Using Lemma 6.13.5, we have

$$\begin{aligned}
&\left|y^T \boldsymbol{A}_t^T \boldsymbol{\Lambda}_t \left(\boldsymbol{P}_t^{(2)} + \frac{n}{m}\boldsymbol{W}_t\right)^{-1} \left(\frac{d}{dt}\frac{n}{m}\boldsymbol{W}_t\right) \left(\boldsymbol{P}_t^{(2)} + \frac{n}{m}\boldsymbol{W}_t\right)^{-1} \boldsymbol{\Lambda}_t \boldsymbol{A}_t y\right| \\
\leq\ & O\left(\ln^2(\frac{me}{n})\right) \|y\|^2_{\nabla^2 p(x_t)} \|h\|_{\nabla^2 p(x_t)}
\end{aligned}$$

Using Lemma 6.13.7, we have

$$\begin{aligned}
&\left|y^T \boldsymbol{A}_t^T \boldsymbol{\Lambda}_t \left(\boldsymbol{P}_t^{(2)} + \frac{n}{m}\boldsymbol{W}_t\right)^{-1} \left(\frac{d}{dt}\boldsymbol{P}_t^{(2)}\right) \boldsymbol{A}_t y\right| \\
\leq\ & \sqrt{y^T \boldsymbol{A}_t^T \boldsymbol{\Lambda}_t \left(\boldsymbol{P}_t^{(2)} + \frac{n}{m}\boldsymbol{W}_t\right)^{-1} \left(\frac{d}{dt}\boldsymbol{P}_t^{(2)}\right) \left(\boldsymbol{P}_t^{(2)} + \frac{n}{m}\boldsymbol{W}_t\right)^{-1} \boldsymbol{\Lambda}_t \boldsymbol{A}_t y} \sqrt{y^T \boldsymbol{A}_t^T \left(\frac{d}{dt}\boldsymbol{P}_t^{(2)}\right) \boldsymbol{A}_t y} \\
\leq\ & \sqrt{O(\ln^3\left(\frac{me}{n}\right)) \|y\|^2_{\nabla^2 p(x_t)} \|h\|_{\nabla^2 p(x_t)}} \sqrt{O(\ln^2(\frac{me}{n})) \|y\|^2_{\nabla^2 p(x_t)} \|h\|_{\nabla^2 p(x_t)}} \\
=\ & O\left(\ln^{5/2}(\frac{me}{n})\right) \|y\|^2_{\nabla^2 p(x_t)} \|h\|_{\nabla^2 p(x_t)}.
\end{aligned}$$

Combining all terms together, we have $\left|\frac{d\alpha}{dt}\right| = O\left(\ln^3(\frac{me}{n})\right) \|y\|^2_{\nabla^2 p(x_t)} \|h\|_{\nabla^2 p(x_t)}$.

Chapter 7

Geometric Median In Nearly-Linear Time

7.1 Introduction

In this chapter, we study Fermat-Weber problem, one of the oldest easily-stated nontrivial problems in computational geometry. Given a set of n points in d *dimensions*, $a^{(1)}, \ldots, a^{(n)} \in \mathbb{R}^d$, find a point $x_* \in \mathbb{R}^d$ that minimizes the sum of Euclidean distances to them:

$$x_* \in \arg\min_{x \in \mathbb{R}^d} f(x) \quad \text{where} \quad f(x) \stackrel{\text{def}}{=} \sum_{i \in [n]} \|x - a^{(i)}\|_2$$

This problem, also known as the *geometric median problem,* is well studied and has numerous applications. It is often considered over low dimensional spaces in the context of the facility location problem [267] and over higher dimensional spaces it has applications to clustering in machine learning and data analysis. For example, computing the geometric median is a subroutine in popular expectation maximization heuristics for k-medians clustering.

The problem is also important to robust estimation, where we like to find a point representative of given set of points that is resistant to outliers. The geometric median is a rotation and translation invariant estimator that achieves the optimal *breakdown point* of 0.5, i.e. it is a good estimator even when up to half of the input data is arbitrarily corrupted [172]. Moreover, if a large constant fraction of the points lie in a ball of diameter ε then the geometric median lies in that ball with diameter $O(\varepsilon)$ (see Lemma 7.7.1). Consequently, the geometric median can be used to turn expected results into high probability results: e.g. if the $a^{(i)}$ are drawn independently such that $\mathbf{E}\|x - x_*\|_2 \leq \varepsilon$ for some $\varepsilon > 0$ and $x \in \mathbb{R}^d$ then this fact, Markov bound, and Chernoff Bound, imply $\|x_* - a^{(i)}\|_2 = O(\varepsilon)$ with high probability in n.

Despite the ancient nature of the Fermat-Weber problem and its many uses there are relatively few theoretical guarantees for solving it (see Table 7.1). To compute a $(1+\varepsilon)$-approximate solution, i.e. $x \in \mathbb{R}^d$ with $f(x) \leq (1+\varepsilon) f(x_*)$, the previous fastest running times were either $O(d \cdot n^{4/3} \varepsilon^{-8/3})$ by [50], $\tilde{O}(d \exp \varepsilon^{-4} \log \varepsilon^{-1})$ by [25], $\tilde{O}(nd + \text{poly}(d, \varepsilon^{-1}))$ by [86], or $O((nd)^{O(1)} \log \frac{1}{\varepsilon})$ time by [222, 271]. In this chapter, we improve upon these running times by providing an $O(nd \log^3 \frac{n}{\varepsilon})$ time algorithm[1] as well as an $O(d/\varepsilon^2)$ time algorithm, provided we have an oracle for sampling a random $a^{(i)}$. Picking the faster algorithm for the particular value of ε improves the running time to $O(nd \log^3 \frac{1}{\varepsilon})$. We also extend these results to compute a $(1+\varepsilon)$-approximate solution to the more general Weber's problem, $\min_{x \in \mathbb{R}^d} \sum_{i \in [n]} w_i \|x - a^{(i)}\|_2$ for non-negative w_i, in time $O(nd \log^3 \frac{1}{\varepsilon})$ (see Section 7.10).

Our $O(nd \log^3 \frac{n}{\varepsilon})$ time algorithm is a careful modification of standard interior point methods for solving the geometric median problem. We provide a long step interior point method tailored to the geometric median problem for which we can implement every iteration in nearly linear time. While our analysis starts with a simple $O((nd)^{O(1)} \log \frac{1}{\varepsilon})$ time interior point method and shows how to improve it,

[1] If z is the total number of nonzero entries in the coordinates of the $a^{(i)}$ then a careful analysis of our algorithm improves our running time to $O(z \log^3 \frac{n}{\varepsilon})$.

our final algorithm is quite non-standard from the perspective of interior point literature. Our result is one of very few cases we are aware of outperforming traditional interior point theory (Chapter 6 and [183]) and the only we are aware of using interior point methods to obtain a nearly linear time algorithm for a canonical optimization problem that traditionally requires superlinear time. We hope our work leads to further improvements in this line of research.

Our $O(d\varepsilon^{-2})$ algorithm is a relatively straightforward application of sampling techniques and stochastic subgradient descent. Some additional insight is required simply to provide a rigorous analysis of the robustness of the geometric median and use this to streamline our application of stochastic subgradient descent. We include it for completeness however, we defer its proof to Section 7.7. The bulk of the work in this chapter is focused on developing our $O(nd\log^3\frac{n}{\varepsilon})$ time algorithm which we believe uses a set of techniques of independent interest.

7.1.1 Previous Work

The geometric median problem was first formulated for the case of three points in the early 1600s by Pierre de Fermat [148, 69]. A simple elegant ruler and compass construction was given in the same century by Evangelista Torricelli. Such a construction does not generalize when a larger number of points is considered: Bajaj has shown the even for five points, the geometric median is not expressible by radicals over the rationals [27]. Hence, the $(1+\varepsilon)$-approximate problem has been studied for larger values of n.

Many authors have proposed algorithms with runtime polynomial in n, d and $1/\varepsilon$. The most cited and used algorithm is Weiszfeld's 1937 algorithm [268]. Unfortunately Weiszfeld's algorithm may not converge and if it does it may do so very slowly. There have been many proposed modifications to Weiszfeld's algorithm [58, 224, 220, 28, 260, 155] that generally give non-asymptotic runtime guarantees. In light of more modern multiplicative weights methods his algorithm can be viewed as a re-weighted least squares iteration. Chin et al. [50] considered the more general L_2 embedding problem: placing the vertices of a graph into $\mathbb{R}^d$, where some of the vertices have fixed positions while the remaining vertices are allowed to float, with the objective of minimizing the sum of the Euclidean edge lengths. Using the multiplicative weights method, they obtained a run time of $O(d\cdot n^{4/3}\varepsilon^{-8/3})$ for a broad class of problems, including the geometric median problem.[2]

Many authors consider problems that generalize the Fermat-Weber problem, and obtain algorithms for finding the geometric median as a specialization. For example, Badoiu et al. give an approximate k-median algorithm by sub-sampling with the runtime for $k=1$ of $\widetilde{O}(d\cdot\exp(O(\varepsilon^{-4})))$ [25]. Parrilo and Sturmfels demonstrated that the problem can be reduced to semidefinite programming, thus obtaining a runtime of $\widetilde{O}(\mathrm{poly}(n,d)\log\varepsilon^{-1})$ [222]. Furthermore, Bose et al. gave a linear time algorithm for fixed d and ε^{-1}, based on low-dimensional data structures [37] and it has been show how to obtain running times of $\widetilde{O}(nd+\mathrm{poly}(d,\varepsilon^{-1}))$ for this problem and a more general class of problems.[113, 86].

An approach very related to ours was studied by Xue and Ye [271]. They give an interior point method with barrier analysis that runs in time $\tilde{O}((d^3+d^2n)\sqrt{n}\log\varepsilon^{-1})$.

7.1.2 Overview of $O(nd\log^3\frac{n}{\varepsilon})$ Time Algorithm

Interior Point Primer

Our algorithm is broadly inspired by interior point methods, a broad class of methods for efficiently solving convex optimization problems [273, 211]. Given an instance of the geometric median problem

[2] The result of [50] was stated in more general terms than given here. However, it easy to formulate the geometric median problem in their model.

Year	Authors	Runtime	Comments
1659	Torricelli [263]	-	Assuming $n = 3$
1937	Weiszfeld [268]	-	Does not always converge
1990	Chandrasekaran and Tamir[47]	$\widetilde{O}(n \cdot \text{poly}(d) \log \varepsilon^{-1})$	Ellipsoid method
1997	Xue and Ye [271]	$\widetilde{O}((d^3 + d^2 n)\sqrt{n} \log \varepsilon^{-1})$	Interior point with barrier method
2000	Indyk [116]	$\widetilde{O}(dn \cdot \varepsilon^{-2})$	Optimizes only over x in the input
2001	Parrilo and Sturmfels [222]	$\widetilde{O}(\text{poly}(n, d) \log \varepsilon^{-1})$	Reduction to SDP
2002	Badoiu et al. [25]	$\widetilde{O}(d \cdot \exp(O(\varepsilon^{-4})))$	Sampling
2003	Bose et al. [37]	$\widetilde{O}(n)$	Assuming $d, \varepsilon^{-1} = O(1)$
2005	Har-Peled and Kushal [113]	$\widetilde{O}(n + \text{poly}(\varepsilon^{-1}))$	Assuming $d = O(1)$
2011	Feldman and Langberg [86]	$\widetilde{O}(nd + \text{poly}(d, \varepsilon^{-1}))$	Coreset
2013	Chin et al. [50]	$\widetilde{O}(dn^{4/3} \cdot \varepsilon^{-8/3})$	Multiplicative weights
-	**This Chapter**	$O(nd \log^3(n/\varepsilon))$	Interior point with custom analysis
-	**This Chapter**	$O(d\varepsilon^{-2})$	Stochastic gradient descent

Table 7.1: Selected Previous Results.

we first put the problem in a more natural form for applying interior point methods. Rather than writing the problem as minimizing a convex function over $\mathbb{R}^d$

$$\min_{x \in \mathbb{R}^d} f(x) \quad \text{where} \quad f(x) \stackrel{\text{def}}{=} \sum_{i \in [n]} \|x - a^{(i)}\|_2 \tag{7.1}$$

we instead write the problem as minimizing a linear function over a larger convex space:

$$\min_{\{\alpha, x\} \in S} 1^\top \alpha \quad \text{where } S = \left\{ \alpha \in \mathbb{R}^n, x \in \mathbb{R}^d \quad | \quad \|x^{(i)} - a^{(i)}\|_2 \leq \alpha_i \quad \text{for all} \quad i \in [n] \right\}. \tag{7.2}$$

Clearly, these problems are the same as at optimality $\alpha_i = \|x^{(i)} - a^{(i)}\|_2$.

To solve problems of the form (7.2) interior point methods relax the constraint $\{\alpha, x\} \in S$ through the introduction of a *barrier function*. In particular they assume that there is a real valued function p such that as $\{\alpha, x\}$ moves towards the boundary of S the value of p goes to infinity. A popular class of interior point methods known as *path following methods* [227, 107], they consider relaxations of (7.2) of the form $\min_{\{\alpha,x\} \in \mathbb{R}^n \times \mathbb{R}^d} t \cdot 1^\top \alpha + p(\alpha, x)$. The minimizers of this function form a path, known as the central path, parameterized by t. The methods then use variants of Newton's method to follow the path until t is large enough that a high quality approximate solution is obtained. The number of iterations of these methods are then typically governed by a property of p known as its self concordance ν. Given a ν-self concordant barrier, typically interior point methods require $O(\sqrt{\nu} \log \frac{1}{\varepsilon})$ iterations to compute a $(1 + \varepsilon)$-approximate solution.

For our particular convex set, the construction of our barrier function is particularly simple, we consider each constraint $\|x - a^{(i)}\|_2 \leq \alpha_i$ individually. In particular, it is known that the function $p^{(i)}(\alpha, x) = -\ln \alpha_i - \ln \left(\alpha_i^2 - \|x - a^{(i)}\|_2\right)$ is a $O(1)$ self-concordant barrier function for the set $S^{(i)} = \left\{x \in \mathbb{R}^d, \alpha \in \mathbb{R}^n \,|\, \|x - a^{(i)}\|_2 \leq \alpha_i\right\}$ [206]. Since $\cap_{i \in [n]} S^{(i)} = S$ we can use the barrier $\sum_{i \in [n]} p^{(i)}(\alpha, x)$ for $p(\alpha, x)$ and standard self-concordance theory shows that this is an $O(n)$ self concordant barrier for S. Consequently, this easily yields an interior point method for solving the geometric median problem in $O((nd)^{O(1)} \log \frac{1}{\varepsilon})$ time.

Difficulties

Unfortunately obtaining a nearly linear time algorithm for geometric median using interior point methods as presented poses numerous difficulties. Particularly troubling is the number of iterations required by standard interior point algorithms. The approach outlined in the previous section produced an $O(n)$-self concordant barrier and even if we use more advanced self concordance machinery, i.e. the universal barrier [211], the best known self concordance of barrier for the convex set $\sum_{i\in[n]} \|x-a^{(i)}\|_2 \leq c$ is $O(d)$. An interesting open question still left open by our work is to determine what is the minimal self concordance of a barrier for this set.

Consequently, even if we could implement every iteration of an interior point scheme in nearly linear time it is unclear whether one should hope for a nearly linear time interior point algorithm for the geometric median. While there are a instances of outperforming standard self-concordance analysis (Chapter 6), these instances are few, complex, and to varying degrees specialized to the problems they solve. Moreover, we are unaware of any interior point scheme providing a provable nearly linear time for a general nontrivial convex optimization problem.

Beyond Standard Interior Point

Despite these difficulties we do obtain a nearly linear time interior point based algorithm that only requires $O(\log \frac{n}{\varepsilon})$ iterations, i.e. increases to the path parameter. After choosing the natural penalty functions $p^{(i)}$ described above, we optimize in closed form over the α_i to obtain the following penalized objective function:[3]

$$\min_x f_t(x) \quad \text{where} \quad f_t(x) = \sum_{i\in[n]} \sqrt{1+t^2\|x-a^{(i)}\|_2^2} - \ln\left[1+\sqrt{1+t^2\|x-a^{(i)}\|_2^2}\right]$$

We then approximately minimize $f_t(x)$ for increasing t. We let $x_t \stackrel{\text{def}}{=} \arg\min_{x\in\mathbb{R}^d} f_t(x)$ for $x \geq 0$, and thinking of $\{x_t : t \geq 0\}$ as a continuous curve known as the central path, we show how to approximately follow this path. As $\lim_{t\to\infty} x_t = x_*$ this approach yields a $(1+\varepsilon)$-approximation.

So far our analysis is standard and interior point theory yields an $\Omega(\sqrt{n})$ iteration interior point scheme. To overcome this we take a more detailed look at x_t. We note that for any t if there is any rapid change in x_t it must occur in the direction of the smallest eigenvector of $\nabla^2 f_t(x)$, denoted v_t, what we henceforth may refer to as the *bad direction* at x_t. Furthermore, we show that this bad direction is *stable* in the sense that for all directions $d \perp v_t$ it is the case that $d^\top(x_t - x_{t'})$ is small for $t' \leq ct$ for a small constant c.

In fact, we show that this movement over such a *long step,* i.e. constant increase in t, in the directions orthogonal to the bad direction is small enough that for any movement around a ball of this size the Hessian of f_t only changes by a small multiplicative constant. In short, starting at x_t there exists a point y obtained just by moving from x_t in the bad direction, such that y is close enough to $x_{t'}$ that standard first order method will converge quickly to $x_{t'}$! Thus, we might hope to find such a y, quickly converge to $x_{t'}$ and repeat. If we increase t by a multiplicative constant in every such iterations, standard interior point theory suggests that $O(\log \frac{n}{\varepsilon})$ iterations suffices.

Building an Algorithm

To turn the structural result in the previous section into a fast algorithm there are several further issues we need to address. We need to

[3] It is unclear how to extend our proof for the simpler function: $\sum_{i\in[n]} \sqrt{1+t^2\|x-a^{(i)}\|_2^2}$.

- (1) Show how to find the point along the bad direction that is close to $x_{t'}$
- (2) Show how to solve linear systems in the Hessian to actually converge quickly to $x_{t'}$
- (3) Show how to find the bad direction
- (4) Bound the accuracy required by these computations

Deferring (1) for the moment, our solution to the rest are relatively straightforward. Careful inspection of the Hessian of f_t reveals that it is well approximated by a multiple of the identity matrix minus a rank 1 matrix. Consequently using explicit formulas for the inverse of of matrix under rank 1 updates, i.e. the Sherman-Morrison formula, we can solve such systems in nearly linear time thereby addressing (2). For (3), we show that the well known power method carefully applied to the Hessian yields the bad direction if it exists. Finally, for (4) we show that a constant approximate geometric median is near enough to the central path for $t = \Theta(\frac{1}{f(x_*)})$ and that it suffices to compute a central path point at $t = O(\frac{n}{f(x_*)\varepsilon})$ to compute a $1+\varepsilon$-geometric median. Moreover, for these values of t, the precision needed in other operations is clear.

The more difficult operation is (1). Given x_t and the bad direction exactly, it is still not clear how to find the point along the bad direction line from x_t that is close to $x_{t'}$. Just performing binary search on the objective function a priori might not yield such a point due to discrepancies between a ball in Euclidean norm and a ball in hessian norm and the size of the distance from the optimal point in euclidean norm. To overcome this issue we still line search on the bad direction, however rather than simply using $f(x_t + \alpha \cdot v_t)$ as the objective function to line search on, we use the function $g(\alpha) = \min_{\|x - x_t - \alpha \cdot v_t\|_2 \leq c} f(x)$ for some constant c, that is given an α we move α in the bad direction and take the best objective function value in a ball around that point. For appropriate choice of c the minimizers of α will include the optimal point we are looking for. Moreover, we can show that g is convex and that it suffices to perform the minimization approximately.

Putting these pieces together yields our result. We perform $O(\log \frac{n}{\varepsilon})$ iterations of interior point (i.e. increasing t), where in each iteration we spend $O(nd \log \frac{n}{\varepsilon})$ time to compute a high quality approximation to the bad direction, and then we perform $O(\log \frac{n}{\varepsilon})$ approximate evaluations on $g(\alpha)$ to binary search on the bad direction line, and then to approximately evaluate g we perform gradient descent in approximate Hessian norm to high precision which again takes $O(nd \log \frac{n}{\varepsilon})$ time. Altogether this yields a $O(nd \log^3 \frac{n}{\varepsilon})$ time algorithm to compute a $1+\varepsilon$ geometric median. Here we made minimal effort to improve the log factors and plan to investigate this further in future work.

7.1.3 Overview of $O(d\varepsilon^{-2})$ Time Algorithm

In addition to providing a nearly linear time algorithm we provide a stand alone result on quickly computing a crude $(1+\varepsilon)$-approximate geometric median in Section 7.7. In particular, given an oracle for sampling a random $a^{(i)}$ we provide an $O(d\varepsilon^{-2})$, i.e. sublinear, time algorithm that computes such an approximate median. Our algorithm for this result is fairly straightforward. First, we show that random sampling can be used to obtain some constant approximate information about the optimal point in constant time. In particular we show how this can be used to deduce an Euclidean ball which contains the optimal point. Second, we perform stochastic subgradient descent within this ball to achieve our desired result.

7.1.4 Organization

The rest of the chapter is structured as follows. After covering preliminaries in Section 2.3, in Section 7.3 we provide various results about the central path that we use to derive our nearly linear time

algorithm. In Section 7.4 we then provide our nearly linear time algorithm. All the proofs and supporting lemmas for these sections are deferred to Section 7.5 and Section 7.6. In Section 7.7 we provide our $O(d/\varepsilon^2)$ algorithm, in Section 7.8 we provide the derivation of our penalized objective function, in Section 7.9 we provide general technical machinery we use throughout, and in Section 7.10 we show how to extend our results to Weber's problem, i.e. weighted geometric median.

7.2 Preliminaries

7.2.1 General Notation

For a symmetric positive semidefinite matrix (PSD), $\boldsymbol{A}$, we let $\lambda_1(\boldsymbol{A}) \geq ... \geq \lambda_n(\boldsymbol{A}) \geq 0$ denote the eigenvalues of $\boldsymbol{A}$ and let $v_1(\boldsymbol{A}), ..., v_n(\boldsymbol{A})$ denote corresponding eigenvectors.

7.2.2 Problem Notation

The central problem of this chapter is as follows: we are given points $a^{(1)}, ..., a^{(n)} \in \mathbb{R}^d$ and we wish to compute a geometric median, i.e. $x_* \in \arg\min_{x \in \mathbb{R}^d} f(x)$ where $f(x) = \sum_{i \in [n]} \|a^{(i)} - x\|_2$. We call a point $x \in \mathbb{R}^d$ an $(1+\varepsilon)$-approximate geometric median if $f(x) \leq (1+\varepsilon) f(x_*)$.

7.2.3 Penalized Objective Notation

To solve this problem we relax the objective function f and instead consider the following family of *penalized objective functions* parameterized by $t > 0$

$$\min_{x \in \mathbb{R}^d} f_t(x) \quad \text{where} \quad f_t(x) = \sum_{i \in [n]} \sqrt{1 + t^2 \|x - a^{(i)}\|_2^2} - \ln\left[1 + \sqrt{1 + t^2 \|x - a^{(i)}\|_2^2}\right]$$

This penalized objective function is derived from a natural interior point formulation of the geometric median problem (See Section 7.8). For all *path parameters* $t > 0$, we let $x_t \stackrel{\text{def}}{=} \arg\min_x f_t(x)$. Our primary goal is to obtain good approximations to the *central path* $\{x_t : t > 0\}$ for increasing values of t.

We let $g_t^{(i)}(x) \stackrel{\text{def}}{=} \sqrt{1 + t^2 \|x - a^{(i)}\|_2^2}$ and $f_t^{(i)}(x) \stackrel{\text{def}}{=} g_t^{(i)}(x) - \ln(1 + g_t^{(i)}(x))$ so $f_t(x) = \sum_{i \in [n]} f_t^{(i)}(x)$. We refer to the quantity $w_t(x) \stackrel{\text{def}}{=} \sum_{i \in [n]} \frac{1}{1 + g_t^{(i)}(x)}$ as *weight* as it is a natural measure of total contribution of the $a^{(i)}$ to $\nabla^2 f_t(x)$. We let

$$\bar{g}_t(x) \stackrel{\text{def}}{=} w_t(x) \left[\sum_{i \in [n]} \frac{1}{(1 + g_t^{(i)}(x_t)) g_t^{(i)}(x_t)}\right]^{-1}$$

denote a natural term that helps upper bound the rate of change of the central path. Furthermore, we let $u^{(i)}(x)$ denote the unit vector corresonding to $x - a^{(i)}$, i.e. $u^{(i)}(x) \stackrel{\text{def}}{=} x - a^{(i)} / \|x - a^{(i)}\|_2$ when $\|x - a^{(i)}\|_2 \neq 0$ and $u^{(i)}(x) \stackrel{\text{def}}{=} 0$ otherwise. Finally we let $\mu_t(x) \stackrel{\text{def}}{=} \lambda_d(\nabla^2 f_t(x))$, denote the minimum eigenvalue of $\nabla^2 f_t(x)$, and let $v_t(x)$ denote a corresponding eigenvector. To simplify notation we often drop the (x) in these definitions when $x = x_t$ and t is clear from context.

7.3 Properties of the Central Path

Here provide various facts regarding the penalized objective function and the central path. While we use the lemmas in this section throughout the chapter, the main contribution of this section is Lemma 7.3.5 in Section 7.3.3. There we prove that with the exception of a single direction, the change in the central path is small over a constant multiplicative change in the path parameter. In addition, we show that our penalized objective function is stable under changes in a $O(\frac{1}{t})$ Euclidean ball (Section 7.3.1), we bound the change in the Hessian over the central path (Section 7.3.2). and we relate $f(x_t)$ to $f(x_*)$ (Section 7.3.4).

7.3.1 How Much Does the Hessian Change in General?

Here, we show that the Hessian of the penalized objective function is stable under changes in a $O(\frac{1}{t})$ sized Euclidean ball. This shows that if we have a point which is close to a central path point in Euclidean norm, then we can use Newton method to find it.

Lemma 7.3.1. Suppose that $\|x - y\|_2 \leq \frac{\varepsilon}{t}$ with $\varepsilon \leq \frac{1}{20}$. Then, we have

$$(1 - 6\varepsilon^{2/3})\nabla^2 f_t(x) \preceq \nabla^2 f_t(y) \preceq (1 + 6\varepsilon^{2/3})\nabla^2 f_t(x).$$

7.3.2 How Much Does the Hessian Change Along the Path?

Here we bound how much the Hessian of the penalized objective function can change along the central path. First we provide the following lemma bound several aspects of the penalized objective function and proving that the weight, w_t, only changes by a small amount multiplicatively given small multiplicative changes in the path parameter, t.

Lemma 7.3.2. For all $t \geq 0$ and $i \in [n]$ the following hold

$$\left\| \frac{d}{dt} x_t \right\|_2 \leq \frac{1}{t^2} \bar{g}_t(x_t) \quad , \quad \left| \frac{d}{dt} g_t^{(i)}(x_t) \right| \leq \frac{1}{t} \left(g_t^{(i)}(x_t) + \bar{g}_t \right) \quad , \text{ and } \quad \left| \frac{d}{dt} w_t \right| \leq \frac{2}{t} w_t$$

Consequently, for all $t' \geq t$ we have that $\left(\frac{t}{t'}\right)^2 w_t \leq w_{t'} \leq \left(\frac{t'}{t}\right)^2 w_t$.

Next we use this lemma to bound the change in the Hessian with respect to t.

Lemma 7.3.3. For all $t \geq 0$ we have

$$-12 \cdot t \cdot w_t \boldsymbol{I} \preceq \frac{d}{dt} \left[\nabla^2 f_t(x_t) \right] \preceq 12 \cdot t \cdot w_t \boldsymbol{I} \tag{7.3}$$

and therefore for all $\beta \in [0, \frac{1}{8}]$

$$\nabla^2 f(x_t) - 15\beta t^2 w_t \boldsymbol{I} \preceq \nabla^2 f(x_{t(1+\beta)}) \preceq \nabla^2 f(x_t) + 15\beta t^2 w_t \boldsymbol{I} \,. \tag{7.4}$$

7.3.3 Where is the Next Optimal Point?

Here we prove our main result of this section. We prove that over a long step the central path moves very little in directions orthogonal to the smallest eigenvector of the Hessian. We begin by noting the Hessian is approximately a scaled identity minus a rank 1 matrix.

Lemma 7.3.4. For all t we have $\frac{1}{2}\left[t^2 \cdot w_t \boldsymbol{I} - (t^2 \cdot w_t - \mu_t) v_t v_t^\top\right] \preceq \nabla^2 f_t(x_t) \preceq t^2 \cdot w_t \boldsymbol{I} - (t^2 \cdot w_t - \mu_t) v_t v_t^\top$

Using this and the lemmas of the previous section we bound the amount x_t can move in every direction far from v_t.

Lemma 7.3.5 (The Central Path is Almost Straight). For all $t \geq 0$, $\beta \in [0, \frac{1}{600}]$, and any unit vector y with $|\langle y, v_t \rangle| \leq \frac{1}{t^2 \cdot \kappa}$ where $\kappa = \max_{\delta \in [t,(1+\beta)t]} \frac{w_\delta}{\mu_\delta}$, we have $y^\top (x_{(1+\beta)t} - x_t) \leq \frac{6\beta}{t}$.

7.3.4 Where is the End?

In this section, we bound the quality of the central path with respect to the geometric median objective. In particular, we show that if we can solve the problem for some $t = \frac{2n}{\varepsilon f(x_*)}$ then we obtain an $(1+\varepsilon)$-approximate solution. As our algorithm ultimately starts from an initial $t = 1/O(f(x_*))$ and increases t by a multiplicative constant in every iteration, this yields an $O(\log \frac{n}{\varepsilon})$ iteration algorithm.

Lemma 7.3.6. $f(x_t) - f(x_*) \leq \frac{2n}{t}$ for all $t > 0$.

7.4 Overview of the Algorithm

Algorithm 13: `AccurateMedian`(ε)

Input: desired accuracy $\varepsilon \in (0, 1)$

`// Compute a 2-approximate geometric median and use it to center`
Compute $x^{(0)} := \frac{1}{n} \sum_{i \in [n]} a^{(i)}$ and $\widetilde{f}_* := f(x^{(0)})$ `// Note` $\widetilde{f}_* \leq 2f(x_*)$ `by Lemma 7.5.6`
Let $t_i = \frac{1}{400 \widetilde{f}_*}(1 + \frac{1}{600})^{i-1}$, $\tilde{\varepsilon}_* = \frac{1}{3}\varepsilon$, and $\tilde{t}_* = \frac{2n}{\tilde{\varepsilon} \cdot \widetilde{f}_*}$
Let $\varepsilon_v = \frac{1}{8}(\frac{\tilde{\varepsilon}_*}{7n})^2$ and let $\varepsilon_c = (\frac{\varepsilon_v}{36})^{\frac{3}{2}}$
$x^{(1)} = \texttt{LineSearch}(x^{(0)}, t_1, t_1, 0, \varepsilon_c)$

`// Iteratively improve quality of approximation`
Let $k = \max_{i \in \mathbb{Z}} t_i \leq \tilde{t}_*$
for $i \in [1, k]$ **do**
  `// Compute` ε_v`-approximate minimum eigenvalue and eigenvector of` $\nabla^2 f_{t_i}(x^{(i)})$
  $(\lambda^{(i)}, u^{(i)}) = \texttt{ApproxMinEig}(x^{(i)}, t_i, \varepsilon_v)$.

  `// Line search to find` $x^{(i+1)}$ `such that` $\|x^{(i+1)} - x_{t_{i+1}}\|_2 \leq \frac{\varepsilon_c}{t_{i+1}}$
  $x^{(i+1)} = \texttt{LineSearch}(x^{(i)}, t_i, t_{i+1}, u^{(i)}, \varepsilon_c)$ w.
end
Output: ε-approximate geometric median $x^{(k)}$

Here we show how to use the structural results from the previous section to obtain a nearly linear time algorithm for computing the geometric median. Our algorithm follows a simple structure (See Algorithm 13). First we use simply average the $a^{(i)}$ to compute a $(1+\frac{1}{n})$-approximate median, denoted $x^{(0)}$. Then for a number of iterations we repeatedly move closer to x_t for some path parameter t, compute the minimum eigenvector of the Hessian, and line search in that direction to find an approximation to a point further along the central path. Ultimately, this yields a point $x^{(k)}$ that is precise enough approximation to a point along the central path with large enough t that we can simply out $x^{(k)}$ as our $(1+\varepsilon)$-approximate geometric median.

We split the remainder of the algorithm specification and its analysis into several parts. First in Section 7.4.1 we show how to compute an approximate minimum eigenvector and eigenvalue of the Hessian of the penalized objective function. Then in Section 7.4.2 we show how to use this eigenvector to line search for the next central path point. Finally, in Section 7.4.3 we put these results together to obtain our nearly linear time algorithm. Throughout this section we will want an upper bound to

$f(x_*)$ and a slight lower bound on ε, the geometric median accuracy we are aiming for. We use an easily computed $\tilde{f}_* \le 2f(x_*)$ for the former and $\tilde{\varepsilon}_* = \frac{1}{3}\varepsilon$ throughout the section.

7.4.1 Eigenvector Computation and Hessian Approximation

Here we show how to compute the minimum eigenvector of $\nabla^2 f_t(x)$ and thereby obtain a concise approximation to $\nabla^2 f_t(x)$. Our main algorithmic tool is the well known power method and the fact that it converges quickly on a matrix with a large eigenvalue gap. To improve our logarithmic terms we need a slightly non-standard analysis of the method and therefore we provide and analyze this method for completeness in Section 7.6.1. Using this tool we estimate the top eigenvector as follows.

Algorithm 14: `ApproxMinEig`(x, t, ε)

Input: Point $x \in \mathbb{R}^d$, path parameter t, and target accuracy ε.
Let $\boldsymbol{A} = \sum_{i\in[n]} \frac{t^4(x-a^{(i)})(x-a^{(i)})^\top}{(1+g_t^{(i)}(x))^2 g_t^{(i)}(x)}$
Let $u :=$ `PowerMethod`$(\boldsymbol{A}, \Theta(\log\left(\frac{n}{\varepsilon}\right)))$
Let $\lambda = u^\top \nabla^2 f_t(x) u$
Output: (λ, u)

Lemma 7.4.1 (Computing Hessian Approximation). Let $x \in \mathbb{R}^d$, $t > 0$, and $\varepsilon \in (0, \frac{1}{4})$. The algorithm `ApproxMinEig`$(x, t, \varepsilon)$ outputs (λ, u) in $O(nd \log \frac{n}{\varepsilon})$ time such that if $\mu_t(x) \le \frac{1}{4}t^2 w_t(x)$ then $\langle v_t(x), u\rangle^2 \ge 1 - \varepsilon$ with high probability in n. Furthermore, if $\varepsilon \le \frac{\mu_t(x)}{8t^2 \cdot w_t(x)}$ then $\frac{1}{4}\boldsymbol{Q} \preceq \nabla^2 f_t(x) \preceq 4\boldsymbol{Q}$ with high probability in n where $\boldsymbol{Q} \stackrel{\text{def}}{=} t^2 \cdot w_t(x) - \left(t^2 \cdot w_t(x) - \lambda\right) uu^\top$.

Furthermore, we show that the $v^{(i)}$ computed by this algorithm is sufficiently close to the bad direction. Combining 7.4.1 with the structural results from the previous section and Lemma 7.9.4, a minor technical lemma regarding the transitivity of large inner products,we provide the following lemma.

Lemma 7.4.2. Let $u =$ `ApproxMinEig`(x, t, ε_v) for $\varepsilon_v < \frac{1}{8}$ and x such that $\|x - x_t\|_2 \le \frac{\varepsilon_c}{t}$ for $\varepsilon_c \le (\frac{\varepsilon_v}{36})^{\frac{3}{2}}$. $\mu_t \le \frac{1}{4}t^2 \cdot w_t$. For all unit vectors $y \perp u$, we have $\langle y, v_t\rangle^2 \le 8\varepsilon_v$.

Note that this lemma assumes μ_t is small. When μ_t is large, we instead show that the next central path point is close to the current point and hence we do not need to compute the bad direction to center quickly.

Lemma 7.4.3. Suppose $\mu_t \ge \frac{1}{4}t^2 \cdot w_t$ and let $t' \in [t, (1 + \frac{1}{600})t]$ then $\|x_{t'} - x_t\|_2 \le \frac{1}{100t}$.

7.4.2 Line Searching

Here we show how to line search along the bad direction to find the next point on the central path. Unfortunately, it is not clear if you can binary search on the objective function directly. If we search over α to minimize $f_{t_{i+1}}(y^{(i)} + \alpha v^{(i)})$ directly it is unclear if we actually obtain a point close to x_{t+1}. It might be the case that even after minimizing α we would be unable to move towards x_{t+1} efficiently.

To overcome this difficulty, we use the fact that over the region $\|x - y\|_2 = O(\frac{1}{t})$ the Hessian changes by at most a constant and therefore we can minimize $f_t(x)$ over this region extremely quickly. Therefore, we instead line search on the following function

$$g_{t,y,v}(\alpha) \stackrel{\text{def}}{=} \min_{\|x-(y+\alpha v)\|_2 \le \frac{1}{49t}} f_t(x) \tag{7.5}$$

and use that we can evaluate $g_{t,y,v}(\alpha)$ approximately by using an appropriate centering procedure. We can show (See Lemma 7.9.6) that $g_{t,y,v}(\alpha)$ is convex and therefore we can minimize it efficiently just by doing an appropriate binary search. By finding the approximately minimizing α and outputting the corresponding approximately minimizing x, we can obtain $x^{(i+1)}$ that is close enough to $x_{t_{i+1}}$. For notational convenience, we simply write $g(\alpha)$ if t, y, v is clear from the context.

First, we show how we can locally center and provide error analysis for that algorithm.

Algorithm 15: `LocalCenter`(y, t, ε)

Input: Point $y \in \mathbb{R}^d$, path parameter $t > 0$, target accuracy $\varepsilon > 0$.
Let $(\lambda, v) := \texttt{ApproxMinEig}(x, t, \varepsilon)$.
Let $\boldsymbol{Q} = t^2 \cdot w_t(y)\boldsymbol{I} - \left(t^2 \cdot w_t(y) - \lambda\right) vv^\top$
Let $x^{(0)} = y$
for $i = 1, ..., k = 64 \log \frac{1}{\varepsilon}$ **do**
  Let $x^{(i)} = \min_{\|x-y\|_2 \le \frac{1}{49t}} f(x^{(i-1)}) + \langle \nabla f_t(x^{(i-1)}), x - x^{(i-1)} \rangle + 4\|x - x^{(i-1)}\|^2_{\boldsymbol{Q}}$.
end
Output: $x^{(k)}$

Lemma 7.4.4. Given some $y \in \mathbb{R}^d$, $t > 0$ and $\varepsilon \in (0, \min\{\frac{1}{4}, \frac{\mu_t(x)}{8t^2 \cdot w_t(x)}\})$. In $O(nd \log(\frac{n}{\varepsilon}))$ time `LocalCenter`(y, t, ε) computes $x^{(k)}$ such that with high probability in n/ε.

$$f_t(x^{(k)}) - \min_{\|x-y\|_2 \le \frac{1}{49t}} f_t(x) \le \varepsilon \left(f_t(y) - \min_{\|x-y\|_2 \le \frac{1}{49t}} f_t(x) \right).$$

Using this local centering algorithm as well as a general result for minimizing one dimensional convex functions using a noisy oracle (See Section 7.9.3) we obtain our line search algorithm.

Algorithm 16: `LineSearch`$(y, t, t', u, \varepsilon)$

Input: Point $y \in \mathbb{R}^d$, current path parameter t, next path parameter t', bad direction u, target accuracy ε
Let $\varepsilon_O = \frac{\varepsilon^2 \cdot \tilde{\varepsilon}_*^2}{300n}$, $\ell = -6\widetilde{f}_*$, $u = 6\widetilde{f}_*$.
Define the oracle $q : \mathbb{R} \to \mathbb{R}$ by $q(\alpha) = f_{t'}\left(\texttt{LocalCenter}\left(y + \alpha u, t', \varepsilon_O\right)\right)$
Let $\alpha' = \texttt{OneDimMinimizer}(\ell, u, \varepsilon_O, q, tn)$
Output: $x' = \texttt{LocalCenter}\left(y + \alpha u, t', \varepsilon_O\right)$

Lemma 7.4.5. Let $\frac{1}{400 f(x_*)} \le t \le t' \le (1 + \frac{1}{600})t \le \frac{2n}{\tilde{\varepsilon}_* \cdot \tilde{f}_*}$ and let $u = \texttt{ApproxMinEig}(y, t, \varepsilon_v)$ for $\varepsilon_v \le \frac{1}{8}(\frac{\tilde{\varepsilon}_*}{7n})^2$ and $y \in \mathbb{R}^d$ such that $\|y - x_t\|_2 \le \frac{1}{t}(\frac{\varepsilon_v}{36})^{\frac{3}{2}}$. In $O(nd \log^2(\frac{n}{\tilde{\varepsilon} \cdot \varepsilon \cdot \varepsilon_v}))$ time, `LineSearch`$(y, t, t', u, \varepsilon)$ outputs x' such that $\|x' - x_{t'}\|_2 \le \frac{\varepsilon}{t'}$ with high probability in n.

We also provide the following lemma useful for finding the first center.

Lemma 7.4.6. Let $\frac{1}{400 f(x_*)} \le t \le t' \le (1 + \frac{1}{600})t \le \frac{2n}{\tilde{\varepsilon}_* \cdot \tilde{f}_*}$ and let $x \in \mathbb{R}^d$ satisfy $\|x - x_t\|_2 \le \frac{1}{100t}$. Then, in $O(nd \log^2(\frac{n}{\varepsilon \cdot \tilde{\varepsilon}_*}))$ time, `LineSearch`$(x, t, t, u, \varepsilon)$ output y such that $\|y - x_t\|_2 \le \frac{\varepsilon}{t}$ for any vector $u \in \mathbb{R}^d$.

7.4.3 Putting It All Together

Combining the results of the previous sections to prove our main theorem.

Theorem 7.4.1. *In $O(nd\log^3(\frac{n}{\varepsilon}))$ time, Algorithm 13 outputs an $(1+\varepsilon)$-approximate geometric median with constant probability.*

7.5 Analysis of the Central Path

Here we provide proofs of the claims in Section 7.3 as well as additional technical lemmas we use throughout the chapter.

7.5.1 Basic Facts

Here we provide basic facts regarding the central path that we will use throughout our analysis. First we compute various derivatives of the penalized objective function.

Lemma 7.5.1 (Path Derivatives). We have

$$\nabla f_t(x) = \sum_{i\in[n]} \frac{t^2(x-a^{(i)})}{1+g_t^{(i)}(x)} \quad , \quad \nabla^2 f_t(x) = \sum_{i\in[n]} \frac{t^2}{1+g_t^{(i)}(x)}\left(\boldsymbol{I} - \frac{t^2(x-a^{(i)})(x-a^{(i)})^\top}{g_t^{(i)}(x)(1+g_t^{(i)}(x))}\right) \quad , \text{ and}$$

$$\frac{d}{dt}x_t = -\left(\nabla^2 f_t(x_t)\right)^{-1} \sum_{i\in[n]} \frac{t(x_t-a^{(i)})}{(1+g_t^{(i)}(x_t))g_t^{(i)}(x_t)}$$

Proof of Lemma 7.5.1. Direct calculation shows that

$$\begin{aligned}\nabla f_t^{(i)}(x) &= \frac{t^2(x-a^{(i)})}{\sqrt{1+t^2\|x-a^{(i)}\|_2^2}} - \frac{1}{1+\sqrt{1+t^2\|x-a^{(i)}\|_2^2}}\left(\frac{t^2(x-a^{(i)})}{\sqrt{1+t^2\|x-a^{(i)}\|_2^2}}\right)\\ &= \frac{t^2(x-a^{(i)})}{1+\sqrt{1+t^2\|x-a^{(i)}\|_2^2}} = \frac{t^2(x-a^{(i)})}{1+g_t^{(i)}(x)}\end{aligned}$$

and

$$\begin{aligned}\nabla^2 f_t^{(i)}(x) &= \frac{t^2}{1+\sqrt{1+t^2\|x-a^{(i)}\|_2^2}}\boldsymbol{I} - \left(\frac{1}{1+\sqrt{1+t^2\|x-a^{(i)}\|_2^2}}\right)^2 \frac{t^4(x-a^{(i)})(x-a^{(i)})^\top}{\sqrt{1+t^2\|x-a^{(i)}\|_2^2}}\\ &= \frac{t^2}{1+g_t^{(i)}(x)}\left(\boldsymbol{I} - \frac{t^2(x-a^{(i)})(x-a^{(i)})^\top}{g_t^{(i)}(x)(1+g_t^{(i)}(x))}\right)\end{aligned}$$

and

$$\begin{aligned}\left(\frac{d}{dt}\nabla f_t^{(i)}\right)(x) &= \frac{2t(x-a^{(i)})}{1+\sqrt{1+t^2\|x-a^{(i)}\|_2^2}} - \frac{t^2\cdot(x-a^{(i)})\cdot t\|x-a^{(i)}\|_2^2}{\left(1+\sqrt{1+t^2\|x-a^{(i)}\|}\right)^2\sqrt{1+t^2\|x-a^{(i)}\|_2^2}}\\ &= \frac{t\cdot(x-a^{(i)})}{1+g_t^{(i)}(x)}\left(2 - \frac{g_t^{(i)}(x)^2-1}{(1+g_t^{(i)}(x))g_t^{(i)}(x)}\right)\\ &= \frac{t\cdot(x-a^{(i)})}{1+g_t^{(i)}(x)}\left(\frac{2g_t^{(i)}(x)-(g_t^{(i)}(x)-1)}{g_t^{(i)}(x)}\right) = \frac{t\cdot(x-a^{(i)})}{g_t^{(i)}(x)}\end{aligned}$$

Finally, by the optimality of x_t we have that $\nabla f_t(x_t) = 0$. Consequently,

$$\nabla^2 f_t(x_t)\frac{d}{dt}x_t + \left(\frac{d}{dt}\nabla f_t\right)(x_t) = 0.$$

and solving for $\frac{d}{dt}x_t$ then yields

$$\begin{aligned}\frac{d}{dt}x_t &= -\left(\nabla^2 f_t(x_t)\right)^{-1}\left(\left(\frac{d}{dt}\nabla f_t\right)(x_t)\right)\\ &= -\left(\nabla^2 f_t(x_t)\right)^{-1}\left(\left(\frac{d}{dt}\nabla f_t\right)(x_t) - \frac{1}{t}\nabla f_t(x_t)\right)\\ &= -\left(\nabla^2 f_t(x_t)\right)^{-1}\left(\sum_{i\in[n]}\left[\frac{t}{g_t^{(i)}} - \frac{t}{1+g_t^{(i)}}\right](x_t - a^{(i)})\right).\end{aligned}$$

□

Next, in we provide simple facts regarding the Hessian of the penalized objective function.

Lemma 7.5.2. For all $t > 0$ and $x \in \mathbb{R}^d$

$$\nabla^2 f_t(x) = \sum_{i\in[n]}\frac{t^2}{1+g_t^{(i)}(x)}\left(\boldsymbol{I} - \left(1 - \frac{1}{g_t^{(i)}(x)}\right)u^{(i)}(x)u^{(i)}(x)^\top\right)$$

and therefore

$$\sum_{i\in[n]}\frac{t^2}{(1+g_t^{(i)}(x))g_t^{(i)}(x)}\boldsymbol{I} \preceq \nabla^2 f_t(x) \preceq \sum_{i\in[n]}\frac{t^2}{1+g_t^{(i)}(x)}\boldsymbol{I}$$

Proof of Lemma 7.5.2. We have that

$$\begin{aligned}\nabla^2 f_t(x) &= \sum_{i\in[n]}\frac{t^2}{1+g_t^{(i)}(x)}\left(\boldsymbol{I} - \frac{t^2(x-a^{(i)})(x-a^{(i)})^\top}{g_t^{(i)}(x)(1+g_t^{(i)}(x))}\right)\\ &= \sum_{i\in[n]}\frac{t^2}{1+g_t^{(i)}(x)}\left(\boldsymbol{I} - \frac{t^2\|x-a^{(i)}\|_2^2}{(1+g_t^{(i)}(x))g_t^{(i)}(x)}u^{(i)}(x)u^{(i)}(x)^\top\right)\\ &= \sum_{i\in[n]}\frac{t^2}{1+g_t^{(i)}(x)}\left(\boldsymbol{I} - u^{(i)}(x)u^{(i)}(x)^\top\right)\end{aligned}$$

Since

$$\frac{g_t^{(i)}(x)^2 - 1}{g_t^{(i)}(x)(1+g_t^{(i)}(x))} = \frac{(g_t^{(i)}(x)+1)(g_t^{(i)}(x)-1)}{(1+g_t^{(i)}(x))g_t^{(i)}(x)} = 1 - \frac{1}{g_t^{(i)}(x)}$$

the result follows. □

7.5.2 Stability of Hessian

Here we show that moving a point $x \in \mathbb{R}^d$ in ℓ_2, does not change the Hessian, $\nabla^2 f_t(x)$, too much spectrally. First we show that such changes do not change $g_t^{(i)}(x)$ by too much (Lemma 7.5.3) and then we use this to prove the claim, i.e. we prove Lemma 7.3.1.

Lemma 7.5.3 (Stability of g). For all $x, y \in \mathbb{R}^d$ and $t > 0$, we have

$$g_t^{(i)}(x) - t\|x-y\|_2 \le g_t^{(i)}(y) \le g_t^{(i)}(x) + t\|x-y\|_2$$

Proof of Lemma 7.5.3. Direct calculation reveals that

$$\begin{aligned} g_t^{(i)}(y)^2 &= 1 + t^2\|x - a^{(i)} + y - x\|_2^2 \\ &= 1 + t^2\|x-a^{(i)}\|_2^2 + 2t^2(x-a^{(i)})^\top(y-x) + t^2\|y-x\|_2^2 \\ &= g_t^{(i)}(x)^2 + 2t^2(x-a^{(i)})^\top(y-x) + t^2\|y-x\|_2^2 . \end{aligned}$$

Consequently by Cauchy Schwarz

$$g_t^{(i)}(y)^2 \le g_t^{(i)}(x)^2 + 2t^2\|x-a^{(i)}\|_2 \cdot \|y-x\|_2 + t^2\|y-x\|_2^2 \le \left(g_t^{(i)}(x) + t\|y-x\|_2\right)^2$$

and

$$g_t^{(i)}(y)^2 \ge g_t^{(i)}(x)^2 - 2t^2\|x-a^{(i)}\|_2 \cdot \|y-x\|_2 + t^2\|y-x\|_2^2 \ge \left(g_t^{(i)}(x) - t\|y-x\|_2\right)^2 .$$

□

Lemma 7.3.1. Suppose that $\|x-y\|_2 \le \frac{\varepsilon}{t}$ with $\varepsilon \le \frac{1}{20}$. Then, we have

$$(1-6\varepsilon^{2/3})\nabla^2 f_t(x) \preceq \nabla^2 f_t(y) \preceq (1+6\varepsilon^{2/3})\nabla^2 f_t(x).$$

Proof of Lemma 7.3.1. Here we prove the following stronger statement, for all $i \in [n]$

$$(1-6\varepsilon^{2/3})\nabla^2 f_t^{(i)}(x) \preceq \nabla^2 f_t^{(i)}(y) \preceq (1+6\varepsilon^{2/3})\nabla^2 f_t^{(i)}(x) .$$

Without loss of generality let $y - x = \alpha v + \beta u^{(i)}(x)$ for some $v \perp u^{(i)}(x)$ with $\|v\|_2 = 1$. Since $\|x-y\|_2^2 \le \frac{\varepsilon^2}{t^2}$, we know that $\alpha^2, \beta^2 \le \frac{\varepsilon^2}{t^2}$. Also, let $\bar{x} = x + \beta u^{(i)}(x)$, so that clearly, $u^{(i)}(x) = u^{(i)}(\bar{x})$. Now some manipulation reveals that for all unit vectors $z \in \mathbb{R}^d$ the following holds (so long as $u^{(i)}(x) \neq 0$ and $u^{(i)}(y) \neq 0$)

$$\begin{aligned} &\left| \left[u^{(i)}(x)^\top z\right]^2 - \left[u^{(i)}(y) z\right]^2 \right| \\ &= \left| \left[u^{(i)}(\bar{x})^\top z\right]^2 - \left[u^{(i)}(y) z\right]^2 \right| \\ &= \left| \left[\frac{(\bar{x}-a^{(i)})^\top z}{\|\bar{x}-a^{(i)}\|_2}\right]^2 - \left[\frac{(y-a^{(i)})^\top z}{\|y-a^{(i)}\|_2}\right]^2 \right| \\ &\le \left| \left[\frac{(\bar{x}-a^{(i)})^\top z}{\|\bar{x}-a^{(i)}\|_2}\right]^2 - \left[\frac{(\bar{x}-a^{(i)})^\top z}{\|y-a^{(i)}\|_2}\right]^2 \right| + \left| \left[\frac{(\bar{x}-a^{(i)})^\top z}{\|y-a^{(i)}\|_2}\right]^2 - \left[\frac{(y-a^{(i)})^\top z}{\|y-a^{(i)}\|_2}\right]^2 \right| \\ &\le \left| 1 - \frac{\|\bar{x}-a^{(i)}\|_2^2}{\|y-a^{(i)}\|_2^2} \right| + \frac{\left| \left[(\bar{x}-a^{(i)} + \alpha v^\top z\right]^2 - \left[(\bar{x}-a^{(i)})^\top z\right]^2 \right|}{\|y-a^{(i)}\|_2^2} \\ &= \frac{\alpha^2 + \left| 2\left[(\bar{x}-a^{(i)})^\top z\right] \cdot \left[\alpha v^\top z\right] + \left[\alpha v^\top z\right]^2 \right|}{\|\bar{x}-a^{(i)}\|_2^2 + \alpha_i^2} . \end{aligned}$$

Where we used that $y = \bar{x} + \alpha v$ and $\|y - a^{(i)}\|_2^2 = \alpha^2 + \|\bar{x} - a^{(i)}\|_2^2$ (since $v \perp (\bar{x} - a^{(i)})$). Now we know that $\alpha^2 \leq \frac{\varepsilon^2}{t^2}$ and and therefore, by Young's inequality and Cauchy Schwarz we have that for all $\gamma > 0$

$$\begin{aligned}
\left|\left[u^{(i)}(x)^\top z\right]^2 - \left[u^{(i)}(y) z\right]^2\right| &\leq \frac{2\alpha^2 + 2\left|\left[(\bar{x} - a^{(i)})^\top z\right] \cdot \left[\alpha v^\top z\right]\right|}{\|\bar{x} - a^{(i)}\|_2^2 + \alpha^2} \\
&\leq \frac{2\alpha^2 + \gamma\left[(\bar{x} - a^{(i)})^\top z\right]^2 + \gamma^{-1}\alpha^2\left[v^\top z\right]^2}{\|\bar{x} - a^{(i)}\|_2^2 + \alpha^2} \\
&\leq \frac{\alpha^2\left(2 + \gamma^{-1}\left(v^\top z\right)^2\right)}{\|\bar{x} - a^{(i)}\|_2^2 + \alpha^2} + \gamma\left[(u^{(i)}(x))^\top z\right]^2 \\
&\leq \frac{\varepsilon^2}{t^2\|\bar{x} - a^{(i)}\|_2^2 + \varepsilon^2}\left(2 + \frac{1}{\gamma}\left(v^\top z\right)^2\right) + \gamma\left[(u^{(i)}(x))^\top z\right]^2 . \qquad (7.6)
\end{aligned}$$

Note that

$$\begin{aligned}
t^2\|\bar{x} - a^{(i)}\|_2^2 &= t^2\left(\|x - a^{(i)}\|_2^2 + 2\beta(x - a^{(i)})^\top u^{(i)}(x) + \beta^2\right) = \left(t\|x - a^{(i)}\|_2 + t\beta\right)^2 \\
&\geq \left(\max\left\{t\|x - a^{(i)}\|_2 - \varepsilon, 0\right\}\right)^2 .
\end{aligned}$$

Consequently, if $t\|x - a^{(i)}\|_2 \geq 2\varepsilon^{1/3}\sqrt{g_t^{(i)}(x)}$ then since $\varepsilon \leq \frac{1}{20}$ we have that $\|x - a^{(i)}\|_2 \geq 2\varepsilon$ and $\|y - a^{(i)}\| \geq \varepsilon$, justifying our assumption that $u^{(i)}(x) \neq 0$ and $u^{(i)}(y) \neq 0$. Furthermore, this implies that $t^2\|\bar{x} - a^{(i)}\|_2^2 \geq 2\varepsilon^{2/3} g_t^{(i)}(x)$ and therefore letting $\gamma = \frac{\varepsilon^{2/3}}{g_t^{(i)}(x)}$ yields

$$\begin{aligned}
\left|\left[u_t^{(i)}(x)^\top z\right]^2 - \left[u_t^{(i)}(y) z\right]^2\right| &\leq \frac{\varepsilon^{4/3}}{2g_t^{(i)}(x)}\left(2 + \frac{g_t^{(i)}(x)}{\varepsilon^{2/3}}\left[v^\top z\right]^2\right) + \frac{\varepsilon^{2/3}}{g_t^{(i)}(x)}\left[(u^{(i)}(x))^\top z\right]^2 \\
&\leq 2\varepsilon^{2/3}\left[v^\top z\right]^2 + \frac{\varepsilon^{2/3}}{g_t^{(i)}(x)}\left[(u^{(i)}(x))^\top z\right]^2 \\
&\leq 2\varepsilon^{2/3}\left(\frac{1 + g_t^{(i)}(x)}{t^2}\right)\|z\|^2_{\nabla^2 f_t^{(i)}(x)}
\end{aligned}$$

and therefore if we let

$$\boldsymbol{H} \stackrel{\text{def}}{=} \frac{t^2}{1 + g_t^{(i)}(x)}\left(\boldsymbol{I} - \left(1 - \frac{1}{g_t^{(i)}(x)}\right) u^{(i)}(y)(u^{(i)}(y))^\top\right),$$

we see that for unit vectors z,

$$\left|z^\top\left(\boldsymbol{H} - \nabla^2 f_t^{(i)}(x)\right) z\right| \leq 2\varepsilon^{2/3}\|z\|^2_{\nabla^2 f_t^{(i)}(x)}$$

Otherwise, $t\|x - a^{(i)}\|_2 < 2\varepsilon^{1/3}\sqrt{g_t^{(i)}(x)}$ and therefore

$$g_t^{(i)}(x)^2 = 1 + t^2\|x - a^{(i)}\|_2^2 \leq 1 + 4\varepsilon^{2/3} g_t^{(i)}(x)$$

Therefore, we have

$$g_t^{(i)}(x) \quad \leq \quad \frac{4\varepsilon^{2/3} + \sqrt{(4\varepsilon^{2/3})^2 + 4}}{2} \leq 1 + 4\varepsilon^{2/3} .$$

Therefore independent of (7.6) and the assumption that $u^{(i)}(x) \neq 0$ and $u^{(i)}(y) \neq 0$ we have

$$\frac{1}{1+4\varepsilon^{2/3}}\boldsymbol{H} \preceq \frac{t^2}{(1+g_t^{(i)}(x))g_t^{(i)}(x)}\boldsymbol{I} \preceq \nabla^2 f_t^{(i)}(x) \preceq \frac{t^2}{(1+g_t^{(i)}(x))}\boldsymbol{I} \preceq \left(1+4\varepsilon^{2/3}\right)\boldsymbol{H}\,.$$

In either case, we have that

$$\left|z^\top\left(\boldsymbol{H}-\nabla^2 f_t^{(i)}(x)\right)z\right| \leq 4\varepsilon^{2/3}\|z\|^2_{\nabla^2 f_t^{(i)}(x)}.$$

Now, we note that $\|x-y\|_2 \leq \frac{\varepsilon}{t} \leq \varepsilon \cdot \frac{g_t^{(i)}(x)}{t}$. Therefore, by Lemma 7.5.3 we have that

$$(1-\varepsilon)g_t^{(i)}(x) \leq g_t^{(i)}(y) \leq (1+\varepsilon)g_t^{(i)}(x)$$

Therefore, we have

$$\frac{1-4\varepsilon^{2/3}}{(1+\varepsilon)^2}\nabla^2 f_t^{(i)}(x) \preceq \frac{1}{(1+\varepsilon)^2}\boldsymbol{H} \preceq \nabla^2 f_t^{(i)}(y) \preceq \frac{1}{(1-\varepsilon)^2}\boldsymbol{H} \preceq \frac{1+4\varepsilon^{2/3}}{(1-\varepsilon)^2}\nabla^2 f_t^{(i)}(x)$$

Since $\varepsilon < \frac{1}{20}$, the result follows. □

Consequently, so long as we have a point within a $O(\frac{1}{t})$ sized Euclidean ball of some x_t Newton's method (or an appropriately transformed first order method) within the ball will converge quickly.

■ 7.5.3 How Much Does the Hessian Change Along the Path?

Lemma 7.3.2. For all $t \geq 0$ and $i \in [n]$ the following hold

$$\left\|\frac{d}{dt}x_t\right\|_2 \leq \frac{1}{t^2}\bar{g}_t(x_t) \quad , \quad \left|\frac{d}{dt}g_t^{(i)}(x_t)\right| \leq \frac{1}{t}\left(g_t^{(i)}(x_t)+\bar{g}_t\right) \quad , \text{ and } \quad \left|\frac{d}{dt}w_t\right| \leq \frac{2}{t}w_t$$

Consequently, for all $t' \geq t$ we have that $\left(\frac{t}{t'}\right)^2 w_t \leq w_{t'} \leq \left(\frac{t'}{t}\right)^2 w_t$.

Proof of Lemma 7.3.2. From Lemma 7.5.1 we know that

$$\frac{d}{dt}x_t = -\left(\nabla^2 f_t(x_t)\right)^{-1}\sum_{i\in[n]}\frac{t(x_t-a^{(i)})}{(1+g_t^{(i)}(x_t))g_t^{(i)}(x_t)}$$

and by Lemma 7.5.2 we know that

$$\nabla^2 f_t(x_t) \succeq \sum_{i\in[n]}\frac{t^2}{(1+g_t^{(i)}(x_t))g_t^{(i)}(x_t)}\boldsymbol{I} = \frac{t^2}{\bar{g}_t(x_t)}\sum_{i\in[n]}\frac{1}{1+g_t^{(i)}(x_t)}\boldsymbol{I}\,.$$

Using this fact and the fact that $t\|x_t-a^{(i)}\|_2 \leq g_t^{(i)}$ we have

$$\begin{aligned}\left\|\frac{d}{dt}x_t\right\|_2 &= \left\|-\left(\nabla^2 f_t(x_t)\right)^{-1}\frac{d}{dt}\nabla f_t(x_t)\right\|_2 \\ &\leq \left(\frac{t^2}{\bar{g}_t(x_t)}\sum_{i\in[n]}\frac{1}{1+g_t^{(i)}(x_t)}\right)^{-1}\sum_{i\in[n]}\left\|\frac{t(x_t-a^{(i)})}{g_t^{(i)}(x_t)(1+g_t^{(i)}(x_t))}\right\|_2 \leq \frac{\bar{g}_t(x_t)}{t^2}\,.\end{aligned}$$

Next, we have

$$\begin{aligned}\frac{d}{dt}g_t^{(i)}(x_t) &= \frac{d}{dt}\left(1+t^2\|x_t-a^{(i)}\|_2^2\right)^{\frac{1}{2}}\\ &= \frac{1}{2}\cdot g_t^{(i)}(x_t)^{-1}\left(2t\|x_t-a^{(i)}\|_2^2+2t^2(x_t-a^{(i)})^\top\frac{d}{dt}x_t\right)\end{aligned}$$

which by Cauchy Schwarz and that $t\|x_t-a^{(i)}\|_2\le g_t^{(i)}(x_t)$ yields the second equation. Furthermore,

$$\begin{aligned}\left|\frac{d}{dt}w_t\right| &= \left|\frac{d}{dt}\sum_{i\in[n]}\frac{1}{1+g_t^{(i)}(x_t)}\right| \le \sum_{i\in[n]}\left|\frac{d}{dt}\frac{1}{1+g_t^{(i)}(x_t)}\right| = \sum_{i\in[n]}\left|\frac{1}{(1+g_t(x_t))^2}\frac{d}{dt}g_t(x_t)\right|\\ &\le \frac{1}{t}\sum_{i\in[n]}\frac{g_t^{(i)}(x_t)+\bar{g}_t}{(1+g_t^{(i)}(x_t))g_t^{(i)}(x_t)} \le 2\frac{w_t}{t}\end{aligned}$$

which yields the third equation.

Finally, using our earlier results and Jensen's inequality yields that

$$|\ln w_{t'}-\ln w_t| = \left|\int_t^{t'}\frac{\frac{d}{d\alpha}w_\alpha}{w_\alpha}d\alpha\right| \le \int_t^{t'}\frac{\left(2\frac{w_\alpha}{\alpha}\right)}{w_\alpha}d\alpha = 2\int_t^{t'}\frac{1}{\alpha}d\alpha = \ln\left(\frac{t'}{t}\right)^2.$$

Exponentiating the above inequality yields the final inequality. □

Lemma 7.3.3. For all $t\ge 0$ we have

$$-12\cdot t\cdot w_t\boldsymbol{I} \preceq \frac{d}{dt}\left[\nabla^2 f_t(x_t)\right] \preceq 12\cdot t\cdot w_t\boldsymbol{I} \tag{7.3}$$

and therefore for all $\beta\in[0,\frac{1}{8}]$

$$\nabla^2 f(x_t)-15\beta t^2 w_t\boldsymbol{I} \preceq \nabla^2 f(x_{t(1+\beta)}) \preceq \nabla^2 f(x_t)+15\beta t^2 w_t\boldsymbol{I}\,. \tag{7.4}$$

Proof of Lemma 7.3.3. Let

$$\boldsymbol{A}_t^{(i)} \stackrel{\text{def}}{=} \frac{t^2(x_t-a^{(i)})(x_t-a^{(i)})^\top}{(1+g_t^{(i)})g_t^{(i)}}$$

and recall that $\nabla^2 f_t(x_t)=\sum_{i\in[n]}\frac{t^2}{1+g_t^{(i)}}\left(\boldsymbol{I}-\boldsymbol{A}_t^{(i)}\right)$. Consequently

$$\begin{aligned}\frac{d}{dt}\nabla^2 f_t(x_t) &= \frac{d}{dt}\left(\sum_{i\in[n]}\frac{t^2}{1+g_t^{(i)}}\left(\boldsymbol{I}-\boldsymbol{A}_t^{(i)}\right)\right)\\ &= 2t\left(\frac{1}{t^2}\right)\nabla^2 f_t(x_t)+t^2\sum_{i\in[n]}\frac{-\frac{d}{dt}g_t^{(i)}}{(1+g_t^{(i)})^2}\left(\boldsymbol{I}-\boldsymbol{A}_t^{(i)}\right)-\sum_{i\in[n]}\frac{t^2}{1+g_t^{(i)}}\frac{d}{dt}\boldsymbol{A}_t^{(i)}\end{aligned}$$

Now, since $\boldsymbol{0}\preceq\boldsymbol{A}_t^{(i)}\preceq\boldsymbol{I}$ we have $0\preceq\nabla^2 f_t(x_t)\preceq t^2 w_t\boldsymbol{I}$. For all unit vectors v, using Lemma 7.3.2

yields

$$\left|v^\top\left(\frac{d}{dt}\nabla^2 f_t(x_t)\right)v\right| \le 2t\cdot w_t\cdot\|v\|_2^2 + t^2\sum_{i\in[n]}\frac{\left|\frac{d}{dt}g_t^{(i)}\right|}{(1+g_t^{(i)})^2}\|v\|_2^2 + \sum_{i\in[n]}\frac{t^2}{1+g_t^{(i)}}\left|v^\top\left(\frac{d}{dt}\boldsymbol{A}_t^{(i)}\right)v\right|$$
$$\le 4t\cdot w_t + \sum_{i\in[n]}\frac{t^2}{1+g_t^{(i)}}\left|v^\top\left(\frac{d}{dt}\boldsymbol{A}_t^{(i)}\right)v\right|.$$

Next

$$\frac{d}{dt}\boldsymbol{A}_t^{(i)} = 2t\left(\frac{1}{t^2}\right)\boldsymbol{A}_t^{(i)} - \left(\frac{t}{(1+g_t^{(i)})g_t^{(i)}}\right)^2\left[(1+g_t^{(i)})\frac{d}{dt}g_t^{(i)} + g_t^{(i)}\frac{d}{dt}g_t^{(i)}\right](x_t-a^{(i)})(x_t-a^{(i)})^\top$$
$$+\frac{t^2}{(1+g_t^{(i)})g_t^{(i)}}\left[(x_t-a^{(i)})\left(\frac{d}{dt}x_t\right)^\top + \left(\frac{d}{dt}x_t\right)(x_t-a^{(i)})^\top\right],$$

and therefore by Lemma 7.3.2 and the that $t\|x_t-a^{(i)}\|_2 \le g_t^{(i)}$ we have

$$\left|v^\top\left(\frac{d}{dt}\boldsymbol{A}_t^{(i)}\right)v\right| \le \left(\frac{2}{t} + \frac{2t^2\left|\frac{d}{dt}g_t^{(i)}\right|}{(1+g_t^{(i)})(g_t^{(i)})^2}\|x_t-a^{(i)}\|_2^2 + \frac{2t^2\|x_t-a^{(i)}\|_2\|\frac{d}{dt}x_t\|_2}{(1+g_t^{(i)})g_t^{(i)}}\right)\|v\|_2^2$$
$$\le \frac{2}{t} + \frac{2}{t}\cdot\frac{g_t^{(i)}+\bar{g}_t}{1+g_t^{(i)}} + \frac{2}{t}\cdot\frac{\bar{g}_t}{1+g_t^{(i)}} \le \frac{4}{t} + \frac{4}{t}\frac{\bar{g}_t}{1+g_t^{(i)}}.$$

Consequently, we have

$$\left|v^\top\left(\frac{d}{dt}\nabla^2 f_t(x_t)\right)v\right| \le 8t\cdot w_t + 4t\sum_{i\in[n]}\frac{\bar{g}_t}{(1+g_t^{(i)})^2} \le 12t\cdot w_t$$

which completes the proof of (7.3). To prove (7.4), let v be any unit vector and note that

$$\left|v^\top\left(\nabla^2 f_{t(1+\beta)}(x) - \nabla^2 f_t(x)\right)v\right| = \left|\int_t^{t(1+\beta)} v^\top\frac{d}{dt}\left[\nabla^2 f_\alpha(x_\alpha)\right]v\cdot d\alpha\right| \le 12\int_t^{t(1+\beta)}\alpha\cdot w_\alpha d\alpha$$
$$\le 12\int_t^{t(1+\beta)}\alpha\left(\frac{\alpha}{t}\right)^2 w_t d\alpha \le \frac{12}{t^2}\left(\frac{1}{4}\left[t(1+\beta)\right]^4 - \frac{1}{4}t^4\right)w_t$$
$$= 3t^2\left[(1+\beta)^4-1\right]w_t \le 15t^2\beta w_t$$

where we used Lemma 7.3.3 and $0\le\beta\le\frac{1}{8}$ at the last line. □

7.5.4 Where is the next Optimal Point?

Lemma 7.5.4. For all t we have $\frac{1}{2}\left[t^2\cdot w_t\boldsymbol{I} - (t^2\cdot w_t-\mu_t)v_tv_t^\top\right] \preceq \nabla^2 f_t(x_t) \preceq t^2\cdot w_t\boldsymbol{I} - (t^2\cdot w_t-\mu_t)v_tv_t^\top$

Proof of Lemma 7.5.4. This follows immediately from Lemma 7.5.2, regarding the hessian of the penalized objective function, and Lemma 7.9.1, regarding the sum of PSD matrices expressed as the identity matrix minus a rank 1 matrix. □

Lemma 7.3.5 (The Central Path is Almost Straight). For all $t\ge 0$, $\beta\in[0,\frac{1}{600}]$, and any unit vector y with $|\langle y,v_t\rangle| \le \frac{1}{t^2\cdot\kappa}$ where $\kappa = \max_{\delta\in[t,(1+\beta)t]}\frac{w_\delta}{\mu_\delta}$, we have $y^\top(x_{(1+\beta)t}-x_t)\le\frac{6\beta}{t}$.

Proof of Lemma 7.3.5. Clearly

$$\begin{aligned} y^\top(x_{(1+\beta)t} - x_t) &= \int_t^{(1+\beta)t} y^\top \frac{d}{d\alpha} x_\alpha d_\alpha \le \int_\beta^{(1+\beta)t} \left| y^\top \frac{d}{d\alpha} x_\alpha \right| d_\alpha \\ &\le \int_t^{(1+\beta)t} \left| y^\top \left(\nabla^2 f_\alpha(x_\alpha)\right)^{-1} \sum_{i\in[n]} \frac{\alpha}{(1+g_\alpha^{(i)})g_\alpha^{(i)}} (x_\alpha - a^{(i)}) \right| d_\alpha \\ &\le \int_t^{(1+\beta)t} \| \left(\nabla^2 f_\alpha(x_\alpha)\right)^{-1} y\|_2 \cdot \| \sum_{i\in[n]} \frac{\alpha}{(1+g_\alpha^{(i)})g_\alpha^{(i)}} (x_\alpha - a^{(i)})\|_2 d_\alpha \end{aligned}$$

Now since clearly $\alpha\|x_\alpha - a^{(i)}\|_2 \le g_\alpha^{(i)}$, invoking Lemma 7.3.2 yields that

$$\left\| \sum_{i\in[n]} \frac{\alpha(x_\alpha - a^{(i)})}{(1+g_\alpha^{(i)})g_\alpha^{(i)}} \right\|_2 \le \sum_{i\in[n]} \frac{1}{1+g_\alpha^{(i)}} = w_\alpha \le \left(\frac{\alpha}{t}\right)^2 w_t \,.$$

Now by invoking Lemma 7.3.3 and the Lemma 7.5.4, we have that

$$\nabla^2 f_\alpha(x_\alpha) \succeq \nabla^2 f_t(x_t) - 15\beta t^2 w_t \boldsymbol{I} \succeq \frac{1}{2}\left[t^2 \cdot w_t \boldsymbol{I} - (t^2 \cdot w_t - \mu_t) v_t v_t^\top\right] - 15\beta t^2 w_t \boldsymbol{I}.$$

For notational convenience let $\boldsymbol{H}_t \stackrel{\text{def}}{=} \nabla^2 f_t(x_t)$ for all $t > 0$. Then Lemma 7.3.3 shows that $\boldsymbol{H}_\alpha = \boldsymbol{H}_t + \Delta_\alpha$ where $\|\Delta_\alpha\|_2 \le 15\beta t^2 w_t$. Using $\boldsymbol{H}_\alpha^2 = \boldsymbol{H}_t^2 + \Delta_\alpha \boldsymbol{H}_t + \boldsymbol{H}_t \Delta_\alpha + \Delta_\alpha^2$, we have

$$\begin{aligned} \|\boldsymbol{H}_\alpha^2 - \boldsymbol{H}_t^2\|_2 &\le \|\Delta_\alpha \boldsymbol{H}_t\|_2 + \|\boldsymbol{H}_t \Delta_\alpha\|_2 + \|\Delta_\alpha^2\|_2 \\ &\le 2\|\Delta\|_2 \|\boldsymbol{H}_t\|_2 + \|\Delta\|_2^2 \le 40\beta t^4 w_t^2 \,. \end{aligned}$$

Let S be the subspace orthogonal to v_t. Then, Lemma 7.5.4 shows that $\boldsymbol{H}_t \succeq \frac{1}{2} t^2 w_t \boldsymbol{I}$ on S and hence $\boldsymbol{H}_t^2 \succeq \frac{1}{4} t^4 w_t^2 \boldsymbol{I}$ on S.[4] Since $\|\boldsymbol{H}_\alpha^2 - \boldsymbol{H}_t^2\|_2 \le 40\beta t^4 w_t^2$, we have that

$$\boldsymbol{H}_\alpha^2 \succeq \frac{1}{4} t^4 w_t^2 - 40\beta t^4 w_t^2 \boldsymbol{I} \text{ on } S$$

and hence

$$\boldsymbol{H}_\alpha^{-2} \preceq \left(\frac{1}{4} t^4 w_t^2 - 40\beta t^4 w_t^2\right)^{-1} \boldsymbol{I} \text{ on } S.$$

Therefore, for any $z \in S$, we have

$$\left\| \left(\nabla^2 f_\alpha(x_\alpha)\right)^{-1} z \right\|_2 = \left\| \boldsymbol{H}_\alpha^{-1} z \right\|_2 \le \frac{\|z\|_2}{\sqrt{\frac{1}{4} t^4 w_t^2 - 40\beta t^4 w_t^2}}.$$

Now, we split $y = z + \langle y, v_t \rangle v_t$ where $z \in S$. Then, we have that

$$\begin{aligned} \left\| \left(\nabla^2 f_\alpha(x_\alpha)\right)^{-1} y \right\|_2 &\le \left\| \left(\nabla^2 f_\alpha(x_\alpha)\right)^{-1} z \right\|_2 + |\langle y, v_t \rangle| \left\| \left(\nabla^2 f_\alpha(x_\alpha)\right)^{-1} v_t \right\|_2 \\ &\le \frac{1}{\sqrt{\frac{1}{4} t^4 w_t^2 - 40\beta t^4 w_t^2}} + \frac{1}{t^2 \cdot \kappa} \left\| \left(\nabla^2 f_\alpha(x_\alpha)\right)^{-1} v_t \right\|_2 . \end{aligned}$$

Note that, we also know that $\lambda_{\min}(\nabla^2 f_\alpha(x_\alpha)) \ge \mu_\alpha$ and hence $\lambda_{\max}(\nabla^2 f_\alpha(x_\alpha)^{-2}) \le \mu_\alpha^{-2}$. Therefore,

[4] By $\boldsymbol{A} \preceq \boldsymbol{B}$ on S we mean that for all $x \in S$ we have $x^\top \boldsymbol{A} x \le x^\top \boldsymbol{B} x$. The meaning of $\boldsymbol{A} \succeq \boldsymbol{B}$ on S is analagous.

we have

$$\left\|\left(\nabla^2 f_\alpha(x_\alpha)\right)^{-1} y\right\|_2 \leq \frac{1}{t^2 w_t \sqrt{\frac{1}{4} - 40\beta}} + \frac{1}{t^2}\frac{\mu_\alpha}{w_\alpha}\frac{1}{\mu_\alpha} \leq \frac{1}{t^2 w_t}\left(2 + \frac{1}{\sqrt{\frac{1}{4} - 40\beta}}\right) \leq \frac{5}{t^2 w_t}.$$

Combining these and using that $\beta \in [0, 1/600]$ yields that

$$\begin{aligned} y^\top (x_{(1+\beta)t} - x_t) &\leq \int_t^{(1+\beta)t} \frac{5}{t^2 w_t}\left(\frac{\alpha}{t}\right)^2 w_t d_\alpha \leq \frac{5}{t^4}\left(\frac{1}{3}(1+\beta)^3 t^3 - \frac{1}{3}t^3\right) \\ &\leq \frac{5}{3t}\left[(1+\beta)^3 - 1\right] \leq \frac{6\beta}{t}. \end{aligned}$$

□

7.5.5 Where is the End?

Lemma 7.3.6. $f(x_t) - f(x_*) \leq \frac{2n}{t}$ for all $t > 0$.

Proof of Lemma 7.3.6. Clearly, $\nabla f_t(x_t) = 0$ by definition of x_t. Consequently $\frac{1}{t}\nabla f_t(x_t)^\top (x_t - x_*) = 0$ and using Lemma 7.5.1 to give the formula for $\nabla f_t(x_t)$ yields

$$0 = \sum_{i\in[n]} \frac{t(x_t - a^{(i)})^\top (x_t - x_*)}{1 + g_t^{(i)}(x)} = \sum_{i\in[n]} \frac{t\|x_t - a^{(i)}\|_2^2 + t(x_t - a^{(i)})^\top (a^{(i)} - x_*)}{1 + g_t^{(i)}(x_t)}.$$

Therefore, by Cauchy Schwarz and the fact that $t\|x_t - a^{(i)}\|_2 \leq g_t^{(i)}(x_t) \leq 1 + g_t^{(i)}$

$$\sum_{i\in[n]} \frac{t\|x_t - a^{(i)}\|_2^2}{1 + g_t^{(i)}(x_t)} \leq \sum_{i\in[n]} \frac{t\|x_t - a^{(i)}\|_2 \|a^{(i)} - x_*\|_2}{1 + g_t^{(i)}(x_t)} \leq f(x_*).$$

Furthermore, since $1 + g_t^{(i)}(x_t) \leq 2 + t\|x_t - a^{(i)}\|_2$ we have

$$\sum_{i\in[n]} \frac{t\|x_t - a^{(i)}\|_2^2}{1 + g_t^{(i)}(x_t)} \geq \sum_{i\in[n]} \|x_t - a^{(i)}\|_2 - \sum_{i\in[n]} \frac{2\|x_t - a^{(i)}\|_2}{1 + g_t^{(i)}(x_t)} \geq f(x_t) - \frac{2n}{t}.$$

Combining yields the result. □

7.5.6 Simple Lemmas

Here we provide various small technical Lemmas that we will use to bound the accuracy with which we need to carry out various operations in our algorithm. Here we use some notation from Section 7.4 to simplify our bounds and make them more readily applied.

Lemma 7.5.5. For any x, we have that $\|x - x_t\|_2 \leq f(x)$.

Proof of Lemma 7.5.5. Since $\sum_{i\in[n]} \|x - a^{(i)}\|_2 = f(x)$, we have that $\|x - a^{(i)}\|_2 \leq f(x)$ for all $i \in [n]$. Since $\nabla f(x_t) = 0$ by Lemma 7.5.1 we see that x_t is a convex combination of the $a^{(i)}$ and therefore $\|x - x_t\|_2 \leq f(x)$ by convexity. □

Lemma 7.5.6. $x^{(0)} = \frac{1}{n}\sum_{i\in[n]} a^{(i)}$ is a 2-approximate geometric median, i.e. $\tilde{f}_* \leq 2 \cdot f(x_*)$.

Proof. For all $x \in \mathbb{R}^d$ we have

$$\|x^{(0)} - x\|_2 = \left\| \frac{1}{n} \sum_{i\in[n]} a^{(i)} - \frac{1}{n} \sum_{i\in[n]} x \right\|_2 \leq \frac{1}{n} \sum_{i\in[n]} \|a^{(i)} - x\|_2 \leq \frac{f(x)}{n}\,.$$

Consequently,

$$f(x^{(0)}) \leq \sum_{i\in[n]} \|x^{(0)} - a^{(i)}\|_2 \leq \sum_{i\in[n]} \left(\|x^{(0)} - x_*\|_2 + \|x_* - a^{(i)}\|_2 \right) \leq 2 \cdot f(x_*)$$

□

Lemma 7.5.7. For all $t \geq 0$, we have

$$1 \leq \frac{t^2 \cdot w_t(x)}{\mu_t(x)} \leq \bar{g}_t(x) \leq \max_{i\in[n]} g_t^{(i)}(x) \leq 1 + t \cdot f(x)\,.$$

In particular, if $\|x - x_t\|_2 \leq \frac{1}{t} + \alpha$ and $t \leq \frac{2n}{\tilde{\varepsilon}_* \cdot \tilde{f}_*}$ then $g_t^{(i)}(x) \leq 6n\varepsilon_*^{-1} + t \cdot \alpha$.

Proof of Lemma 7.5.7. The first claim $1 \leq \frac{t^2 \cdot w_t(x)}{\mu_t(x)} \leq \bar{g}_t(x)$, follows from $\mu_t(x) \geq \sum_{i\in[n]} \frac{t^2}{g_t^{(i)}(x)(1+g_t^{(i)}(x))}$ and the fact that the largest eigenvalue of $\nabla^2 f_t(x)$ is at most $t^2 \cdot w_t(x)$. The second follows from the fact that $\bar{g}_t(x)$ is a weighted harmonic mean of $g_t^{(i)}(x)$ and therefore

$$\bar{g}_t(x) \leq \max_{i\in[n]} g_t^{(i)}(x) \leq 1 + t \cdot \max_{i\in[n]} \|x - a^{(i)}\|_2 \leq 1 + t \cdot f(x)\,.$$

The final inequality comes from the fact that $\|x - a^{(i)}\|_2 \leq \|x - x_t\|_2 + \|x_t - a^{(i)}\|_2 \leq \|x - x_t\|_2 + f(x_t)$ and the fact that $f(x_t) \leq f(x_*) + \frac{2n}{t}$ by Lemma 7.3.6. □

Lemma 7.5.8. For all $x \in \mathbb{R}^d$ and $t > 0$ such that $\|x - x_t\|_2 \leq \frac{1}{t} + \alpha$ and $t \leq \frac{2n}{\tilde{\varepsilon}_* \tilde{f}_*}$ we have

$$n \left(\frac{t}{7n\tilde{\varepsilon}_*^{-1} + t \cdot \alpha} \right)^2 \|x - x_t\|_2^2 \leq f_t(x) - f_t(x_t) \leq \frac{nt^2}{2} \|x - x_t\|_2^2$$

Proof of Lemma 7.5.8. For the first inequality, note that $\nabla^2 f_t(x) \preceq \sum_{i\in[n]} \frac{t^2}{1+g_t^{(i)}(x)} \boldsymbol{I} \preceq n \cdot t^2 \boldsymbol{I}$. Consequently, if we let $n \cdot t^2 \boldsymbol{I} = \boldsymbol{H}$ in Lemma 7.9.5, we have that

$$f_t(x) - f_t(x_t) \leq \frac{1}{2} \|x - x_t\|_{\boldsymbol{H}}^2 \leq \frac{nt^2}{2} \|x - x_t\|_2^2\,.$$

For the second inequality, note that Lemma 7.5.2 and Lemma 7.5.7 yields that

$$\nabla^2 f_t(x) \succeq \sum_{i\in[n]} \frac{t^2}{(1 + g_t^{(i)}(x)) g_t^{(i)}(x)} \boldsymbol{I} \succeq n \left(\frac{t}{7n\tilde{\varepsilon}^{-1} + t \cdot \alpha} \right)^2 \boldsymbol{I}\,.$$

Consequently, if we let S denote the set of all $x \in \mathbb{R}^n$ with $\|x - x_t\|_2 \leq \frac{1}{t} + \alpha$ then we see that S is convex and contains the minimizer of f_t and therefore by applying 7.9.5 again yields the lower bound. □

7.6 Analysis of the Algorithm

Here we provide proofs, algorithms, and technical lemmas from Section 7.4.

7.6.1 Eigenvector Computation and Hessian Approximation

Below we prove that the power method can be used to compute an ε-approximate top eigenvector of a symmetric PSD matrix $\boldsymbol{A} \in \mathbb{R}^{d\times d}$ with a non-zero eigenvalue gap $g = \frac{\lambda_1(\boldsymbol{A})-\lambda_2(\boldsymbol{A})}{\lambda_1(\boldsymbol{A})}$. While it is well know that this can be by applying $\boldsymbol{A}$ to a random initial vector $O(\frac{\alpha}{g}\log(\frac{d}{\varepsilon}))$ times in the following theorem we provide a slightly less known refinement that the dimension d can be replaced with the stable rank of $\boldsymbol{A}$, $s = \sum_{i\in[d]}\frac{\lambda_i(\boldsymbol{A})}{\lambda_1(\boldsymbol{A})}$. We use this fact to avoid a dependence on d in our logarithmic factors.

Algorithm 17: `PowerMethod`$(\boldsymbol{A}, k)$

Input: symmetric PSD matrix $\boldsymbol{A} \in \mathbb{R}^{d\times d}$ and a number of iterations $k \geq 1$.
Let $x \sim \mathcal{N}(0,1)$ be drawn from a d dimensional normal distribution.
Let $y = \boldsymbol{A}^k z$
Output: $u = y/\|y\|_2$

Lemma 7.6.1 (Power Method). Let $\boldsymbol{A} \in \mathbb{R}^{d\times d}$ be a symmetric PSD matrix , let $g \stackrel{\text{def}}{=} \frac{\lambda_1(\boldsymbol{A})-\lambda_2(\boldsymbol{A})}{\lambda_1(\boldsymbol{A})}$, $s = \sum_{i\in d}\frac{\lambda_i}{\lambda_1}$, and let $\varepsilon > 0$ and $k \geq \frac{\alpha}{g}\log(\frac{n}{\varepsilon})$ for large enough constant α. In time $O(\text{nnz}(\boldsymbol{A})\cdot\log(\frac{n}{\varepsilon}))$, the algorithm `PowerMethod`$(\boldsymbol{A}, k)$ outputs a vector u such that $\langle v_1(\boldsymbol{A}), u\rangle^2 \geq 1-\varepsilon$ and $u^\top \boldsymbol{A} u \geq (1-\varepsilon)\lambda_1(\boldsymbol{A})$with high probability in n/ε.

Proof. We write $u = \sum_{i\in[d]}\alpha_i v_i(\boldsymbol{A})$. Then, we have

$$\langle v_1(\boldsymbol{A}), u\rangle^2 = \left\langle v_1(\boldsymbol{A}), \frac{\sum_{i\in[d]}\alpha_i\lambda_i(\boldsymbol{A})^k v_i(\boldsymbol{A})}{\sqrt{\sum_{i\in[d]}\alpha_i^2\lambda_i(\boldsymbol{A})^{2k}}}\right\rangle^2 = \frac{\alpha_1^2}{\alpha_1^2 + \sum_{j\neq 1}\alpha_j^2\left(\frac{\lambda_j(\boldsymbol{A})}{\lambda_1(\boldsymbol{A})}\right)^{2k}} \geq 1 - \sum_{j\neq 1}\frac{\alpha_j^2}{\alpha_1^2}\left(\frac{\lambda_j(\boldsymbol{A})}{\lambda_1(\boldsymbol{A})}\right)^{2k}$$

Re arranging terms we have

$$\begin{aligned}1 - \langle v_1(\boldsymbol{A}), u\rangle^2 &\leq \sum_{j\neq 1}\frac{\alpha_j^2}{\alpha_1^2}\left(\frac{\lambda_j(\boldsymbol{A})}{\lambda_1(\boldsymbol{A})}\right)\left(\frac{\lambda_j(\boldsymbol{A})}{\lambda_1(\boldsymbol{A})}\right)^{2k-1} \leq \sum_{j\neq 1}\frac{\alpha_j^2}{\alpha_1^2}\left(\frac{\lambda_j(\boldsymbol{A})}{\lambda_1(\boldsymbol{A})}\right)\left(\frac{\lambda_2(\boldsymbol{A})}{\lambda_1(\boldsymbol{A})}\right)^{2k-1}\\ &= \sum_{j\neq 1,}\frac{\alpha_j^2}{\alpha_1^2}\cdot\left(\frac{\lambda_j(\boldsymbol{A})}{\lambda_1(\boldsymbol{A})}\right)\cdot(1-g)^{2k-1} \leq \sum_{j\neq 1}\frac{\alpha_j^2}{\alpha_1^2}\cdot\left(\frac{\lambda_j(\boldsymbol{A})}{\lambda_1(\boldsymbol{A})}\right)\cdot\exp(-(2k-1)g)\end{aligned}$$

where we used that $\frac{\lambda_2}{\lambda_1} = 1-g \leq e^{-g}$.

Now with high probability in n we have that $\alpha_1^2 \geq \frac{1}{O(\text{poly}(n/\varepsilon))}$ by known properties of the chi-squared distribution. All that remains is to upper bound $\sum_{j\neq 1}\alpha_j^2\cdot\left(\frac{\lambda_j(\boldsymbol{A})}{\lambda_1(\boldsymbol{A})}\right)$. To bound this consider $h(\alpha) \stackrel{\text{def}}{=} \sqrt{\sum_{j\neq 1}\alpha_j^2(\lambda_j(\boldsymbol{A})/\lambda_1(\boldsymbol{A}))}$. Note that

$$\|\nabla h(\alpha)\|_2 = \left\|\frac{\sum_{j\neq 1}\vec{1}_j\cdot\alpha_j\left(\frac{\lambda_j(\boldsymbol{A})}{\lambda_1(\boldsymbol{A}}\right)}{\sqrt{\sum_{j\neq 1}\alpha_j^2\left(\frac{\lambda_j(\boldsymbol{A})}{\lambda_1(\boldsymbol{A})}\right)}}\right\|_2 = \sqrt{\frac{\sum_{j\neq 1}\alpha_j^2\left(\frac{\lambda_j(\boldsymbol{A})}{\lambda_1(\boldsymbol{A})}\right)^2}{\sum_{j\neq 1}\alpha_j^2\left(\frac{\lambda_j(\boldsymbol{A})}{\lambda_1(\boldsymbol{A})}\right)}} \leq 1\,.$$

where $\vec{1}_j$ is the indicator vector for coordinate j. Consequently h is 1-Lipschitz and by Gaussian concentration for Lipschitz functions we know there are absolute constants C and c such that

$$\Pr\left[h(\alpha) \geq \mathbf{E}h(\alpha) + \lambda\right] \leq C\exp(-c\lambda^2)\,.$$

By the concavity of square root and the expected value of the chi-squared distribution we have

$$\mathbf{E}h(\alpha) \leq \sqrt{\mathbf{E}\sum_{j\neq i}\alpha_j^2 \cdot \left(\frac{\lambda_j(\boldsymbol{A})}{\lambda_1(\boldsymbol{A})}\right)} = \sqrt{\sum_{j\neq i}\left(\frac{\lambda_j(\boldsymbol{A})}{\lambda_1(\boldsymbol{A})}\right)} \leq \sqrt{s}\,.$$

Consequently, since $s \geq 1$ we have that $\Pr[h(\alpha) \geq (1+\sqrt{t})^2 \cdot s] \leq C\exp(-c\cdot t)$ for $t \geq 1$ and that $\sum_{j\neq 1}\alpha_j \cdot \left(\frac{\lambda_j(\boldsymbol{A})}{\lambda_1(\boldsymbol{A})}\right) = O(\mathrm{poly}(n/\varepsilon)\cdot s)$ with high probability in n/ε. Since $k = \Omega(\frac{1}{g}\log(\frac{ns}{\varepsilon}))$, we have $\langle v_1(\boldsymbol{A}), u\rangle^2 \geq 1-\varepsilon$ with high probability in n. Furthermore, this implies that

$$u^\top \boldsymbol{A}u = u^\top\left(\sum_{i\in[d]}\lambda_i(\boldsymbol{A})v_i(\boldsymbol{A})v_i(\boldsymbol{A})^\top\right)u \geq \lambda_1(\boldsymbol{A})\langle v_1(\boldsymbol{A}), u\rangle^2 \geq (1-\varepsilon)\lambda_1(\boldsymbol{A})\,.$$

□

Lemma 7.4.1 (Computing Hessian Approximation). Let $x \in \mathbb{R}^d$, $t > 0$, and $\varepsilon \in (0, \frac{1}{4})$. The algorithm `ApproxMinEig`$(x,t,\varepsilon)$ outputs (λ, u) in $O(nd\log\frac{n}{\varepsilon})$ time such that if $\mu_t(x) \leq \frac{1}{4}t^2 w_t(x)$ then $\langle v_t(x), u\rangle^2 \geq 1-\varepsilon$ with high probability in n. Furthermore, if $\varepsilon \leq \frac{\mu_t(x)}{8t^2\cdot w_t(x)}$ then $\frac{1}{4}\boldsymbol{Q} \preceq \nabla^2 f_t(x) \preceq 4\boldsymbol{Q}$ with high probability in n where $\boldsymbol{Q} \stackrel{\text{def}}{=} t^2\cdot w_t(x) - \left(t^2\cdot w_t(x) - \lambda\right)uu^\top$.

Proof of Lemma 7.4.1. By Lemma 7.5.4 we know that $\frac{1}{2}\boldsymbol{Z} \preceq \nabla^2 f_t(x) \preceq \boldsymbol{Z}$ where

$$\boldsymbol{Z} = t^2\cdot w_t(x) - \left(t^2\cdot w_t(x) - \mu_t(x)\right)v_t(x)v_t(x)^\top.$$

Consequently, if $\mu_t(x) \leq \frac{1}{4}t^2 w_t(x)$, then for all unit vectors $w \perp v_t(x)$, we have that

$$w^\top\nabla^2 f_t(x)w \geq \frac{1}{2}w^\top\boldsymbol{Z}w \geq \frac{1}{2}t^2 w_t(x).$$

Since $\nabla^2 f_t(x) = t^2\cdot w_t(x) - \boldsymbol{A}$, for $\boldsymbol{A}$ in the definition of `ApproxMinEig` (Algorithm 14) this implies that $v_t(x)^\top\boldsymbol{A}v_t(x) \geq \frac{3}{4}t^2\cdot w_t(x)$ and $w^\top\boldsymbol{A}w \leq \frac{1}{2}t^2 w_t(x)$. Furthermore, we see that

$$\sum_{i\in[d]}\lambda_i(\boldsymbol{A}) = \mathrm{Tr}(\boldsymbol{A}) = \sum_{i\in[n]}\frac{t^4\|x-a^{(i)}\|_2^2}{(1+g_t^{(i)}(x))^2 g_t^{(i)}(x)} \leq t^2\cdot w_t(x)$$

Therefore, in this case, $\boldsymbol{A}$ has a constant multiplicative gap between its top two eigenvectors stable rank at most a constant (i.e. $g = O(1)$ and $s = O(1)$ in Theorem 7.6.1). Consequently, by Theorem 7.6.1 we have $\langle v_t(x), u\rangle^2 \geq 1-\varepsilon$.

For the second claim, we note that

$$t^2\cdot w_t(x) - \mu_t(x) \geq u^\top\boldsymbol{A}u \geq (1-\varepsilon)\lambda_1(\boldsymbol{A}) = (1-\varepsilon)(t^2\cdot w_t(x) - \mu_t(x))$$

Therefore, since $\lambda = u^\top\nabla^2 f_t(x)u = t^2\cdot w_t(x) - u^\top\boldsymbol{A}u$, we have

$$(1-\varepsilon)\mu_t(x) - \varepsilon\cdot t^2 w_t(x) \leq \lambda \leq \mu_t(x). \tag{7.7}$$

On the other hand, by Lemma 7.9.2, we have that

$$\sqrt{\varepsilon}\boldsymbol{I} \preceq v_t(x)v_t(x)^\top - uu^\top \preceq \sqrt{\varepsilon}\boldsymbol{I}. \tag{7.8}$$

Combining (7.7) and (7.8), we have $\frac{1}{2}\boldsymbol{Z} \preceq \boldsymbol{Q} \preceq 2\boldsymbol{Z}$ if $\varepsilon \leq \frac{\mu_t(x)}{8t^2\cdot w_t(x)}$ and $\frac{1}{4}\boldsymbol{Q} \preceq \nabla^2 f_t(x) \preceq 4\boldsymbol{Q}$ follows. On the other hand, when $\mu_t(x) > \frac{1}{4}t^2 w_t(x)$. t is the case that $\frac{1}{4}t^2\cdot w_t(x)\boldsymbol{I} \preceq \nabla^2 f_t(x) \preceq t^2\cdot w_t(x)\boldsymbol{I}$ and $\frac{1}{4}t^2\cdot w_t(x)\boldsymbol{I} \preceq \boldsymbol{Q} \preceq t^2\cdot w_t(x)\boldsymbol{I}$ again yielding $\frac{1}{4}\boldsymbol{Q} \preceq \nabla^2 f_t(x) \preceq 4\boldsymbol{Q}$. □

Lemma 7.4.2. Let $u = \texttt{ApproxMinEig}(x, t, \varepsilon_v)$ for $\varepsilon_v < \frac{1}{8}$ and x such that $\|x - x_t\|_2 \leq \frac{\varepsilon_c}{t}$ for $\varepsilon_c \leq (\frac{\varepsilon_v}{36})^{\frac{3}{2}}$. $\mu_t \leq \frac{1}{4}t^2\cdot w_t$. For all unit vectors $y \perp u$, we have $\langle y, v_t\rangle^2 \leq 8\varepsilon_v$.

Proof of Lemma 7.4.2. By Lemma 7.4.1 we know that $\langle v_t(x), u\rangle^2 \geq 1-\varepsilon_v$. Since clearly $\|x-x_t\|_2 \leq \frac{1}{20t}$, by assumption, Lemma 7.3.1 swhows

$$(1 - 6\varepsilon_c^{2/3})\nabla^2 f_t(x_t) \preceq \nabla^2 f_t(x) \preceq (1 + 6\varepsilon_c^{2/3})\nabla^2 f_t(x_t).$$

Furthermore, since $\mu_t \leq \frac{1}{4}t^2\cdot w_t$, as in Lemma 7.4.1 we know that the largest eigenvalue of $\boldsymbol{A}$ defined in $\texttt{ApproxMinEig}(x, t, \varepsilon)$ is at least $\frac{3}{4}t^2\cdot w_t$ while the second largest eigenvalue is at most $\frac{1}{2}t^2\cdot w_t$. Consequently, the eigenvalue gap, g, defined in Lemma 7.9.3 is at least $\frac{1}{3}$ and this lemma shows that $\langle v_t(x), v_t\rangle^2 \geq 1 - 36\varepsilon_c^{2/3} \geq 1 - \varepsilon_v$. Consequently, by Lemma 7.9.4, we have that $\langle u, v_t\rangle^2 \geq 1 - 4\varepsilon_v$.

To prove the final claim, we write $u = \alpha v_t + \beta w$ for an unit vector $w \perp v_t$. Since $y \perp u$, we have that $0 = \alpha\langle v_t, y\rangle + \beta\langle w, y\rangle$. Then, either $\langle v_t, y\rangle = 0$ and the result follows or $\alpha^2\langle v_t, y\rangle^2 = \beta^2\langle w, y\rangle^2$ and since $\alpha^2 + \beta^2 = 1$, we have

$$\langle v_t, y\rangle^2 \leq \frac{\beta^2\langle w, y\rangle^2}{\alpha^2} \leq \frac{1-\alpha^2}{\alpha^2} \leq 2(1-\alpha^2) \leq 8\varepsilon_v$$

where in the last line we used that $\alpha^2 \geq 1 - 4\varepsilon_v > \frac{1}{2}$ since $\varepsilon_v \leq \frac{1}{8}$. □

Lemma 7.4.3. Suppose $\mu_t \geq \frac{1}{4}t^2\cdot w_t$ and let $t' \in [t, (1 + \frac{1}{600})t]$ then $\|x_{t'} - x_t\|_2 \leq \frac{1}{100t}$.

Proof of Lemma 7.4.3. Note that $t' = (1+\beta)t$ where $\beta \in [0, \frac{1}{600}]$. Since $\frac{1}{4}t^2\cdot w_t\boldsymbol{I} \preceq \mu_t\boldsymbol{I} \preceq \nabla^2 f(x_t)$ applying Lemma 7.3.3 then yields that for all $s \in [t, t']$

$$\nabla^2 f(x_s) \succeq \nabla^2 f(x_t) - 15\beta t^2 w_t\boldsymbol{I} \succeq \left(\frac{1}{4} - 15\beta\right)t^2\cdot w_t\boldsymbol{I} \succeq \frac{t^2\cdot w_t}{6}\boldsymbol{I}.$$

Consequently, by Lemma 7.5.1, the fact that $t\|x_t - a^{(i)}\|_2 \leq g_t^{(i)}$, and Lemma 7.3.2 we have

$$\begin{aligned}
\|x_{t'} - x_t\|_2 &\leq \int_t^{t'} \left\|\frac{d}{ds}x_s\right\|_2 ds = \int_t^{t'} \left\|\left(\nabla^2 f_s(x_s)\right)^{-1}\sum_{i\in[n]}\frac{s}{(1+g_s^{(i)})g_s^{(i)}}(x_s - a^{(i)})\right\|_2 ds \\
&\leq \int_t^{t'} \frac{6}{t^2\cdot w_t}\sum_{i\in[n]}\frac{s\|x_s - a^{(i)}\|_2}{(1+g_s^{(i)})g_s^{(i)}}ds \leq \int_t^{t'}\frac{6w_s}{t^2\cdot w_t}ds \leq \int_t^{t'}\frac{6}{t^2}\cdot\left(\frac{s}{t}\right)^2 ds \\
&= \frac{6}{3t^4}[(t')^2 - (t)^3] = \frac{6}{3t}\left[(1+\beta)^3 - 1\right] \leq \frac{6\beta}{t} \leq \frac{1}{100t}
\end{aligned}$$

where in the last line we used that for $\alpha \geq 0$ we have $1 + 3\alpha \leq (1+\alpha)^3$. □

7.6.2 Line Searching

Here we prove the main results we use on centering, Lemma 7.4.4, and line searching Lemma 7.4.5. These results are our main tools for computing approximations to the central path. To prove Lemma 7.4.5 we also include here two preliminary lemmas, Lemma 7.6.2 and Lemma 7.6.3, on the structure of $g_{t,y,v}$ defined in (7.5).

Lemma 7.4.4. Given some $y \in \mathbb{R}^d$, $t > 0$ and $\varepsilon \in (0, \min\{\frac{1}{4}, \frac{\mu_t(x)}{8t^2 \cdot w_t(x)}\})$. In $O(nd\log(\frac{n}{\varepsilon}))$ time $\texttt{LocalCenter}(y, t, \varepsilon)$ computes $x^{(k)}$ such that with high probability in n/ε.

$$f_t(x^{(k)}) - \min_{\|x-y\|_2 \le \frac{1}{49t}} f_t(x) \le \varepsilon \left(f_t(y) - \min_{\|x-y\|_2 \le \frac{1}{49t}} f_t(x) \right).$$

Proof of Lemma 7.4.4. By Lemma 7.4.1 we know that $\frac{1}{4}\boldsymbol{Q} \preceq \nabla^2 f_t(y) \preceq 4\boldsymbol{Q}$ with high probability in n/ε. Furthermore for x such that $\|x - y\|_2 \le \frac{1}{50t}$ Lemma 7.3.1 shows that $\frac{1}{2}\nabla^2 f_t(x) \preceq \nabla^2 f_t(y) \preceq 2\nabla^2 f_t(x)$. Combining these we have that $\frac{1}{8}\boldsymbol{Q} \preceq \nabla^2 f_t(x) \preceq 8\boldsymbol{Q}$ for all x with $\|x-y\|_2 \le \frac{1}{50t}$. Therefore, Lemma 7.9.5 shows that

$$f_t(x^{(k)}) - \min_{\|x-y\|_2 \le \frac{1}{49t}} f_t(x) \le \left(1 - \frac{1}{64}\right)^k \left(f_t(x^{(0)}) - \min_{\|x-y\|_2 \le \frac{1}{49t}} f_t(x) \right).$$

The guarantee on $x^{(k)}$ then follows from our choice of k.

For the running time, Lemma 7.4.1 showed the cost of $\texttt{ApproxMinEig}$ is $O(nd\log(\frac{n}{\varepsilon}))$. Using Lemma 7.9.7 we see that the cost per iteration is $O(nd)$ and therefore, the total cost of the k iterations is $O(nd\log(\frac{1}{\varepsilon}))$. Combining yields the running time. □

Lemma 7.6.2. For $t > 0$, $y \in \mathbb{R}^d$, and unit vector $v \in \mathbb{R}^d$, the function $g_{t,y,v} : \mathbb{R} \to \mathbb{R}$ defined by (7.5) is convex and nt-Lipschitz.

Proof. Changing variables yields $g_{t,y,v}(\alpha) = \min_{z \in S} f_t(z + \alpha v)$ for $S \stackrel{\text{def}}{=} \{z \in \mathbb{R}^d : \|z - y\|_2 \le \frac{1}{49t}\}$. Since f_t is convex and S is a convex set, by Lemma 7.9.6 we have that $g_{t,y,v}$ is convex.

Next, by Lemma 7.5.1, triangle inequality, and the fact that $t\|x - a^{(i)}\|_2 \le g_t^{(i)}(x)$ we have

$$\|\nabla f_t(x)\|_2 = \left\| \sum_{i \in [n]} \frac{t^2(x - a^{(i)})}{1 + g_t^{(i)}(x)} \right\|_2 \le \sum_{i \in [n]} \frac{t^2\|x - a^{(i)}\|_2}{1 + g_t^{(i)}(x)} \le tn\,. \tag{7.9}$$

Consequently, $f_t(x)$ is nt-Lipschitz and for all $x, y \in \mathbb{R}^n$ we have $|f_t(x) - f_t(y)| \le nt\|x - y\|_2$. Now if we consider the set $S_\alpha \stackrel{\text{def}}{=} \{x \in \mathbb{R}^d : \|x - (y + \alpha v)\|_2 \le \frac{1}{49t}\}$ then we see that for all $\alpha, \beta \in \mathbb{R}$ there is a bijection from S_α to S_β were every point in the set moves by at most $\|(\alpha - \beta)v\|_2 \le |\alpha - \beta|$. Consequently, since $g_{t,y,v}(\alpha)$ simply minimizes f_t over S_α we have that $|g_{t,y,v}(\alpha) - g_{t,y,v}(\beta)| \le nt|\alpha - \beta|$ for all $\alpha, \beta \in \mathbb{R}$ we have that $g_{t,yv}$ is nt-Lipschitz as desired. □

Lemma 7.6.3. Let $\frac{1}{400f(x_*)} \le t \le t' \le (1 + \frac{1}{600})t \le \frac{2n}{\tilde{\varepsilon} \cdot \tilde{f}_*}$ and let $u = \texttt{ApproxMinEig}(y, t, \varepsilon_v)$ for $\varepsilon_v \le \frac{1}{8}(\frac{\tilde{\varepsilon}_*}{7n})^2$ and $y \in \mathbb{R}^d$ such that $\|y - x_t\|_2 \le \frac{1}{t}(\frac{\varepsilon_v}{36})^{\frac{3}{2}}$. The function $g_{t,t',y,v} : \mathbb{R} \to \mathbb{R}$ defined in (7.5) satisfies $g_{t,y,v}(\alpha_*) = \min_\alpha g_{t,y,v}(\alpha) = f_{t'}(x_{t'})$ for some $\alpha_* \in [-6f(x_*), 6f(x_*)]$.

Proof of Lemma 7.4.5. Let $z \in \mathbb{R}^d$ be an arbitrary unit vector. By Lemma 7.3.5 we know if $|\langle z, v_t\rangle| \le \min_{\delta \in [t,(1+\beta)t]} \frac{\mu_\delta}{t^2 \cdot w_\delta}$ for $\beta \in [0, \frac{1}{600}]$ then $z^\top(x_{(1+\beta)t} - x_t) \le \frac{6\beta}{t}$. Now by Lemma 7.5.7 and our bound on β we know that $\frac{\delta^2 \cdot w_\delta}{\mu_\delta} \le 7n\tilde{\varepsilon}_*^{-1}$ and consequently $\frac{\tilde{\varepsilon}_*}{6n} \le \min_{\delta \in [t,(1+\beta)t]} \frac{w_\delta}{t^2 \cdot \mu_\delta}$. Consequently, if $\mu_t \le$

$\frac{1}{4}t^2 \cdot w_t$ then by Lemma 7.4.2 and our choice of ε_v we have that if $z \perp u$ then $|\langle z, v_t\rangle|^2 \leq (\frac{\tilde{\varepsilon}_*}{6n})^2$ and $z^\top(x_{t'} - x_t) \leq \frac{6}{600t} = \frac{1}{100t}$. Otherwise, $\mu_t \geq \frac{1}{4}t^2 \cdot w_t$ and by Lemma 7.4.3 we have $\|x_{t'} - x_t\|_2 \leq \frac{1}{100t}$. In either case, since $\|y - x_t\|_2 \leq \frac{1}{100t}$, we can reach $x_{t'}$ from y by first moving an Euclidean distance of $\frac{1}{100t}$ to go from x to x_t, then adding some multiple of v, then moving an Euclidean distance of $\frac{1}{100t}$ in a direction perpendicular to v. Since the total movement perpendicular to v is $\frac{1}{100t} + \frac{1}{100t} \leq \frac{1}{49t'}$ we have that $\min_\alpha g_{t',y,v}(\alpha) = f_{t'}(x_{t'})$ as desired.

All that remains is to show that there is a minimizer of $g_{t,y,v}$ in the range $[-6f(x_*), 6f(x_*)]$. However, by Lemma 7.3.6 and Lemma 7.5.5 we know that

$$\|y - x_{t'}\|_2 \leq \|y - x_t\|_2 + \|x_t - x_*\|_2 + \|x_* - x_{t'}\|_2 \leq \frac{1}{100t} + f(x_*) + f(x_*) \leq 6f(x_*)\,.$$

Consequently, $\alpha_* \in [-6f(x_*), 6f(x_*)]$ as desired.

Lemma 7.4.5. Let $\frac{1}{400f(x_*)} \leq t \leq t' \leq (1 + \frac{1}{600})t \leq \frac{2n}{\tilde{\varepsilon}\cdot\tilde{f}_*}$ and let $u = \texttt{ApproxMinEig}(y, t, \varepsilon_v)$ for $\varepsilon_v \leq \frac{1}{8}(\frac{\tilde{\varepsilon}_*}{7n})^2$ and $y \in \mathbb{R}^d$ such that $\|y - x_t\|_2 \leq \frac{1}{t}(\frac{\varepsilon_v}{36})^{\frac{3}{2}}$. In $O(nd\log^2(\frac{n}{\tilde{\varepsilon}\cdot\varepsilon\cdot\varepsilon_v}))$ time, $\texttt{LineSearch}(y, t, t', u, \varepsilon)$ outputs x' such that $\|x' - x_{t'}\|_2 \leq \frac{\varepsilon}{t'}$ with high probability in n.

By (7.9) we know that $f_{t'}$ is nt' Lipschitz and therefore

$$f_t(y) - \min_{\|x-y\|_2 \leq \frac{1}{49t'}} f_{t'}(x) \leq \frac{nt'}{49t'} = \frac{n}{49}\,.$$

Furthermore, for $\alpha \in [-6\tilde{f}_*, 6\tilde{f}_*]$ we know that by Lemma 7.5.5

$$\|y + \alpha u - x_{t'}\|_2 \leq \|y - x_t\|_2 + |\alpha| + \|x_t - x_*\|_2 + \|x_{t'} - x_*\|_2 \leq \frac{1}{t'} + 14f(x_*)$$

consequently by Lemma 7.5.7 we have

$$\frac{(t')^2 \cdot w_{t'}(y + \alpha u)}{\mu_{t'}(y + \alpha u)} \leq 6n\tilde{\varepsilon}^{-1} + t' \cdot 14f(x_*) \leq 34n\tilde{\varepsilon}^{-1}\,.$$

Since $\varepsilon_O \leq \frac{1}{8} \cdot \frac{\tilde{\varepsilon}_*}{34n}$ invoking Lemma 7.4.4 yields that $|q(\alpha) - g_{t',y,u}(\alpha) \leq \frac{n\varepsilon_O}{49}$ with high probability in $n/\tilde{\varepsilon}_*$ by and each call to `LocalCenter` takes $O(nd\log\frac{n}{\varepsilon_O})$ time. Furthermore, by Lemma 7.6.2 we have that $g_{t',y,u}$ is a nt'-Lipschitz convex function and by Lemma 7.6.3 we have that the minimizer has value $f_{t'}(x_{t'})$ and is achieved in the range $[-12\tilde{f}_*, 12\tilde{f}_*]$. Consequently, combining all these facts and invoking Lemma 7.9.3, i.e. our result on on one dimensional function minimization, we have $f_{t'}(x') - f_{t'}(x_{t'}) \leq \frac{\varepsilon_O}{49t'}$ using only $O(\log\frac{nt'f(x_*)}{\varepsilon_O})$ calls to `LocalCenter`.

Finally, by Lemma 7.5.8 and Lemma 7.3.6 we have

$$\|x - x_{t'}\| \leq \frac{1}{\sqrt{n}} \cdot \left(\frac{7n\tilde{\varepsilon}^{-1} + t'\alpha}{t'}\right) \cdot \sqrt{\frac{n\varepsilon_O}{49}} \leq \frac{34n}{7 \cdot \tilde{\varepsilon}_* \cdot t'}\sqrt{\varepsilon_O} \leq \frac{6n}{\tilde{\varepsilon}_* \cdot t'}\sqrt{\varepsilon_O}\,.$$

Since $\varepsilon_O \leq \frac{\varepsilon^2\cdot\tilde{\varepsilon}_*^2}{36n^2}$ we have that $\|x - x_{t'}\|_2 \leq \frac{\varepsilon}{t'}$ as desired. □

Lemma 7.4.6. Let $\frac{1}{400f(x_*)} \leq t \leq t' \leq (1 + \frac{1}{600})t \leq \frac{2n}{\tilde{\varepsilon}_*\cdot\tilde{f}_*}$ and let $x \in \mathbb{R}^d$ satisfy $\|x - x_t\|_2 \leq \frac{1}{100t}$. Then, in $O(nd\log^2(\frac{n}{\varepsilon\cdot\tilde{\varepsilon}_*}))$ time, $\texttt{LineSearch}(x, t, t, u, \varepsilon)$ output y such that $\|y - x_t\|_2 \leq \frac{\varepsilon}{t}$ for any vector $u \in \mathbb{R}^d$.

Proof of Lemma 7.4.6. The proof is strictly easier than the proof of Lemma 7.4.5 as $\|x - \alpha^* u - x_t\|_2 \leq \frac{1}{100t}$ is satisfied automatically for $\alpha^* = 0$. Note that this lemma assume less for the initial point. □

7.6.3 Putting It All Together

Theorem 7.4.1. *In $O(nd\log^3(\frac{n}{\varepsilon}))$ time, Algorithm 13 outputs an $(1+\varepsilon)$-approximate geometric median with constant probability.*

Proof of Theorem 7.4.1. By Lemma 7.5.6 we know that $x^{(0)}$ is a 2-approximate geometric median and therefore $f(x^{(0)}) = \tilde{f}_* \le 2 \cdot f(x_*)$. Furthermore, since $\|x^{(0)} - x_{t_1}\|_2 \le f(x^{(0)})$ by Lemma 7.5.5 and $t_1 = \frac{1}{400\tilde{f}_*}$ we have $\|x^{(0)} - x_{t_1}\|_2 \le \frac{1}{400t_1}$. Hence, by Lemma 7.4.6, we have $\|x^{(1)} - x_{t_1}\|_2 \le \frac{\varepsilon_c}{t_1}$ with high probability in n/ε. Consequently, by Lemma 7.4.5 we have that$\|x^{(k)} - x_{t_i}\|_2 \le \frac{\varepsilon_c}{t_i}$ for all i with high probability in n/ε.

Now, Lemma 7.3.6 shows that

$$f(x_{t_k}) - f(x^*) \le \frac{2n}{t_k} \le \frac{2n}{\tilde{t}_*}\left(1 + \frac{1}{600}\right) \le \tilde{\varepsilon}_* \cdot \tilde{f}_*\left(1 + \frac{1}{600}\right) \le \frac{2}{3}\left(1 + \frac{1}{600}\right)\varepsilon \cdot f(x_*)\,.$$

Since $\|x^{(k)} - x_{t_k}\|_2 \le \frac{\varepsilon_c}{t_k} \le 400 \cdot \tilde{f}_* \cdot \varepsilon_c$ we have that $f(x_k) \le f(x_{t_k}) + 400n \cdot \tilde{f}_* \cdot \varepsilon_c$ by triangle inequality. Combining these facts and using that ε_c is sufficiently small yields that $f(x^{(k)}) \le (1+\varepsilon)f(x_*)$ as desired.

To bound the running time, Lemma 7.4.1 shows `ApproxMinEvec` takes $O(nd\log(\frac{n}{\varepsilon}))$ per iteration and Lemma 7.4.5 shows `LineSearch` takes $O\left(nd\log^2\left(\frac{n}{\varepsilon}\right)\right)$ time per iteration, using that ε_v and ε_c are $O(\Omega(\varepsilon/n))$. Since for $l = \Omega(\log\frac{n}{\varepsilon})$ we have that $t_l > \tilde{t}_*$ we have that $k = O(\log\frac{n}{\varepsilon})$. $t_{i+1} \le \frac{1}{400}$. Since there are $O(\log(\frac{n}{\varepsilon}))$ iterations taking time $O\left(nd\log^2\left(\frac{n}{\varepsilon}\right)\right)$the running time follows. □

7.7 Pseudo Polynomial Time Algorithm

Here we provide a self-contained result on computing a $1+\varepsilon$ approximate geometric median in $O(d\varepsilon^{-2})$ time. Note that it is impossible to achieve such approximation for the mean, $\min_{x\in\mathbb{R}^d}\sum_{i\in[n]}\|x-a^{(i)}\|_2^2$, because the mean can be changed arbitrarily by changing only 1 point. However, [172] showed that the geometric median is far more stable. In Section 7.7.1, we show how this stability property allows us to get an constant approximate in $O(d)$ time. In Section 7.7.2, we show how to use stochastic subgradient descent to then improve the accuracy.

7.7.1 A Constant Approximation of Geometric Median

We first prove that the geometric median is stable even if we are allowed to modify up to half of the points. The following lemma is a strengthening of the robustness result in [172].

Lemma 7.7.1. Let x_* be a geometric median of $\{a^{(i)}\}_{i\in[n]}$ and let $S \subseteq [n]$ with $\|S\| < \frac{n}{2}$. For all x

$$\|x_* - x\|_2 \le \left(\frac{2n - 2|S|}{n - 2|S|}\right)\max_{i\notin S}\|a^{(i)} - x\|_2\,.$$

Proof. For notational convenience let $r = \|x_* - x\|_2$ and let $M = \max_{i\notin S}\|a^{(i)} - x\|_2$.

For all $i \notin S$, we have that $\|x - a^{(i)}\|_2 \le M$, hence, we have

$$\begin{aligned}\|x_* - a^{(i)}\|_2 &\ge r - \|x - a^{(i)}\|_2 \\ &\ge r - 2M + \|x - a^{(i)}\|_2\,.\end{aligned}$$

Furthermore, by triangle inequality for all $i \in S$, we have

$$\|x_* - a^{(i)}\|_2 \ge \|x - a^{(i)}\|_2 - r\,.$$

Hence, we have that

$$\sum_{i\in[n]} \|x_* - a^{(i)}\|_2 \geq \sum_{i\in[n]} \|x - a^{(i)}\|_2 + (n-|S|)(r-2M) - |S|r\,.$$

Since x^* is a minimizer of $\sum_{i\in[n]} \|x_* - a^{(i)}\|_2$, we have that

$$(n-|S|)(r-2M) - |S|r \leq 0.$$

Hence, we have

$$\|x^* - x\|_2 = r \leq \frac{2n-2|S|}{n-2|S|}M.$$

□

Now, we use Lemma 7.7.1 to show that the algorithm `CrudeApproximate` outputs a constant approximation of the geometric median with high probability.

Algorithm 18: $\texttt{CrudeApproximate}_K$

Input: $a^{(1)}, a^{(2)}, \cdots, a^{(n)} \in \mathbb{R}^d$.
Sample two independent random subset of $[n]$ of size K. Call them S_1 and S_2.
Let $i^* \in \arg\min_{i\in S_2} \alpha_i$ where α_i is the 65 percentile of the numbers $\{\|a^{(i)} - a^{(j)}\|_2\}_{j\in S_1}$.
Output: Output $a^{(i^*)}$ and α_{i^*}.

Lemma 7.7.2. Let x_* be a geometric median of $\{a^{(i)}\}_{i\in[n]}$ and $(\widetilde{x}, \lambda)$ be the output of $\texttt{CrudeApproximate}_K$. We define $d_T^k(x)$ be the k-percentile of $\left\{\|x - a^{(i)}\|\right\}_{i\in T}$. Then, we have that $\|x_* - \widetilde{x}\|_2 \leq 6d_{[n]}^{60}(\widetilde{x})$. Furthermore, with probability $1 - e^{-\Theta(K)}$, we have

$$d_{[n]}^{60}(\widetilde{x}) \leq \lambda = d_{S_1}^{60}(\widetilde{x}) \leq 2d_{[n]}^{70}(x_*).$$

Proof. Lemma 7.7.1 shows that for all x and $T \subseteq [n]$ with $|T| \leq \frac{n}{2}$

$$\|x_* - x\|_2 \leq \left(\frac{2n-2|T|}{n-2|T|}\right) \max_{i\notin T} \|a^{(i)} - x\|_2\,.$$

Picking T to be the indices of largest 40% of $\|a^{(i)} - \widetilde{x}\|_2$, we have

$$\|x_* - \widetilde{x}\|_2 \leq \left(\frac{2n-0.8n}{n-0.8n}\right) d_{[n]}^{60}(\widetilde{x}) = 6d_{[n]}^{60}(\widetilde{x}). \tag{7.10}$$

For any point x, we have that $d_{[n]}^{60}(x) \leq d_{S_1}^{65}(x)$ with probability $1 - e^{-\Theta(K)}$ because S_1 is a random subset of $[n]$ with size K. Taking union bound over elements on S_2, with probability $1 - Ke^{-\Theta(K)} = 1 - e^{-\Theta(K)}$, for all points $x \in S_2$

$$d_{[n]}^{60}(x) \leq d_{S_1}^{65}(x). \tag{7.11}$$

yielding that $d_{[n]}^{60}(\widetilde{x}) \leq \lambda$.

Next, for any $i \in S_2$, we have

$$\|a^{(i)} - a^{(j)}\|_2 \leq \|a^{(i)} - x_*\|_2 + \|x_* - a^{(j)}\|_2.$$

and hence

$$d_{[n]}^{70}(a^{(i)}) \leq \|a^{(i)} - x_*\|_2 + d_{[n]}^{70}(x_*).$$

Again, since S_1 is a random subset of $[n]$ with size K, we have that $d^{65}_{S_1}(a^{(i)}) \le d^{70}_{[n]}(a^{(i)})$ with probability $1 - Ke^{-\Theta(K)} = 1 - e^{-\Theta(K)}$. Therefore,

$$d^{65}_{S_1}(a^{(i)}) \le \|a^{(i)} - x_*\|_2 + d^{70}_{[n]}(x_*).$$

Since S_2 is an independent random subset, with probability $1 - e^{-\Theta(K)}$, there is $i \in S_2$ such that $\|a^{(i)} - x_*\|_2 \le d^{70}_{[n]}(x_*)$. In this case, we have

$$d^{65}_{S_1}(a^{(i)}) \le 2d^{70}_{[n]}(x_*).$$

Since i^* minimize $d^{65}_{S_1}(a^{(i)})$ over all $i \in S_2$, we have that

$$\lambda \stackrel{\text{def}}{=} d^{65}_{S_1}(\widetilde{x}) \stackrel{\text{def}}{=} d^{65}_{S_1}(a^{(i^*)}) \le d^{65}_{S_1}(a^{(i)}) \le 2d^{70}_{[n]}(x_*)\,.$$

□

■ 7.7.2 A $1+\varepsilon$ Approximation of Geometric Median

Here we show how to improve the constant approximation in the previous section to a $1+\varepsilon$ approximation. Our algorithm is essentially stochastic subgradient where we use the information from the previous section to bound the domain in which we need to search for a geometric median.

Algorithm 19: `ApproximateMedian`(ε)

Input: $a^{(1)}, a^{(2)}, \cdots, a^{(n)} \in \mathbb{R}^d$.

Let $T = (120/\varepsilon)^2$ and let $\eta = \frac{6\lambda}{n}\sqrt{\frac{2}{T}}$.

Let $(x^{(1)}, \lambda) = \texttt{CrudeApproximate}_{\sqrt{T}}(a^{(1)}, a^{(2)}, \cdots, a^{(n)})$.

for $k \leftarrow 1, 2, \cdots, T$ **do**

Sample i_k from $[n]$ and let

$$g^{(k)} = \begin{cases} n(x^{(k)} - a^{(i_k)})/\|x^{(k)} - a^{(i_k)}\|_2 & \text{if } x^{(i)} \neq a^{(i_k)} \\ 0 & \text{otherwise} \end{cases}$$

Let $x^{(k+1)} = \arg\min_{\|x - x^{(1)}\|_2 \le 6\lambda} \eta \left\langle g^{(k)}, x - x^{(k)} \right\rangle + \frac{1}{2}\|x - x^{(k)}\|_2^2$.

end

Output: Output $\frac{1}{T}\sum_{i=1}^{T} x^{(k)}$.

Theorem 7.7.1. *Let x be the output of* `ApproximateMedian`(ε). *With probability $1 - e^{-\Theta(1/\varepsilon)}$, we have*

$$\mathbb{E} f(x) \le (1+\varepsilon) \min_{x \in \mathbb{R}^d} f(x).$$

Furthermore, the algorithm takes $O(d/\varepsilon^2)$ time.

Proof. After computing $x^{(1)}$ and λ the remainder of our algorithm is the stocastic subgradient descent method applied to $f(x)$. It is routine to check that $\mathbf{E}_{i(k)} g^{(k)}$ is a subgradient of f at $x^{(k)}$. Furthermore, since the diameter of the domain, $\left\{x : \|x - x^{(1)}\|_2 \le 6\lambda\right\}$, is clearly λ and the norm of sampled gradient, $g^{(k)}$, is at most n, we have that

$$\mathbb{E} f\left(\frac{1}{T}\sum_{i=1}^{T} x^{(k)}\right) - \min_{\|x - x^{(1)}\|_2 \le 6\lambda} f(x) \le 6n\lambda\sqrt{\frac{2}{T}}$$

(see [41, Thm 6.1]). Lemma 7.7.2 shows that $\|x^* - x^{(1)}\|_2 \leq 6\lambda$ and $\lambda \leq 2d^{70}_{[n]}(x^*)$ with probability $1 - \sqrt{T}e^{-\Theta(\sqrt{T})}$. In this case, we have

$$\mathbb{E}f\left(\frac{1}{T}\sum_{i=1}^{T} x^{(k)}\right) - f(x^*) \quad \leq \quad \frac{12\sqrt{2}nd^{70}_{[n]}(x_*)}{\sqrt{T}}.$$

Since $d^{70}_{[n]}(x^*) \leq \frac{1}{0.3n}f(x^*)$, we have

$$\mathbb{E}f\left(\frac{1}{T}\sum_{i=1}^{T} x^{(k)}\right) \quad \leq \quad \left(1 + \frac{60}{\sqrt{T}}\right) f(x_*) \leq \left(1 + \frac{\varepsilon}{2}\right) f(x_*)\,.$$

□

■ 7.8 Derivation of Penalty Function

Here we derive our penalized objective function. Consider the following optimization problem:

$$\min_{x\in\mathbb{R}^d, \alpha\geq 0\in\mathbb{R}^n} f_t(x,\alpha) \quad \text{where} \quad t\cdot 1^T\alpha + \sum_{i\in[n]} -\ln\left(\alpha_i^2 - \|x - a^{(i)}\|_2^2\right) \quad .$$

Since $p_i(\alpha, x) \stackrel{\text{def}}{=} -\ln\left(\alpha_i^2 - \|x - a^{(i)}\|_2^2\right)$ is a barrier function for the set $\alpha_i^2 \geq \|x - a^{(i)}\|_2^2$, i.e. as $\alpha_i \to \|x - a^{(i)}\|_2$we have $p_i(\alpha, x) \to \infty$, we see that as we minimize $f_t(x, \alpha)$ for increasing values of t the x values converge to a solution to the geometric median problem. Our penalized objective function, $f_t(x)$, is obtain simply by minizing the α_i in the above formula and dropping terms that do not affect the minimizing x. In the remainder of this section we show this formally.

Fix some $x \in \mathbb{R}^d$ and $t > 0$. Note that for all $j \in [n]$ we have

$$\frac{\partial}{\partial\alpha_j} f_t(x,\alpha) = t - \left(\frac{1}{\alpha_j^2 - \|x - a^{(i)}\|_2^2}\right) 2\alpha_j\,.$$

Since $f(x, \alpha)$ is convex in α, the minimum α_j^* must satisfy

$$t\left(\left(\alpha_j^*\right)^2 - \|x - a^{(i)}\|_2^2\right) - 2\alpha_j^* = 0 \quad . \tag{7.12}$$

Solving for such α_j^* under the restriction $\alpha_j^* \geq 0$ we obtain

$$\alpha_j^* = \frac{2 + \sqrt{4 + 4t^2\|x - a^{(i)}\|_2^2}}{2t} = \frac{1}{t}\left[1 + \sqrt{1 + t^2\|x - a^{(i)}\|_2^2}\right] \quad . \tag{7.13}$$

Using (7.12) and (7.13) we have that

$$\min_{\alpha\geq 0\in\mathbb{R}^n} f_t(x,\alpha) = \sum_{i\in[n]} \left[1 + \sqrt{1 + t^2\|x - a^{(i)}\|_2^2} - \ln\left[\frac{2}{t^2}\left(1 + \sqrt{1 + t^2\|x - a^{(i)}\|_2^2}\right)\right]\right] \quad .$$

If we drop the terms that do not affect the minimizing x we obtain our penalty function f_t :

$$f_t(x) = \sum_{i\in[n]} \left[\sqrt{1 + t^2\|x - a^{(i)}\|_2^2} - \ln\left(1 + \sqrt{1 + t^2\|x - a^{(i)}\|_2^2}\right)\right] \quad .$$

7.9 Technical Lemmas

Here we provide various technical lemmas we use through this chapter.

7.9.1 Linear Algebra

First we provide the following lemma that shows that any matrix obtained as a non-negative linear combination of the identity minus a rank 1 matrix less than the identity results in a matrix that is well approximated spectrally by the identity minus a rank 1 matrix. We use this lemma to characterize the Hessian of our penalized objective function and thereby imply that it is possible to apply the inverse of the Hessian to a vector with high precision.

Lemma 7.9.1. Let $\boldsymbol{A} = \sum_i \left(\alpha_i \boldsymbol{I} - \beta_i a_i a_i^\top\right) \in \mathbb{R}^{d\times d}$ where the a_i are unit vectors and $0 \le \beta_i \le \alpha_i$ for all i. Let v denote a unit ector that is the maximum eigenvector of $\sum_i \beta_i a_i a_i^\top$ and let λ denote the corresponding eigenvalue. Then,

$$\frac{1}{2}\left(\sum_i \alpha_i \boldsymbol{I} - \lambda v v^\top\right) \preceq \boldsymbol{A} \preceq \sum_i \alpha_i \boldsymbol{I} - \lambda v v^\top .$$

Proof. Let $\alpha \stackrel{\text{def}}{=} \sum_i \alpha_i$. Since clearly $v^\top \boldsymbol{A} v = v^\top \left(\sum_i \alpha_i \boldsymbol{I} - \lambda v v^\top\right) v$ it suffices to show that for $w \perp v$ it is the case that $\frac{1}{2}\alpha\|w\|_2^2 \preceq w^\top \boldsymbol{A} w \preceq \alpha\|w\|_2^2$ or equivalently, that $\lambda_i(\boldsymbol{A}) \in [\frac{1}{2}\alpha, \alpha]$ for $i \neq d$. However we know that $\sum_{i\in[d]} \lambda_i(\boldsymbol{A}) = \mathrm{Tr}(\boldsymbol{A}) = d\alpha - \sum_i \beta_i \ge (d-1)\alpha$ and $\lambda_i(\boldsymbol{A}) \le \alpha$ for all $i \in [d]$. Consequently, since $\lambda_d(\boldsymbol{A}) \in [0, \lambda_{d-1}]$ we have

$$2 \cdot \lambda_{d-1}(\boldsymbol{A}) \ge (d-1)\alpha - \sum_{i=1}^{d-2} \lambda_i(\boldsymbol{A}) \ge (d-1)\alpha - (d-2)\alpha = \alpha .$$

Consequently, $\lambda_{d-1}(\boldsymbol{A}) \in [\frac{\alpha}{2}, \alpha]$ and the result holds by the monotonicity of λ_i. □

Next we bound the spectral difference between the outer product of two unit vectors by their inner product. We use this lemma to bound the amount of precision required in our eigenvector computations.

Lemma 7.9.2. For unit vectors u_1 and u_2 we have

$$\|u_1 u_1^\top - u_2 u_2^\top\|_2^2 = 1 - (u_1^\top u_2)^2 \tag{7.14}$$

Consequently if $\left(u_1^\top u_2\right)^2 \ge 1 - \varepsilon$ for $\varepsilon \le 1$ we have that

$$-\sqrt{\varepsilon}\boldsymbol{I} \preceq u_1 u_1^\top - u_2 u_2^\top \preceq \sqrt{\varepsilon}\boldsymbol{I}$$

Proof. Note that $u_1 u_1^\top - u_2 u_2^\top$ is a symmetric matrix and all eigenvectors are either orthogonal to both u_1 and u_2 (with eigenvalue 0) or are of the form $v = \alpha u_1 + \beta u_2$ where α and β are real numbers that are not both 0. Thus, if v is an eigenvector of eigenvalue λ it must be that

$$\begin{aligned}\lambda\left(\alpha u_1 + \beta u_2\right) &= \left(u_1 u_1^\top - u_2 u_2^\top\right)\left(\alpha u_1 + \beta u_2\right) \\ &= \left(\alpha + \beta(u_1^\top u_2)\right)u_1 - \left(\alpha(u_1^\top u_2) + \beta\right)u_2\end{aligned}$$

or equivalently

$$\begin{pmatrix} (1-\lambda) & u_1^\top u_2 \\ -(u_1^\top u_2) & -(1+\lambda) \end{pmatrix}\begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} .$$

By computing the determinant we see this has a solution only when

$$-(1-\lambda^2) + (u_1^\top u_2)^2 = 0$$

Solving for λ then yields (7.14) and completes the proof. □

Next we show how the top eigenvectors of two spectrally similar matrices are related. We use this to bound the amount of spectral approximation we need to obtain accuracate eigenvector approximations.

Lemma 7.9.3. Let $\boldsymbol{A}$ and $\boldsymbol{B}$ be symmetric PSD matrices such that $(1-\varepsilon)\boldsymbol{A} \preceq \boldsymbol{B} \preceq (1+\varepsilon)\boldsymbol{A}$. Then if $g \stackrel{\text{def}}{=} \frac{\lambda_1(\boldsymbol{A})-\lambda_2(\boldsymbol{A})}{\lambda_1(\boldsymbol{A})}$ satisfies $g > 0$ we have $[v_1(\boldsymbol{A})^\top v_1(\boldsymbol{B})]^2 \geq 1 - 2(\varepsilon/g)$.

Proof. Without loss of generality $v_1(\boldsymbol{B}) = \alpha v_1(\boldsymbol{A}) + \beta v$ for some unit vector $v \perp v_1(\boldsymbol{A})$ and $\alpha, \beta \in \mathbb{R}$ such that $\alpha^2 + \beta^2 = 1$. Now we know that

$$v_1(\boldsymbol{B})^\top \boldsymbol{B} v_1(\boldsymbol{B}) \leq (1+\varepsilon) v_1(\boldsymbol{B})^\top \boldsymbol{A} v_1(\boldsymbol{B}) \leq (1+\varepsilon)\left[\alpha^2 \lambda_1(\boldsymbol{A}) + \beta^2 \lambda_2(\boldsymbol{A})\right]$$

Furthermore, by the optimality of $v_1(\boldsymbol{B})$ we have that

$$v_1(\boldsymbol{B})^\top \boldsymbol{B} v_1(\boldsymbol{B}) \geq (1-\varepsilon) v_1(\boldsymbol{A})^\top \boldsymbol{A} v_1(\boldsymbol{A}) \geq (1-\varepsilon)\lambda_1(\boldsymbol{A}).$$

Now since $\beta^2 = 1 - \alpha^2$ combining these inequalities yields

$$(1-\varepsilon)\lambda_1(\boldsymbol{A}) \leq (1+\varepsilon)\alpha^2\left(\lambda_1(\boldsymbol{A}) - \lambda_2(\boldsymbol{A})\right) + (1+\varepsilon)\lambda_2(\boldsymbol{A}).$$

Rearranging terms, using the definition of g, and that $g \in (0,1]$ and $\varepsilon \geq 0$yields

$$\alpha^2 \geq \frac{\lambda_1(\boldsymbol{A}) - \lambda_2(\boldsymbol{A}) - \varepsilon(\lambda_1(\boldsymbol{A}) + \lambda_2(\boldsymbol{A}))}{(1+\varepsilon)(\lambda_1(\boldsymbol{A}) - \lambda_2(\boldsymbol{A}))} = 1 - \frac{2\varepsilon\lambda_1(\boldsymbol{A})}{(1+\varepsilon)(\lambda_1(\boldsymbol{A}) - \lambda_2(\boldsymbol{A}))} \geq 1 - 2(\varepsilon/g).$$

□

Here we prove a an approximate transitivity lemma for inner products of vectors. We use this to bound the accuracy need for certain eigenvector computations.

Lemma 7.9.4. Suppose that we have vectors $v_1, v_2, v_3 \in \mathbb{R}^n$ such that $\langle v_1, v_2\rangle^2 \geq 1-\varepsilon$ and $\langle v_2, v_3\rangle^2 \geq 1-\varepsilon$ for $0 < \varepsilon \leq \frac{1}{2}$ then $\langle v_1, v_3\rangle^2 \geq 1 - 4\varepsilon$.

Proof. Without loss of generality, we can write $v_1 = \alpha_1 v_2 + \beta_1 w_1$ for $\alpha_1^2 + \beta_1^2 = 1$ and unit vector $w_1 \perp v_2$. Similarly we can write $v_3 = \alpha_3 v_2 + \beta_3 w_3$ for $\alpha_3^2 + \beta_3^2 = 1$ and unit vector $w_3 \perp v_2$. Now, by the inner products we know that $\alpha_1^2 \geq 1-\varepsilon$ and $\alpha_3^2 \geq 1-\varepsilon$ and therefore $|\beta_1| \leq \sqrt{\varepsilon}$ and $|\beta_3| \leq \sqrt{\varepsilon}$. Consequently, since $\varepsilon \leq \frac{1}{2}$, $|\beta_1\beta_3| \leq \varepsilon \leq 1-\varepsilon \leq |\alpha_1\alpha_3|$, and we have

$$\begin{aligned}\langle v_1, v_3\rangle^2 &\geq \langle \alpha_1 v_2 + \beta_1 w_1, \alpha_3 v_2 + \beta_3 w_3\rangle^2 \geq \left(|\alpha_1\alpha_3| - |\beta_1\beta_3|\right)^2 \\ &\geq (1-\varepsilon-\varepsilon)^2 = (1-2\varepsilon)^2 \geq 1 - 4\varepsilon.\end{aligned}$$

□

7.9.2 Convex Optimization

First we provide a single general lemma about about first order methods for convex optimization. We use this lemma for multiple purposes including bounding errors and quickly compute approximations to the central path.

Lemma 7.9.5. [206]Let $f : \mathbb{R}^n \to \mathbb{R}$ be a twice differentiable function, let $B \subseteq \mathbb{R}$ be a convex set, and let x_* be a point that achieves the minimum value of f restricted to B. Further suppose that for a symmetric positive definite matrix $\boldsymbol{H} \in \mathbb{R}^{n\times n}$ we have that $\mu\boldsymbol{H} \preceq \nabla^2 f(y) \preceq L\boldsymbol{H}$ for all $y \in B$.Then for all $x \in B$ we have

$$\frac{1}{2L}\|\nabla f(x)\|^2_{\boldsymbol{H}^{-1}} \leq f(x) - f(x_*) \leq \frac{L}{2}\|x - x_*\|^2_{\boldsymbol{H}}$$

and

$$\frac{\mu}{2}\|x - x_*\|^2_{\boldsymbol{H}} \leq f(x) - f(x_*) \leq \frac{1}{2\mu}\|\nabla f(x)\|^2_{\boldsymbol{H}^{-1}}\,.$$

Furthermore, if

$$x_1 = \arg\min_{x\in B}\left[f(x_0) + \langle\nabla f(x_0), x - x_0\rangle + \frac{L}{2}\|x_0 - x\|^2_{\boldsymbol{H}}\right]$$

then

$$f(x_1) - f(x_*) \leq \left(1 - \frac{\mu}{L}\right)(f(x_0) - f(x_*))\,. \tag{7.15}$$

Next we provide a short technical lemma about the convexity of functions that arises naturally in our line searching procedure.

Lemma 7.9.6. Let $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ be a convex function and and let $g(\alpha) \stackrel{\text{def}}{=} \min_{x\in S} f(x + \alpha d)$ for any convex set S and $d \in \mathbb{R}^n$. Then g is convex.

Proof. Let $\alpha, \beta \in \mathbb{R}$ and define $x_\alpha = \arg\min_{x\in S} f(x + \alpha d)$ and $x_\beta = \arg\min_{x\in S} f(x + \beta d)$. For any $t \in [0, 1]$ we have

$$\begin{aligned}
g\left(t\alpha + (1-t)\beta\right) &= \min_{x\in S} f\left(x + (t\alpha + (1-t)\beta\right) \\
&\leq f(tx_\alpha + (1-t)x_\beta + (t\alpha + (1-t)\beta)d) && \text{(Convexity of } S\text{)} \\
&\leq t\cdot f(x_\alpha + \alpha d) + (1-t)\cdot f(x_\beta + \beta\cdot d) && \text{(Convexity of } f\text{)} \\
&= t\cdot g(\alpha) + (1-t)\cdot g(\beta)
\end{aligned}$$

□

Lemma 7.9.7. For any vectors $y, z, v \in \mathbb{R}^d$ and scalar α, we can minimize $\min_{\|x-y\|_2^2\leq\alpha}\|x - z\|^2_{\boldsymbol{I}-vv^\top}$ exactly in time $O(d)$.

Proof. Let x^* be the solution of this problem. If $\|x^* - y\|_2^2 < \alpha$, then $x^* = z$. Otherwise, there is $\lambda > 0$ such that x^* is the minimizer of

$$\min_{x\in\mathbb{R}^d}\|x - z\|^2_{\boldsymbol{I}-vv^\top} + \lambda\|x - y\|_2^2\,.$$

Let $\boldsymbol{Q} = \boldsymbol{I} - vv^\top$. Then, the optimality condition of the above equation shows that

$$\boldsymbol{Q}(x^* - z) + \lambda(x^* - y) = \vec{0}\,.$$

Therefore,

$$x^* = (\boldsymbol{Q} + \lambda\boldsymbol{I})^{-1}(\boldsymbol{Q}z + \lambda y)\,. \tag{7.16}$$

Hence,

$$\alpha = \|x - y\|_2^2 = (z - y)^\top\boldsymbol{Q}(\boldsymbol{Q} + \lambda\boldsymbol{I})^{-2}\boldsymbol{Q}(z - y).$$

Let $\eta = 1 + \lambda$, then we have $(\boldsymbol{Q} + \lambda\boldsymbol{I}) = \eta\boldsymbol{I} - vv^\top$and hence Sherman–Morrison formula shows that

$$(\boldsymbol{Q} + \lambda\boldsymbol{I})^{-1} = \eta^{-1}\boldsymbol{I} + \frac{\eta^{-2}vv^\top}{1 - \|v\|^2\eta^{-1}} = \eta^{-1}\left(\boldsymbol{I} + \frac{vv^\top}{\eta - \|v\|^2}\right)\,.$$

Hence, we have

$$(\boldsymbol{Q}+\lambda \boldsymbol{I})^{-2} = \eta^{-2}\left(I + \frac{2vv^\top}{\eta - \|v\|^2} + \frac{vv^\top \|v\|^2}{(\eta-\|v\|^2)^2}\right) = \eta^{-2}\left(I + \frac{2\eta - \|v\|^2}{(\eta-\|v\|^2)^2} vv^\top\right).$$

Let $c_1 = \|\boldsymbol{Q}(z-y)\|_2^2$ and $c_2 = \left(v^\top \boldsymbol{Q}(z-y)\right)^2$, then we have

$$\alpha \quad = \quad \eta^{-2}\left(c_1 + \frac{2\eta - \|v\|^2}{(\eta-\|v\|^2)^2} c_2\right).$$

Hence, we have

$$\alpha\eta^2\left(\eta - \|v\|^2\right)^2 = c_1\left(\eta - \|v\|^2\right)^2 + c_2\left(2\eta - \|v\|^2\right).$$

Note that this is a polynomial of degree 4 in η and all coefficients can be computed in $O(d)$ time. Solving this by explicit formula, one can test all 4 possible η's into the formula (7.16) of x. Together with trivial case $x^* = z$, we simply need to check among 5 cases to check which is the solution. □

7.9.3 Noisy One Dimensional Convex Optimization

Here we show how to minimize a one dimensional convex function giving a noisy oracle for evaluating the function. While this could possibly be done using general results on convex optimization with a membership oracle, the proof in one dimension is much simpler and we provide it here for completeness.

Algorithm 20: `OneDimMinimizer`$(\ell, u, \varepsilon, g, L)$

Input: Interval $[\ell, u] \subseteq \mathbb{R}$ and target additive error $\varepsilon \in \mathbb{R}$
Input: noisy additive evaluation oracle $g : \mathbb{R} \to \mathbb{R}$ and Lipschitz bound $L > 0$
Let $x = \ell$, $y_\ell^{(0)} = \ell$, $y_u^{(0)} = u$
for $i = 1, ..., \left\lceil \log_{3/2}(\frac{L(u-\ell)}{\varepsilon}) \right\rceil$ **do**
  Let $z_\ell^{(i)} = \frac{2y_\ell^{(i-1)} + y_u^{(i-1)}}{3}$ and $z_u^{(i)} = \frac{y_\ell^{(i-1)} + 2y_u^{(i-1)}}{3}$
  if $g(z_\ell^{(i)}) \leq g(z_u^{(i)})$ **then**
    Let $(y_\ell^{(i)}, y_u^{(i)}) = (y_\ell^{(i-1)}, z_u^{(i)})$.
    If $g(z_\ell^{(i)}) \leq g(x^{(i-1)})$ update $x := z_\ell^{(i)}$..
  else if $g(z_\ell^{(i)}) > g(z_u^{(i)})$ **then**
    Let $(y_\ell^{(i)}, y_u^{(i)}) = (z_\ell^{(i)}, y_u^{(i-1)})$.
    If $g(z_u^{(i)}) \leq g(x^{(i-1)})$ update $x := z_u^{(i)}$.
  end
end
Output: x

Lemma 7.9.8. Let $f : \mathbb{R} \to \mathbb{R}$ be an L-Lipschitz convex function defined on the $[\ell, u]$ interval and let $g : \mathbb{R} \to \mathbb{R}$ be an oracle such that $|g(y) - f(y)| \leq \varepsilon$ for all y. In $O(\log(\frac{L(u-\ell)}{\varepsilon}))$ time and with $O(\log(\frac{L(u-\ell)}{\varepsilon}))$ calls to g, the algorithm `OneDimMinimizer`$(\ell, u, \varepsilon, g, L)$ outputs a point x such that

$$f(x) - \min_{y \in [\ell, u]} f(y) \leq 4\varepsilon.$$

Proof. First, note that for any $y, y' \in \mathbb{R}$ if $f(y) < f(y') - 2\varepsilon$ then$g(y) < g(y')$. This directly follows from our assumption on on g. Second, note that the output of the algorithm, x, is simply the point

queried by the algorithm (i.e. ℓ and the z_ℓ^i and z_u^i) with the smallest value of g. Combining these facts implies that $f(x)$ is within 2ε of the minimum value of f among the points queried. It thus suffices to show that the algorithm queries some point within 2ε of optimal.

To do this, we break into two cases. First, consider the case where the intervals $[y_\ell^{(i)}, y_u^{(i)}]$ all contain a minimizer of f. In this case, the final interval contains an optimum, and is of size at most $\frac{\varepsilon}{L}$. Thus, by the Lipschitz property, all points in the interval are within $\varepsilon \le 2\varepsilon$ of optimal, and at least one endpoint of the interval must have been queried by the algorithm.

For the other case, consider the last i for which this interval does contain an optimum of f. This means that $g(z_\ell^{(i)}) \le g(z_u^{(i)})$ while a minimizer x^* is to the right of $z_u^{(i)}$, or the symmetric case with a minimizer is to the left of $z_\ell^{(i)}$. Without loss of generality, we assume the former. We then have $z_\ell^{(i)} \le z_u^{(i)} \le x^*$ and $x^* - z_u^{(i)} \le z_u^{(i)} - z_\ell^{(i)}$. Consequently $z_u^{(i)} = \alpha z_l^{(i)} + (1-\alpha)x^*$ where $\alpha \in [0, \frac{1}{2}]$ and the convexity of f implies $f(z_u^{(i)}) \le \frac{1}{2} f(z_l^{(i)}) + \frac{1}{2} f(x^*))$ or equivalently $f(z_u^{(i)}) - f(x^*) \le f(z_\ell^{(i)}) - f(z_u^{(i)})$. But $f(z_\ell^{(i)}) - f(z_u^{(i)}) \le 2\varepsilon$ since $g(z_\ell^{(i)}) \le g(z_u^{(i)})$. Thus, $f(z_u^{(i)}) - f(x^*) \le 2\varepsilon$, and $z_u^{(i)}$ is queried by the algorithm, as desired. □

■ 7.10 Weighted Geometric Median

In this section, we show how to extend our results to the *weighted geometric median* problem, also known as the Weber problem: given a set of n points in d dimensions, $a^{(1)}, \ldots, a^{(n)} \in \mathbb{R}^d$, with corresponding weights $w^{(1)}, \ldots, w^{(n)} \in \mathbb{R}_{>0}$, find a point $x_* \in \mathbb{R}^d$ that minimizes the weighted sum of Euclidean distances to them:

$$x_* \in \arg\min_{x \in \mathbb{R}^d} f(x) \quad \text{where} \quad f(x) \stackrel{\text{def}}{=} \sum_{i \in [n]} w^{(i)} \|x - a^{(i)}\|_2.$$

As in the unweighted problem, our goal is to compute $(1+\varepsilon)$-approximate solution, i.e. $x \in \mathbb{R}^d$ with $f(x) \le (1+\varepsilon) f(x_*)$.

First, we show that it suffices to consider the case where the weights are integers with bounded sum (Lemma 7.10.1). Then, we show that such an instance of the weighted geometric median problem can be solved using the algorithms developed for the unweighted problem.

Lemma 7.10.1. Given points $a^{(1)}, a^{(2)}, \ldots, a^{(n)} \in \mathbb{R}^d$, non-negative weights $w^{(1)}, w^{(2)}, \ldots, w^{(n)} \in \mathbb{R}_{>0}$, and $\varepsilon \in (0,1)$, we can compute in linear time weights $w_1^{(1)}, w_1^{(2)}, \ldots, w_1^{(n)}$ such that:

- Any $(1+\varepsilon/5)$-approximate weighted geometric median of $a^{(1)}, \ldots, a^{(n)}$ with the weights $w_1^{(1)}, \ldots, w_1^{(n)}$ is also a $(1+\varepsilon)$-approximate weighted geometric median of $a^{(1)}, \ldots, a^{(n)}$ with the weights $w^{(1)}, \ldots, w^{(n)}$, and
- $w_1^{(1)}, \ldots, w_1^{(n)}$ are nonnegative integers and $\sum_{i \in [n]} w_1^{(i)} \le 5n\varepsilon^{-1}$.

Proof. Let

$$f(x) = \sum_{i \in [n]} w^{(i)} \|a^{(i)} - x\|$$

and $W = \sum_{i \in [n]} w^{(i)}$. Furthermore, let $\varepsilon' = \varepsilon/5$ and for each $i \in [n]$, define $w_0^{(i)} = \frac{n}{\varepsilon' W} w^{(i)}$, $w_1^{(i)} = \left\lfloor w_0^{(i)} \right\rfloor$, and $w_2^{(i)} = w_0^{(i)} - w_1^{(i)}$. We also define $f_0, f_1, f_2, W_0, W_1, W_2$ analogously to f and W.

Now, assume $f_1(x) \le (1+\varepsilon') f_1(x_*)$, where x_* is the minimizer of f and f_0. Then:

$$f_0(x) = f_1(x) + f_2(x) \le f_1(x) + f_2(x_*) + W_2 \|x - x_*\|_2$$

and

$$W_2\|x-x_*\|_2 = \frac{W_2}{W_1}\sum_{i\in[n]} w_1^{(i)}\|x-x_*\|_2 \le \frac{W_2}{W_1}\sum_{i\in[n]} w_1^{(i)}\left(\|x-a^{(i)}\|_2 + \|a^{(i)}-x_*\|\right)$$
$$\le \frac{W_2}{W_1}(f_1(x)+f_1(x_*))\,.$$

Now, since $W_0 = \frac{n}{\varepsilon'}$ and $W_1 \ge W_0 - n$ we have

$$\frac{W_2}{W_1} = \frac{W_0-W_1}{W_1} = \frac{W_0}{W_1} - 1 \le \frac{W_0}{W_0-n} - \frac{W_0-n}{W_0-n} = \frac{n}{\frac{n}{\varepsilon'}-n} = \frac{\varepsilon'}{1-\varepsilon'}\,.$$

Combining these yields that

$$f_0(x) \le f_1(x) + f_2(x_*) + \frac{\varepsilon'}{1-\varepsilon'}(f_1(x)+f_1(x_*))$$
$$\le \left(1+\frac{\varepsilon'}{1-\varepsilon'}\right)(1+\varepsilon')f_1(x^*) + \frac{\varepsilon'}{1-\varepsilon'}f_1(x_*) + f_2(x_*)$$
$$\le (1+5\varepsilon')f_0(x_*) = (1+\varepsilon)f_0(x_*)\,.$$

□

We now proceed to show the main result of this section.

Lemma 7.10.2. A $(1+\varepsilon)$-approximate weighted geometric median of n points in $\mathbb{R}^d$ can be computed in $O(nd\log^3\varepsilon^{-1})$ time.

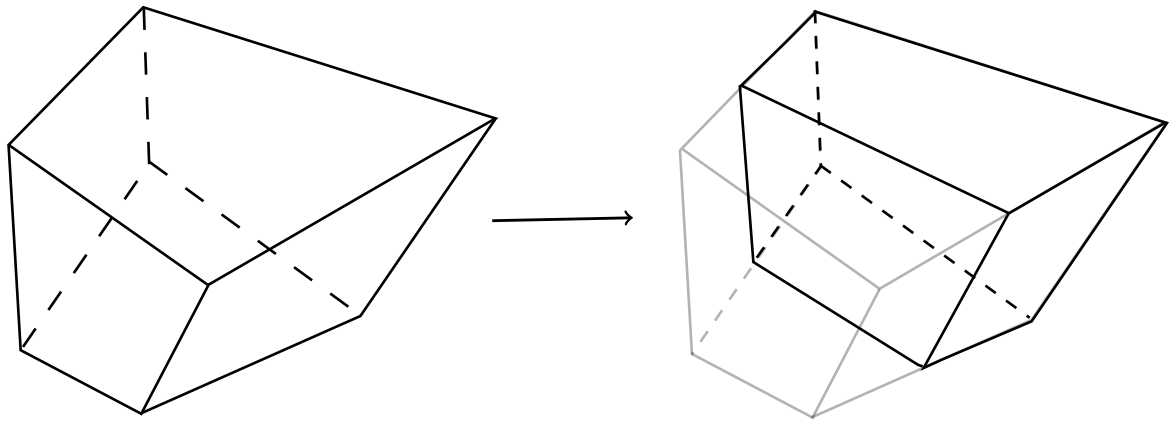
Proof. By applying Lemma 7.10.1, we can assume that the weights are integer and their sum does not exceed $n\varepsilon^{-1}$. Note that computing the weighted geometric median with such weights is equivalent to computing an unweighted geometric median of $O(n\varepsilon^{-1})$ points (where each point of the original input is repeated with the appropriate multiplicity). We now show how to simulate the behavior of our unweighted geometric median algorithms on such a set of points without computing it explicitly.

If $\varepsilon > n^{-1/2}$, we will apply the algorithm `ApproximateMedian`(ε), achieving a runtime of $O(d\varepsilon^{-2}) = O(nd)$. It is only necessary to check that we can implement weighted sampling from our points with $O(n)$ preprocessing and $O(1)$ time per sample. This is achieved by the alias method [154].

Now assume $\varepsilon < n^{-1/2}$. We will employ the algorithm `AccurateMedian`(ε). Note that the initialization using `ApproximateMedian`(2) can be performed as described in the previous paragraph. It remains to check that we can implement the subroutines `LineSearch` and `ApproxMinEig` on the implicitly represented multiset of $O(n\varepsilon^{-1})$ points. It is enough to observe only n of the points are distinct, and all computations performed by these subroutines are identical for identical points. The total runtime will thus be $O(nd\log^3(n/\varepsilon^2)) = O(nd\log^3\varepsilon^{-1})$. □

Part II

Cutting



Chapter 8

Convex Minimization In Nearly-Cubic Time

8.1 Introduction

Throughout this chapter, we study the following *feasibility problem*:

Definition 8.1.1 (Feasibility Problem)**.** Given a separation oracle (see Definition 2.3.5) for a set $K \subseteq \mathbb{R}^n$ contained in a box of radius R either find a point $\boldsymbol{x} \in K$ or prove that K does not contain a ball of radius ε.

This feasibility problem is one of the most fundamental and classic problems in optimization. Since the celebrated result of Shor [236], Yudin and Nemirovski [274] and Khachiyan [142] in 1970s essentially proving that it can be solved in time $O(\text{poly}(n) \cdot \text{SO} \cdot \log(R/\varepsilon))$, this problem has served as one of the key primitives for solving numerous problems in both combinatorial and convex optimization.

Despite the prevalence of this feasibility problem, the best known running time for solving this problem has not been improved in over 25 years. In a seminal paper of Vaidya in 1989 [253], he showed how to solve the problem in $\widetilde{O}(n \cdot \text{SO} \cdot \log(nR/\varepsilon) + n^{\omega+1} \log(nR/\varepsilon))$ time. Despite interesting generalizations and practical improvements [11, 226, 101, 20, 103, 204, 272, 102], the best theoretical guarantees for solving this problem have not been improved since.

In this chapter, we show how to improve upon Vaidya's running time in certain regimes. We provide a cutting plane algorithm which achieves an expected running time of $O(n \cdot \text{SO} \cdot \log(nR/\varepsilon) + n^3 \log^{O(1)}(nR/\varepsilon))$, improving upon the previous best known running time for the current known value of $\omega < 2.373$ [269, 99] when $R/\varepsilon = O(\text{poly}(n))$.

We achieve our results by the combination of multiple techniques. First we show how to use techniques from the work of Vaidya and Atkinson to modify Vaidya's scheme so that it is able to tolerate random noise in the computation in each iteration. We then show how to use known numerical machinery [257, 242, 164] in combination with some new techniques (Section 8.4.1 and Section 8.4.2) to implement each of these relaxed iterations efficiently. We hope that both these numerical techniques as well as our scheme for approximating complicated methods, such as Vaidya's, may find further applications.

8.1.1 Previous Work

Throughout this chapter, we restrict our attention to algorithms for the feasibility problem that have a polynomial dependence on SO, n, and $\log(R/\varepsilon)$. Such "efficient" algorithms typically follow the following iterative framework. First, they compute some trivial region Ω that contains K. Then, they call the separation oracle at some point $\boldsymbol{x} \in \Omega$. If $\boldsymbol{x} \in K$, the algorithm successfully solved the problem. If $\boldsymbol{x} \notin K$ then the separation oracle must return a half-space containing K. The algorithm then uses this half-space to shrink the region Ω while maintaining the invariant that $K \subseteq \Omega$. The algorithm then repeats this process until it finds a point $\boldsymbol{x} \in K$ or the region Ω becomes too small to contain a ball with radius ε.

Year	Algorithm	Complexity
1979	Ellipsoid Method [236, 274, 142]	$O(n^2\text{SO}\log\kappa + n^4\log\kappa)$
1988	Inscribed Ellipsoid [144, 213]	$O(n\text{SO}\log\kappa + (n\log\kappa)^{4.5})$
1989	Volumetric Center [253]	$O(n\text{SO}\log\kappa + n^{1+\omega}\log\kappa)$
1995	Analytic Center [20]	$O\left(\begin{array}{c} n\text{SO}\log^2\kappa + n^{\omega+1}\log^2\kappa \\ +(n\log\kappa)^{2+\omega/2}\end{array}\right)$
2004	Random Walk [36]	$\rightarrow O(n\text{SO}\log\kappa + n^7\log\kappa)$
-	This Chapter	$O(n\text{SO}\log\kappa + n^3\log^{O(1)}\kappa)$

Table 8.1: Algorithms for the Feasibility Problem. κ indicates nR/ε. The arrow, $\rightarrow$, indicates that it solves a more general problem where only a membership oracle is given.

Previous works on efficient algorithms for the feasibility problem all follow this iterative framework. They vary in terms of what set Ω they maintain, how they compute the center to query the separation oracle, and how they update the set. In Table 8.1, we list the previous running times for solving the feasibility problem. As usual SO indicates the cost of the separation oracle.

The first efficient algorithm for the feasibility problem is the *ellipsoid* method, due to Shor [236], Nemirovksii and Yudin [274], and Khachiyan [142]. The ellipsoid method maintains an ellipsoid as Ω and uses the center of the ellipsoid as the next query point. It takes $\Theta(n^2\log(nR/\varepsilon))$ calls of oracle which is far from the lower bound $\Omega(n\log(R/\varepsilon))$ calls [201].

To alleviate the problem, the algorithm could maintain all the information from the oracle, i.e., the polytope created from the intersection of all half-spaces obtained. The center of gravity method [168] achieves the optimal oracle complexity using this polytope and the center of gravity of this polytope as the next point. However, computing center of gravity is computationally expensive and hence we do not list its running time in Table 8.1. The Inscribed Ellipsoid Method [144] also achieved an optimal oracle complexity using this polytope as Ω but instead using the center of the maximal inscribed ellipsoid in the polytope to query the separation oracle. We listed it as occurring in year 1988 in Table 8.1 because it was [213] that yielded the first polynomial time algorithm to actually compute this maximal inscribed ellipsoid for polytope.

Vaidya [253] obtained a faster algorithm by maintaining an approximation of this polytope and using a different center, namely the volumetric center. Although the oracle complexity of this volumetric center method is very good, the algorithm is not extremely efficient as each iteration involves matrix inversion. Atkinson and Vaidya [20] showed how to avoid this computation in certain settings. However, they were unable to achieve the desired convergence rate from their method.

Bertsimas and Vempala [36] also gives an algorithm that avoids these expensive linear algebra operations while maintaining the optimal convergence rate by using techniques in sampling convex sets. Even better, this result works for a much weaker oracle, the membership oracle. However, the additional cost of this algorithm is relatively high. We remark that while there are considerable improvemenst on the sampling techniques [180, 133, 164], the additional cost is still quite high compared to standard linear algebra.

8.1.2 Challenges in Improving Previous Work

Our algorithm builds upon the previous fastest algorithm of Vaidya [253]. Ignoring implementation details and analysis, Vaidya's algorithm is quite simple. This algorithm simply maintains a polytope $P^{(k)} = \{x \in \mathbb{R}^n : \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} > \boldsymbol{0}\}$ as the current Ω and uses the *volumetric center*, the minimizer of the

following *volumetric barrier function*

$$\arg\min_{\boldsymbol{x}} \frac{1}{2}\log\det\left(\boldsymbol{A}^T\boldsymbol{S}_{\boldsymbol{x}}^{-2}\boldsymbol{A}\right) \quad \text{where} \quad \boldsymbol{S}_{\boldsymbol{x}} \stackrel{\text{def}}{=} \boldsymbol{Diag}(\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}) \tag{8.1}$$

as the point at which to query the separation oracle. The polytope is then updated by adding shifts of the half-spaces returned by the separation oracle and dropping unimportant constraints. By choosing the appropriate shift, picking the right rule for dropping constraints, and using Newton's method to compute the volumetric center he achieved a running time of $O(n \cdot SO \cdot \log\kappa + n^{1+\omega}\log\kappa)$.

While Vaidya's algorithm's dependence on SO is essentially optimal, the additional per-iteration costs of his algorithm could possibly be improved. The computational bottleneck in each iteration of Vaidya's algorithm is computing the gradient of $\log\det$ which in turn involves computing the leverage scores $\boldsymbol{\sigma}(\boldsymbol{x}) \stackrel{\text{def}}{=} \boldsymbol{diag}(\boldsymbol{S}_x^{-1}\boldsymbol{A}\left(\boldsymbol{A}^T\boldsymbol{S}_x^{-2}\boldsymbol{A}\right)^{-1}\boldsymbol{A}^T\boldsymbol{S}_x^{-1})$, a key concept in part I. As the best known algorithms for computing leverage scores exactly in this setting take time $O(n^\omega)$, directly improving the running time of Vaidya's algorithm seems challenging.

However, similar to previous chapters, these leverage scores can be computed up to a multiplicative $(1\pm\varepsilon)$ error by solving $O(\varepsilon^{-2}\log n)$ linear systems involving $\boldsymbol{A}^T\boldsymbol{S}_x^{-2}\boldsymbol{A}$. While in general this still takes time $O(\varepsilon^{-2}n^\omega)$, there are known techniques for efficiently maintaining the inverse of a matrix so that solving linear systems take amortized $O(n^2)$ time [257, 164]. Consequently if it could be shown that computing *approximate* leverage scores sufficed, this would potentially decrease the amortized cost per iteration of Vaidya's method.

Unfortunately, Vaidya's method does not seem to tolerate this type of multiplicative error. If we compute compute gradients for (8.1) using this approximation of leverage scores, it seems that the point computed would be far from the true center. Moreover, without being fairly close to the true volumetric center, it is difficult to argue that such a cutting plane method would make sufficient progress.

To overcome this issue, it is tempting to directly use the linear program result in Chapter 6. In this chapter, we faced a similar issue where a volumetric, i.e. $\log\det$, potential function had the right analytic and geometric properties, however was computational expensive to minimize. To overcome this issue the authors instead computed a weighted analytic center:

$$\arg\min_{\boldsymbol{x}} -\sum_{i\in[m]} w_i \log s_i(\boldsymbol{x}) \quad \text{where} \quad \boldsymbol{s}(\boldsymbol{x}) \stackrel{\text{def}}{=} \boldsymbol{A}\boldsymbol{x}-\boldsymbol{b} \quad .$$

For carefully chosen weights this center provides the same convergence guarantees as the volumetric potential function, while each step can be computed by solving few linear systems (rather than forming the matrix inverse).

Unfortunately, it is unclear how to directly extend the work in Chapter 6 on solving an explicit linear program to the feasibility problem specified by a separation oracle. While it is possible to approximate the volumetric barrier by a weighted analytic center in many respects, proving that this approximation suffices for fast convergence remains open. In fact, the volumetric barrier function as used in Vaidya's algorithm is well approximated simply by the standard analytic center

$$\arg\min_{\boldsymbol{x}} -\sum_{i\in[m]} \log s_i(\boldsymbol{x}) \quad \text{where} \quad \boldsymbol{s}(\boldsymbol{x}) \stackrel{\text{def}}{=} \boldsymbol{A}\boldsymbol{x}-\boldsymbol{b} \quad .$$

as all the unimportant constraints are dropped during the algorithm. However, despite decades of research, the best running times known for solving the feasibility problem using the analytic center are Vaidya and Atkinson algorithm from 1995 [20]. While the running time of this algorithm could possibly be improved using approximate leverage score computations and amortized efficient linear

system solvers, unfortunately at best, without further insight this would yield an algorithm which requires a suboptimal $O(n \log^{O(1)} \kappa)$ queries to the separation oracle.

As pointed out in [20], the primary difficulty in using any sort of analytic center is quantifying the amount of progress made in each step. We still believe providing direct near-optimal analysis of weighted analytic center is a tantalizing open question warranting further investigation. However, rather than directly address the question of the performance of weighted analytic centers for the feasibility problem, we take a slightly different approach that side-steps this issue. We provide a partial answer that still sheds some light on the performance of the weighted analytic center while still providing our desired running time improvements.

8.1.3 Our Approach

To overcome the shortcoming of the volumetric and analytic centers we instead consider a hybrid barrier function

$$\arg\min_{\boldsymbol{x}} - \sum_{i \in [m]} w_i \log s_i(\boldsymbol{x}) + \log\det(\boldsymbol{A}^T \boldsymbol{S}_x^{-1} \boldsymbol{A}) \quad \text{where} \quad \boldsymbol{s}(\boldsymbol{x}) \stackrel{\text{def}}{=} \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} \quad .$$

for careful chosen weights. Our key observation is that for correct choice of weights, we can compute the gradient of this potential function. In particular if we let $\boldsymbol{w} = \boldsymbol{\tau} - \boldsymbol{\sigma}(\boldsymbol{x})$ then the gradient of this potential function is the same as the gradients of $\sum_{i \in [m]} \tau_i \log s_i(\boldsymbol{x})$, which we can compute efficiently. Moreover, since we are using $\log\det$, we can use analysis similar to Vaidya's algorithm [253] to analyze the convergence rate of this algorithm.

Unfortunately, this is a simple observation and does not immediately change the problem substantially. It simply pushes the problem of computing gradients of $\log\det$ to computing $\boldsymbol{w}$. Therefore, for this scheme to work, we would need to ensure that the weights do not change too much and that when they change, they do not significantly hurt the progress of our algorithm. In other words, for this scheme to work, we would still need very precise estimates of leverage scores.

However, we note that the leverage scores $\boldsymbol{\sigma}(\boldsymbol{x})$ do not change too much between iterations. Moreover, we provide what we believe is an interesting technical result that an unbiased estimates to the changes in leverage scores can be computed using linear system solvers such that the *total error* of the estimate is bounded by the total change of the leverage scores (See Section 8.4.1). Using this result our scheme simply follows Vaidya's basic scheme in [253], however instead of minimizing the hybrid barrier function directly we alternate between taking Newton steps we can compute, changing the weights so that we can still compute Newton steps, and computing accurate unbiased estimates of the changes in the leverage scores so that the weights do not change adversarially by too much.

To make this scheme work, there are two additional details that need to be dealt with. First, we cannot let the weights vary too much as this might ultimately hurt the rate of progress of our algorithm. Therefore, in every iteration we compute a single leverage score to high precision to control the value of w_i and we show that by careful choice of the index we can ensure that no weight gets too large (See Section 8.4.2).

Second, we need to show that changing weights does not affect our progress by much more than the progress we make with respect to $\log\det$. To do this, we need to show the slacks are bounded above and below. We enforce this by adding regularization terms and consider the potential function

$$p_{\boldsymbol{e}}(\boldsymbol{x}) = - \sum_{i \in [m]} w_i \log s_i(\boldsymbol{x}) + \frac{1}{2} \log\det\left(\boldsymbol{A}^T \boldsymbol{S}_x^{-2} \boldsymbol{A} + \lambda \boldsymbol{I}\right) + \frac{\lambda}{2} \|x\|_2^2$$

This allows us to ensure that the entries of $\boldsymbol{s}(\boldsymbol{x})$ do not get too large or too small and therefore

changing the weighting of the analytic center cannot affect the function value too much.

Third, we need to make sure our potential function is convex. If we simply take $\boldsymbol{w} = \boldsymbol{\tau} - \boldsymbol{\sigma}(\boldsymbol{x})$ with $\boldsymbol{\tau}$ as an estimator of $\boldsymbol{\sigma}(\boldsymbol{x})$, $\boldsymbol{w}$ can be negative and the potential function could be non-convex. To circumvent this issue, we use $\boldsymbol{w} = c_e + \boldsymbol{\tau} - \boldsymbol{\sigma}(\boldsymbol{x})$ and make sure $\|\boldsymbol{\tau} - \boldsymbol{\sigma}(\boldsymbol{x})\|_\infty < c_e$.

Combining these insights, using efficient algorithms for solving a sequence of slowly changing linear systems [257, 164], and providing careful analysis ultimately allows us to achieve a running time of $O(n\mathrm{SO}\log\kappa + n^3\log^{O(1)}\kappa)$ for the feasibility problem. Furthermore, in the case that K does not contain a ball of radius ε, our algorithm provides a proof that the polytope does not contain a ball of radius ε. This proof ultimately allows us to achieve running time improvements for strongly polynomial submodular minimization in Chapter 10.

8.1.4 Organization

The rest of Chapter 8 is organized as follows. In Section 8.2 we provide some preliminary information and notation we use throughout Chapter 8. In Section 8.3 we then provide and analyze our cutting plane method. In Section 8.4 we provide key technical tools which may be of independent interest.

8.2 Preliminaries

Here we introduce some notation and concepts we use throughout this chapter.

8.2.1 Hybrid Barrier Function

As explained in Section 8.1.3 our cutting plane method maintains a polytope $P = \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b}\}$ for $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ and $\boldsymbol{b} \in \mathbb{R}^n$ that contains some target set K. We then maintain a minimizer of the following hybrid barrier function:

$$p_{\boldsymbol{e}}(\boldsymbol{x}) \stackrel{\text{def}}{=} -\sum_{i\in[m]} (c_e + e_i)\log\left(s_i(\boldsymbol{x})/R\right) + \frac{1}{2}\log\det\left(R^2\left(\boldsymbol{A}^T\boldsymbol{S}_x^{-2}\boldsymbol{A} + \lambda\boldsymbol{I}\right)\right) + \frac{\lambda}{2}\|x\|_2^2$$

where $\boldsymbol{e} \in \mathbb{R}^m$ is a variable we maintain, $c_e \geq 0$ and $\lambda \geq 0$ are constants we fix later, $\boldsymbol{s}(\boldsymbol{x}) \stackrel{\text{def}}{=} \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}$, $\boldsymbol{S}_x = \boldsymbol{Diag}(\boldsymbol{s}(\boldsymbol{x}))$, and R is the radius of the box containing P. When the meaning is clear from context we often use the shorthand $\boldsymbol{A}_x \stackrel{\text{def}}{=} \boldsymbol{S}_x^{-1}\boldsymbol{A}$. The R term is only used in the proof and can be ignored in the algorithm because it only affects a constant term in the potential function.

Rather than maintaining $\boldsymbol{e}$ explicitly, we instead maintain a vector $\boldsymbol{\tau} \in \mathbb{R}^m$ that approximates the leverage score

$$\boldsymbol{\psi}(\boldsymbol{x}) \stackrel{\text{def}}{=} \boldsymbol{diag}\left(\boldsymbol{A}_x\left(\boldsymbol{A}_x^T\boldsymbol{A}_x + \lambda\boldsymbol{I}\right)^{-1}\boldsymbol{A}_x^T\right) \quad .$$

and pick $\boldsymbol{e}$ using the function $\boldsymbol{e}_P(\boldsymbol{\tau}, \boldsymbol{x}) \stackrel{\text{def}}{=} \boldsymbol{\tau} - \boldsymbol{\psi}(\boldsymbol{x})$. Note that $\boldsymbol{\psi}(\boldsymbol{x})$ is simply the leverage scores of certain rows of the matrix

$$\begin{bmatrix} \boldsymbol{A}_x \\ \sqrt{\lambda}\boldsymbol{I} \end{bmatrix}.$$

and therefore the usual properties of leverage scores hold, i.e. $\psi_i(\boldsymbol{x}) \in (0, 1)$ and $\|\psi_i(\boldsymbol{x})\|_1 \leq n$. We write $\boldsymbol{\psi}(\boldsymbol{x})$ equivalently as $\boldsymbol{\psi}_x$ or $\boldsymbol{\psi}_P$ when we want the matrix to be clear. Furthermore, we let $\boldsymbol{\Psi}_x \stackrel{\text{def}}{=} \boldsymbol{Diag}(\boldsymbol{\psi}(\boldsymbol{x}))$ and $\mu(\boldsymbol{x}) \stackrel{\text{def}}{=} \min_i \psi_i(\boldsymbol{x})$. Again, we use the subscripts of x and P interchangeably and often drop them when the meaning is clear from context.

We remark that the last term $\frac{\lambda}{2}\|x\|_2^2$ ensures that our point is always within a certain region (Lemma 8.3.18) and hence the term $(c_e + e_i)\log s_i(\boldsymbol{x})_i$ never gets too large. However, this ℓ^2 term

changes the Hessian of the potential function and hence we need to put a $\lambda \boldsymbol{I}$ term inside both the log det and the leverage score to reflect this. This is the reason why we use $\boldsymbol{\psi}$ instead of the standard leverage score.

8.3 Our Cutting Plane Method

In this section we develop and prove the correctness of our cutting plane method. We break the presentation and proof of correctness of our cutting plane methods into multiple parts. First in Section 8.3.1 we describe how we maintain a center of the hybrid barrier function p_e and analyze this procedure. Then, in Section 8.3.2 we carefully analyze the effect of changing constraints on the hybrid barrier function and in Section 8.3.3 we prove properties of an approximate center of hybrid barrier function, which we call a hybrid center. In Section 8.3.4 we then provide our cutting plane method and in Section 8.3.5 we prove that the cutting plane method solves the feasibility problem as desired.

8.3.1 Centering

In this section we show how to compute approximate *centers* or minimizers of the hybrid barrier function for the current polytope $P = \{\boldsymbol{x} : \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b}\}$. We split this proof up into multiple parts. First we simply bound the gradient and Hessian of the hybrid barrier function, p_e, as follows.

Lemma 8.3.1. *For* $f(\boldsymbol{x}) \stackrel{\text{def}}{=} \frac{1}{2}\log\det\left(\boldsymbol{A}^T\boldsymbol{S}_x^{-2}\boldsymbol{A} + \lambda\boldsymbol{I}\right)$, *we have that*

$$\nabla f(\boldsymbol{x}) = -\boldsymbol{A}_x^T\boldsymbol{\psi}(\boldsymbol{x}) \quad \text{and} \quad \boldsymbol{A}_x^T\boldsymbol{\Psi}(\boldsymbol{x})\boldsymbol{A}_x \preceq \nabla^2 f(\boldsymbol{x}) \preceq 3\boldsymbol{A}_x^T\boldsymbol{\Psi}(\boldsymbol{x})\boldsymbol{A}_x \quad .$$

Proof. Our proof is similar to [10, Appendix] which proved the statement when $\lambda = 0$. This case does not change the derivation significantly, however for completeness we include the proof below.

We take derivatives on $\boldsymbol{s}$ first and then apply chain rule. Let $f(\boldsymbol{s}) = \frac{1}{2}\log\det\left(\boldsymbol{A}^T\boldsymbol{S}^{-2}\boldsymbol{A} + \lambda\boldsymbol{I}\right)$. We use the notation $Df(\boldsymbol{x})[\boldsymbol{h}]$ to denote the directional derivative of f along the direction $\boldsymbol{h}$ at the point $\boldsymbol{x}$. Using the standard formula for the derivative of log det, i.e. $\frac{d}{dt}\log\det\boldsymbol{B}_t = \mathrm{Tr}((\boldsymbol{B}_t)^{-1}(\frac{d\boldsymbol{B}_t}{dt}))$, we have

$$\begin{aligned} Df(\boldsymbol{s})[\boldsymbol{h}] &= \frac{1}{2}\mathrm{Tr}((\boldsymbol{A}^T\boldsymbol{S}^{-2}\boldsymbol{A} + \lambda\boldsymbol{I})^{-1}(\boldsymbol{A}^T(-2)\boldsymbol{S}^{-3}\boldsymbol{H}\boldsymbol{A})) \\ &= -\sum_i \frac{h_i}{s_i}\mathbf{1}_i^T\boldsymbol{S}^{-1}\boldsymbol{A}\left(\boldsymbol{A}^T\boldsymbol{S}^{-2}\boldsymbol{A} + \lambda\boldsymbol{I}\right)^{-1}\boldsymbol{A}\boldsymbol{S}^{-1}\mathbf{1}_i = -\sum_i \frac{\psi_i h_i}{s_i} \quad . \end{aligned} \tag{8.2}$$

Applying chain rules, we have $\nabla f(\boldsymbol{x}) = -\boldsymbol{A}_x^T\boldsymbol{\psi}$. Now let $\boldsymbol{P} \stackrel{\text{def}}{=} \boldsymbol{S}^{-1}\boldsymbol{A}\left(\boldsymbol{A}^T\boldsymbol{S}^{-2}\boldsymbol{A} + \lambda\boldsymbol{I}\right)^{-1}\boldsymbol{A}^T\boldsymbol{S}^{-1}$. Taking the derivative of (8.2) again and using the cyclic property of trace, we have

$$\begin{aligned} D^2 f(\boldsymbol{s})[\boldsymbol{h}_1, \boldsymbol{h}_2] &= \mathrm{Tr}\left(\left(\boldsymbol{A}^T\boldsymbol{S}^{-2}\boldsymbol{A} + \lambda\boldsymbol{I}\right)^{-1}\left(\boldsymbol{A}^T(-2)\boldsymbol{S}^{-3}\boldsymbol{H}_2\boldsymbol{A}\right)\left(\boldsymbol{A}^T\boldsymbol{S}^{-2}\boldsymbol{A} + \lambda\boldsymbol{I}\right)^{-1}\left(\boldsymbol{A}^T\boldsymbol{S}^{-3}\boldsymbol{H}_1\boldsymbol{A}\right)\right) \\ &\quad -\mathrm{Tr}\left(\left(\boldsymbol{A}^T\boldsymbol{S}^{-2}\boldsymbol{A} + \lambda\boldsymbol{I}\right)^{-1}\left(\boldsymbol{A}^T(-3)\boldsymbol{S}^{-4}\boldsymbol{H}_2\boldsymbol{H}_1\boldsymbol{A}\right)\right) \\ &= 3\mathrm{Tr}\left(\boldsymbol{P}\boldsymbol{S}^{-2}\boldsymbol{H}_2\boldsymbol{H}_1\right) - 2\mathrm{Tr}\left(\boldsymbol{P}\boldsymbol{S}^{-1}\boldsymbol{H}_2\boldsymbol{P}\boldsymbol{S}^{-1}\boldsymbol{H}_1\right) \\ &= 3\sum_i P_{ii}\frac{\boldsymbol{h}_1(i)\boldsymbol{h}_2(i)}{s_i^2} - 2\sum_{ij} P_{ij}\frac{\boldsymbol{h}_2(j)}{s_j}P_{ji}\frac{\boldsymbol{h}_2(i)}{s_i} \\ &= 3\sum_i \psi_i\frac{\boldsymbol{h}_1(i)\boldsymbol{h}_2(i)}{s_i^2} - 2\sum_{ij} P_{ij}^2\frac{\boldsymbol{h}_2(j)}{s_j}\frac{\boldsymbol{h}_2(i)}{s_i} \quad . \end{aligned}$$

Consequently, $D^2 f(\boldsymbol{x})[\mathbf{1}_i, \mathbf{1}_j] = [\boldsymbol{S}^{-1}\left(3\boldsymbol{\Psi} - 2\boldsymbol{P}^{(2)}\right)\boldsymbol{S}^{-1}]_{ij}$ where $\boldsymbol{P}^{(2)}$ is the Schur product of $\boldsymbol{P}$ with itself.

Now note that

$$\begin{aligned}\sum_i P_{ij}^2 &= \mathbf{1}_j \boldsymbol{S}^{-1}\boldsymbol{A}\left(\boldsymbol{A}^T\boldsymbol{S}^{-2}\boldsymbol{A} + \lambda\boldsymbol{I}\right)^{-1}\boldsymbol{A}^T\boldsymbol{S}^{-2}\boldsymbol{A}\left(\boldsymbol{A}^T\boldsymbol{S}^{-2}\boldsymbol{A} + \lambda\boldsymbol{I}\right)^{-1}\boldsymbol{A}^T\boldsymbol{S}^{-1}\mathbf{1}_j \\ &\leq \mathbf{1}_j \boldsymbol{S}^{-1}\boldsymbol{A}\left(\boldsymbol{A}^T\boldsymbol{S}^{-2}\boldsymbol{A} + \lambda\boldsymbol{I}\right)^{-1}\boldsymbol{A}^T\boldsymbol{S}^{-1}\mathbf{1}_j = P_{jj} = \boldsymbol{\Psi}_{jj} \quad .\end{aligned}$$

Hence, the Gershgorin circle theorem shows that the eigenvalues of $\boldsymbol{\Psi} - \boldsymbol{P}^{(2)}$ are lies in union of the interval $[0, 2\psi_j]$ over all j. Hence, $\boldsymbol{\Psi} - \boldsymbol{P}^{(2)} \succeq \mathbf{0}$. On the other hand, Schur product theorem shows that $\boldsymbol{P}^{(2)} \succeq \mathbf{0}$ as $\boldsymbol{P} \succeq \mathbf{0}$. Hence, the result follows by chain rule. □

Lemma 8.3.1 immediately shows that under our choice of $\boldsymbol{e} = \boldsymbol{e}_P(\boldsymbol{x}, \boldsymbol{\tau})$ we can compute the gradient of the hybrid barrier function, $p_{\boldsymbol{e}}(\boldsymbol{x})$ efficiently. Formally, Lemma 8.3.1 immediately implies the following:

Lemma 8.3.2 (Gradient). *For $\boldsymbol{x} \in P = \{\boldsymbol{y} \in \mathbb{R}^n : \boldsymbol{A}\boldsymbol{y} \geq \boldsymbol{b}\}$ and $\boldsymbol{e} \in \mathbb{R}^m$ we have*

$$\nabla p_{\boldsymbol{e}}(\boldsymbol{x}) = -\boldsymbol{A}_x^T(c_e\mathbf{1} + \boldsymbol{e} + \boldsymbol{\psi}_P(\boldsymbol{x})) + \lambda\boldsymbol{x}$$

and therefore for all $\boldsymbol{\tau} \in \mathbb{R}^m$, we have

$$\nabla p_{\boldsymbol{e}(\boldsymbol{\tau},\boldsymbol{x})}(\boldsymbol{x}) = -\boldsymbol{A}_x^T\left(c_e\mathbf{1} + \boldsymbol{\tau}\right) + \lambda\boldsymbol{x}.$$

Remark 8.3.3. To be clear, the vector $\nabla p_{\boldsymbol{e}(\boldsymbol{\tau},\boldsymbol{x})}(\boldsymbol{x})$ is defined as the vector such that

$$[\nabla p_{\boldsymbol{e}(\boldsymbol{\tau},\boldsymbol{x})}(\boldsymbol{x})]_i = \lim_{t\to 0}\frac{1}{t}\left(p_{\boldsymbol{e}(\boldsymbol{\tau},\boldsymbol{x})}(\boldsymbol{x} + t\mathbf{1}_i) - p_{\boldsymbol{e}(\boldsymbol{\tau},\boldsymbol{x})}(\boldsymbol{x})\right) \quad .$$

In other words, we treat the parameter $\boldsymbol{e}(\boldsymbol{\tau}, \boldsymbol{x})$ as fixed. This is the reason we denote it by subscript to emphasize that $p_{\boldsymbol{e}}(\boldsymbol{x})$ is a family of functions, $p_{\boldsymbol{e}(\boldsymbol{\tau},\boldsymbol{x})}$ is one particular function, and $\nabla p_{\boldsymbol{e}(\boldsymbol{\tau},\boldsymbol{x})}$ means taking gradient on that particular function.

Consequently, we can always compute $\nabla p_{\boldsymbol{e}(\boldsymbol{\tau},\boldsymbol{x})}(\boldsymbol{x})$ efficiently. Now, we measure *centrality* or how close we are to the hybrid center as follows.

Definition 8.3.4 (Centrality). For $\boldsymbol{x} \in P = \{\boldsymbol{y} \in \mathbb{R}^n : \boldsymbol{A}\boldsymbol{y} \geq \boldsymbol{b}\}$ and $\boldsymbol{e} \in \mathbb{R}^m$, we define the *centrality* of $\boldsymbol{x}$ by

$$\delta_{\boldsymbol{e}}(\boldsymbol{x}) \stackrel{\text{def}}{=} \|\nabla p_{\boldsymbol{e}}(\boldsymbol{x})\|_{\boldsymbol{H}(\boldsymbol{x})^{-1}}$$

where $\boldsymbol{H}(\boldsymbol{x}) \stackrel{\text{def}}{=} \boldsymbol{A}_x^T\left(c_e\boldsymbol{I} + \boldsymbol{\Psi}(\boldsymbol{x})\right)\boldsymbol{A}_x + \lambda\boldsymbol{I}$. Often, we use *weights* $\boldsymbol{w} \in \mathbb{R}^m_{>0}$ to approximate this Hessian and consider $\boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{w}) \stackrel{\text{def}}{=} \boldsymbol{A}_x^T\left(c_e\boldsymbol{I} + \boldsymbol{W}\right)\boldsymbol{A}_x + \lambda\boldsymbol{I}$.

Next, we bound how much slacks can change in a region close to a nearly central point.

Lemma 8.3.5. *Let $\boldsymbol{x} \in P = \{\boldsymbol{y} \in \mathbb{R}^n : \boldsymbol{A}\boldsymbol{y} \geq \boldsymbol{b}\}$ and $\boldsymbol{y} \in \mathbb{R}^n$ such that $\|\boldsymbol{x} - \boldsymbol{y}\|_{\boldsymbol{H}(\boldsymbol{x})} \leq \varepsilon\sqrt{c_e + \mu(\boldsymbol{x})}$ for $\varepsilon < 1$. Then $\boldsymbol{y} \in P$ and $(1-\varepsilon)\boldsymbol{S}_x \preceq \boldsymbol{S}_y \preceq (1+\varepsilon)\boldsymbol{S}_x$.*

Proof. Direct calculation reveals the following:

$$\begin{aligned}\|\boldsymbol{S}_x^{-1}(\boldsymbol{s}_{\boldsymbol{y}} - \boldsymbol{s}_x)\|_\infty \leq \|\boldsymbol{A}_x(\boldsymbol{y} - \boldsymbol{x})\|_2 &\leq \frac{1}{\sqrt{c_e + \mu(\boldsymbol{x})}}\|\boldsymbol{A}_x(\boldsymbol{y} - \boldsymbol{x})\|_{c_e\boldsymbol{I} + \boldsymbol{\Psi}(\boldsymbol{x})} \\ &\leq \frac{1}{\sqrt{c_e + \mu(\boldsymbol{x})}}\|\boldsymbol{y} - \boldsymbol{x}\|_{\boldsymbol{H}(\boldsymbol{x})} \leq \varepsilon \quad .\end{aligned}$$

Consequently, $(1-\varepsilon)\boldsymbol{S}_x \preceq \boldsymbol{S}_y \preceq (1+\varepsilon)\boldsymbol{S}_x$. Since $y \in P$ if and only if $\boldsymbol{S}_y \succeq \mathbf{0}$ the result follows. □

Combining the previous lemmas we obtain the following.

Lemma 8.3.6. *Let $\boldsymbol{x} \in P = \{\boldsymbol{y} \in \mathbb{R}^n : \boldsymbol{A}\boldsymbol{y} \geq \boldsymbol{b}\}$ and $\boldsymbol{e}, \boldsymbol{w} \in \mathbb{R}^m$ such that $\|\boldsymbol{e}\|_\infty \leq \frac{1}{2}c_e \leq 1$ and $\boldsymbol{\Psi}(\boldsymbol{x}) \preceq \boldsymbol{W} \preceq \frac{4}{3}\boldsymbol{\Psi}(\boldsymbol{x})$. If $\boldsymbol{y} \in \mathbb{R}^n$ satisfies $\|\boldsymbol{x} - \boldsymbol{y}\|_{\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{w})} \leq \frac{1}{10}\sqrt{c_e + \mu(\boldsymbol{x})}$, then*

$$\frac{1}{4}\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{w}) \preceq \nabla^2 p_{\boldsymbol{e}}(\boldsymbol{y}) \preceq 8\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{w}) \quad \text{and} \quad \frac{1}{2}\boldsymbol{H}(\boldsymbol{x}) \preceq \boldsymbol{H}(\boldsymbol{y}) \preceq 2\boldsymbol{H}(\boldsymbol{x}) \quad .$$

Also, we have that $\boldsymbol{H}(\boldsymbol{x}) \preceq \boldsymbol{Q}(\boldsymbol{x},\boldsymbol{w}) \preceq \frac{4}{3}\boldsymbol{H}(\boldsymbol{x})$.

Proof. Lemma 8.3.1 shows that

$$\boldsymbol{A}_y^T \left(c_e\boldsymbol{I} + \boldsymbol{E} + \boldsymbol{\Psi}(\boldsymbol{y})\right)\boldsymbol{A}_y + \lambda\boldsymbol{I} \preceq \nabla^2 p_{\boldsymbol{e}}(\boldsymbol{y}) \preceq \boldsymbol{A}_y^T \left(c_e\boldsymbol{I} + \boldsymbol{E} + 3\boldsymbol{\Psi}(\boldsymbol{y})\right)\boldsymbol{A}_y + \lambda\boldsymbol{I} \quad . \tag{8.3}$$

Since $\boldsymbol{W} \succeq \boldsymbol{\Psi}$, we have that $\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{w}) \succeq \boldsymbol{H}(\boldsymbol{x})$ and therefore $\|\boldsymbol{x}-\boldsymbol{y}\|_{\boldsymbol{H}(\boldsymbol{x})} \leq \varepsilon\sqrt{c_e + \mu(\boldsymbol{x})}$ with $\varepsilon = 0.1$. Consequently, by Lemma 8.3.5 we have $(1-\varepsilon)\boldsymbol{S}_x \preceq \boldsymbol{S}_y \preceq (1+\varepsilon)\boldsymbol{S}_x$ and therefore

$$\frac{(1-\varepsilon)^2}{(1+\varepsilon)^2}\boldsymbol{\Psi}(\boldsymbol{x}) \preceq \boldsymbol{\Psi}(\boldsymbol{y}) \preceq \frac{(1+\varepsilon)^2}{(1-\varepsilon)^2}\boldsymbol{\Psi}(\boldsymbol{x})$$

and

$$\frac{1}{2}\boldsymbol{H}(\boldsymbol{x}) \preceq \frac{(1-\varepsilon)^2}{(1+\varepsilon)^4}\boldsymbol{H}(\boldsymbol{x}) \preceq \boldsymbol{H}(\boldsymbol{y}) \preceq \frac{(1+\varepsilon)^2}{(1-\varepsilon)^4}\boldsymbol{H}(\boldsymbol{x}) \preceq 2\boldsymbol{H}(\boldsymbol{x})$$

Furthermore, (8.3) shows that

$$\begin{aligned}
\nabla^2 p_{\boldsymbol{e}}(\boldsymbol{y}) &\preceq \boldsymbol{A}_y^T \left(c_e\boldsymbol{I} + \boldsymbol{E} + 3\boldsymbol{\Psi}(\boldsymbol{y})\right)\boldsymbol{A}_y + \lambda\boldsymbol{I} \\
&\preceq \frac{(1+\varepsilon)^2}{(1-\varepsilon)^4}\boldsymbol{A}_x^T \left(c_e\boldsymbol{I} + \boldsymbol{E} + 3\boldsymbol{\Psi}(\boldsymbol{x})\right)\boldsymbol{A}_x + \lambda\boldsymbol{I} \\
&\preceq 2\boldsymbol{A}_x^T \left(\frac{3}{2}c_e\boldsymbol{I} + 3\boldsymbol{W}\right)\boldsymbol{A}_x + \lambda\boldsymbol{I} \preceq 8\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{w})
\end{aligned}$$

and

$$\begin{aligned}
\nabla^2 p_{\boldsymbol{e}}(\boldsymbol{y}) &\succeq \boldsymbol{A}_y^T \left(c_e\boldsymbol{I} + \boldsymbol{E} + \boldsymbol{\Psi}(\boldsymbol{y})\right)\boldsymbol{A}_y + \lambda\boldsymbol{I} \\
&\succeq \frac{(1-\varepsilon)^4}{(1+\varepsilon)^2}\boldsymbol{A}_x^T \left(c_e\boldsymbol{I} + \boldsymbol{E} + \boldsymbol{\Psi}(\boldsymbol{x})\right)\boldsymbol{A}_x + \lambda\boldsymbol{I} \\
&\succeq \frac{1}{2}\boldsymbol{A}_x^T \left(\frac{1}{2}c_e\boldsymbol{I} + \frac{3}{4}\boldsymbol{W}\right)\boldsymbol{A}_x + \lambda\boldsymbol{I} \succeq \frac{1}{4}\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{w}).
\end{aligned}$$

The last inequality follows from the definition of $\boldsymbol{H}(\boldsymbol{x})$ and $\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{w})$ and the fact $\boldsymbol{\Psi}(\boldsymbol{x}) \preceq \boldsymbol{W} \preceq \frac{4}{3}\boldsymbol{\Psi}(\boldsymbol{x})$. □

To analyze our centering scheme we use standard facts about gradient descent we prove in Lemma 8.3.7.

Lemma 8.3.7 (Gradient Descent)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be twice differentiable and $\boldsymbol{Q} \in \mathbb{R}^{n\times n}$ be positive definite. Let $\boldsymbol{x}_0 \in \mathbb{R}^n$ and $\boldsymbol{x}_1 \stackrel{\text{def}}{=} \boldsymbol{x}_0 - \frac{1}{L}\boldsymbol{Q}^{-1}\nabla f(\boldsymbol{x}_0)$. Furthermore, let $\boldsymbol{x}_\alpha = \boldsymbol{x}_0 + \alpha(\boldsymbol{x}_1 - \boldsymbol{x})$ and suppose that $\mu\boldsymbol{Q} \preceq \nabla^2 f(\boldsymbol{x}_\alpha) \preceq L\boldsymbol{Q}$ for all $\alpha \in [0,1]$. Then,*

1. $\|\nabla f(\boldsymbol{x}_1)\|_{\boldsymbol{Q}^{-1}} \leq \left(1 - \frac{\mu}{L}\right)\|\nabla f(\boldsymbol{x}_0)\|_{\boldsymbol{Q}^{-1}}$
2. $f(\boldsymbol{x}_1) \geq f(\boldsymbol{x}_0) - \frac{1}{L}\|\nabla f(\boldsymbol{x}_0)\|^2_{\boldsymbol{Q}^{-1}}$

Proof. Integrating we have that

$$\nabla f(\boldsymbol{x}_1) = \nabla f(\boldsymbol{x}_0) + \int_0^1 \nabla^2 f(\boldsymbol{x}_\alpha)(\boldsymbol{x}_1 - \boldsymbol{x}_0)d\alpha = \int_0^1 \left(\boldsymbol{Q} - \frac{1}{L}\nabla^2 f(\boldsymbol{x}_\alpha)\right)\boldsymbol{Q}^{-1}\nabla f(\boldsymbol{x}_0)d\alpha$$

Consequently, by applying Jensen's inequality we have

$$\begin{aligned}\|\nabla f(\boldsymbol{x}_1)\|_{\boldsymbol{Q}^{-1}} &= \left\|\int_0^1 \left(\boldsymbol{Q} - \frac{1}{L}\nabla^2 f(\boldsymbol{x}_\alpha)\right)\boldsymbol{Q}^{-1}\nabla f(\boldsymbol{x}_0)d\alpha\right\|_{\boldsymbol{Q}^{-1}} \\ &\leq \int_0^1 \left\|\left(\boldsymbol{Q} - \frac{1}{L}\nabla^2 f(\boldsymbol{x}_\alpha)\right)\boldsymbol{Q}^{-1}\nabla f(\boldsymbol{x}_0)\right\|_{\boldsymbol{Q}^{-1}} d\alpha \\ &\leq \|\boldsymbol{Q}^{-1/2}\nabla f(\boldsymbol{x}_0)\|_{\left[\boldsymbol{Q}^{-1/2}\left(\boldsymbol{Q} - \frac{1}{L}\nabla^2 f(\boldsymbol{x}_\alpha)\right)\boldsymbol{Q}^{-1/2}\right]^2}\end{aligned}$$

Now we know that by assumption that

$$\boldsymbol{0} \preceq \boldsymbol{Q}^{-1/2}\left(\boldsymbol{Q} - \frac{1}{L}\nabla^2 f(\boldsymbol{x}_\alpha)\right)\boldsymbol{Q}^{-1/2} \preceq \left(1 - \frac{\mu}{L}\right)\boldsymbol{I}$$

and therefore combining these (1) holds.

Using the convexity of f, we have

$$\begin{aligned} f(\boldsymbol{x}_1) &\geq f(\boldsymbol{x}_0) + \langle \nabla f(\boldsymbol{x}_0), \boldsymbol{x}_1 - \boldsymbol{x}_0\rangle \\ &\geq f(\boldsymbol{x}_0) - \|\nabla f(\boldsymbol{x}_0)\|_{\boldsymbol{Q}^{-1}}\|\boldsymbol{x}_1 - \boldsymbol{x}_0\|_{\boldsymbol{Q}}\end{aligned}$$

and since $\|\boldsymbol{x}_1 - \boldsymbol{x}_0\|_{\boldsymbol{Q}} = \frac{1}{L}\|\nabla f(\boldsymbol{x}_0)\|_{\boldsymbol{Q}^{-1}}$, (2) holds as well. □

Next we bound the effect of changing $\boldsymbol{e}$ on the hybrid barrier function $p_{\boldsymbol{e}}(\boldsymbol{x})$.

Lemma 8.3.8. *For $\boldsymbol{x} \in P = \{\boldsymbol{y} \in \mathbb{R}^n : \boldsymbol{A}\boldsymbol{y} \geq \boldsymbol{b}\}$, $\boldsymbol{e}, \boldsymbol{f} \in \mathbb{R}^m$, and $\boldsymbol{w} \in \mathbb{R}^m_{>0}$ such that $\boldsymbol{W} \succeq \boldsymbol{\Psi}(\boldsymbol{x})$*

$$\|\nabla p_{\boldsymbol{f}}(\boldsymbol{x})\|_{\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{w})^{-1}} \leq \|\nabla p_{\boldsymbol{e}}(\boldsymbol{x})\|_{\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{w})^{-1}} + \frac{1}{\sqrt{c_e + \mu(\boldsymbol{x})}}\|\boldsymbol{f} - \boldsymbol{e}\|_2$$

Proof. Direct calculation shows the following

$$\begin{aligned}\|\nabla p_{\boldsymbol{f}}(\boldsymbol{x})\|_{\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{w})^{-1}} &= \|-\boldsymbol{A}_x^T(c_e\boldsymbol{1} + \boldsymbol{f} + \boldsymbol{\psi}_P(\boldsymbol{x})) + \lambda\boldsymbol{x}\|_{\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{w})^{-1}} && \text{(Formula for } \nabla p_{\boldsymbol{f}}(\boldsymbol{x})\text{)} \\ &\leq \|\nabla p_{\boldsymbol{e}}(\boldsymbol{x})\|_{\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{w})^{-1}} + \|\boldsymbol{A}_x^T(\boldsymbol{f} - \boldsymbol{e})\|_{\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{w})^{-1}} && \text{(Triangle inequality)} \\ &\leq \|\nabla p_{\boldsymbol{e}}(\boldsymbol{x})\|_{\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{w})^{-1}} + \frac{1}{\sqrt{c_e + \mu(\boldsymbol{x})}}\|\boldsymbol{A}_x^T(\boldsymbol{f} - \boldsymbol{e})\|_{(\boldsymbol{A}_x^T\boldsymbol{A}_x)^{-1}} && \text{(Bound on } \boldsymbol{Q}(\boldsymbol{x},\boldsymbol{w})\text{)} \\ &\leq \|\nabla p_{\boldsymbol{e}}(\boldsymbol{x})\|_{\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{w})^{-1}} + \frac{1}{\sqrt{c_e + \mu(\boldsymbol{x})}}\|\boldsymbol{f} - \boldsymbol{e}\|_2 && \text{(Property of projection matrix)}\end{aligned}$$

where in the second to third line we used $\boldsymbol{Q}(\boldsymbol{x},\boldsymbol{w}) \succeq \boldsymbol{H}(\boldsymbol{x}) \succeq (c_e + \mu(\boldsymbol{x}))\boldsymbol{A}_x^T\boldsymbol{A}_x$. □

We now have everything we need to analyze our centering algorithm.

Lemma 8.3.9. *Let $\boldsymbol{x}^{(0)} \in P = \{\boldsymbol{y} \in \mathbb{R}^n : \boldsymbol{A}\boldsymbol{y} \geq \boldsymbol{b}\}$ and let $\boldsymbol{\tau}^{(0)} \in \mathbb{R}^m$ such that $\|\boldsymbol{e}(\boldsymbol{\tau}^{(0)}, \boldsymbol{x}^{(0)})\|_\infty \leq \frac{1}{3}c_e \leq \frac{1}{3}$. Assume that r is a positive integer, $0 \leq c_\Delta \leq 0.01c_e$ and $\delta_{\boldsymbol{e}^{(0)}}(\boldsymbol{x}^{(0)}) \leq \frac{1}{100}\sqrt{c_e + \mu(\boldsymbol{x}^{(0)})}$. With high probability in n, the algorithm `Centering`$(\boldsymbol{x}^{(0)}, \boldsymbol{\tau}^{(0)}, r, c_\Delta)$ outputs $(\boldsymbol{x}^{(r)}, \boldsymbol{\tau}^{(r)})$ such that*

1. $\delta_{\boldsymbol{e}^{(r)}}(\boldsymbol{x}^{(r)}) \leq 2\left(1 - \frac{1}{64}\right)^r \delta_{\boldsymbol{e}^{(0)}}(\boldsymbol{x}^{(0)})$.
2. $\mathbf{E}[p_{\boldsymbol{e}^{(k)}}(\boldsymbol{x}^{(r)})] \geq p_{\boldsymbol{e}^{(0)}}(\boldsymbol{x}^{(0)}) - 8\left(\delta_{\boldsymbol{e}^{(0)}}(\boldsymbol{x}^{(0)})\right)^2$.

Algorithm 21: $(\boldsymbol{x}^{(r)}, \boldsymbol{\tau}^{(r)}) = \texttt{Centering}(\boldsymbol{x}^{(0)}, \boldsymbol{\tau}^{(0)}, r, c_\Delta)$

Input: Initial point $\boldsymbol{x}^{(0)} \in P = \{\boldsymbol{y} \in \mathbb{R}^n : \boldsymbol{A}\boldsymbol{y} \geq \boldsymbol{b}\}$, Estimator of leverage scores $\boldsymbol{\tau}^{(0)} \in \mathbb{R}^n$
Input: Number of iterations $r > 0$, Accuracy of the estimator $0 \leq c_\Delta \leq 0.01 c_e$.
Given: $\|\boldsymbol{e}^{(0)}\|_\infty \leq \frac{1}{3} c_e \leq \frac{1}{3}$ where $\boldsymbol{e}^{(0)} = \boldsymbol{e}(\boldsymbol{\tau}^{(0)}, \boldsymbol{x}^{(0)})$.
Given: $\delta_{\boldsymbol{e}^{(0)}}(\boldsymbol{x}^{(0)}) = \|\nabla p_{\boldsymbol{e}^{(0)}}(\boldsymbol{x}^{(0)})\|_{\boldsymbol{H}(\boldsymbol{x}^{(0)})^{-1}} \leq \frac{1}{100}\sqrt{c_e + \mu(\boldsymbol{x}^{(0)})}$.
Compute $\boldsymbol{w}$ such that $\boldsymbol{\Psi}(\boldsymbol{x}^{(0)}) \preceq \boldsymbol{W} \preceq \frac{4}{3}\boldsymbol{\Psi}(\boldsymbol{x}^{(0)})$ (Similar to Lemma 2.3.4)
Let $\boldsymbol{Q} \stackrel{\text{def}}{=} \boldsymbol{Q}(\boldsymbol{x}^{(0)}, \boldsymbol{w})$.
for $k = 1$ *to* r **do**
  $\boldsymbol{x}^{(k)} := \boldsymbol{x}^{(k-1)} - \frac{1}{8}\boldsymbol{Q}^{-1}\nabla p_{\boldsymbol{e}^{(k-1)}}(\boldsymbol{x}^{(k-1)})$.
  Sample $\boldsymbol{\Delta}^{(k)} \in \mathbb{R}^n$ s.t.
    $\mathbf{E}[\boldsymbol{\Delta}^{(k)}] = \boldsymbol{\psi}(\boldsymbol{x}^{(k)}) - \boldsymbol{\psi}(\boldsymbol{x}^{(k-1)})$ and
    with high probability in n,
   $\|\boldsymbol{\Delta}^{(k)} - (\boldsymbol{\psi}(\boldsymbol{x}^{(k)}) - \boldsymbol{\psi}(\boldsymbol{x}^{(k-1)}))\|_2 \leq c_\Delta \|\boldsymbol{S}^{-1}_{\boldsymbol{x}^{(k-1)}}(\boldsymbol{s}_{\boldsymbol{x}^{(k)}} - \boldsymbol{s}_{\boldsymbol{x}^{(k-1)}})\|_2$ (See Section 8.4.1)
  $\boldsymbol{\tau}^{(k)} := \boldsymbol{\tau}^{(k-1)} + \boldsymbol{\Delta}^{(k)}$.
  $\boldsymbol{e}^{(k)} := \boldsymbol{e}(\boldsymbol{\tau}^{(k)}, \boldsymbol{x}^{(k)})$.
end
Output: $(\boldsymbol{x}^{(r)}, \boldsymbol{\tau}^{(r)})$

3. $\mathbf{E}\boldsymbol{e}^{(r)} = \boldsymbol{e}^{(0)}$ *and* $\|\boldsymbol{e}^{(r)} - \boldsymbol{e}^{(0)}\|_2 \leq \frac{1}{10} c_\Delta$.

4. $\left\|\boldsymbol{S}^{-1}_{\boldsymbol{x}^{(0)}}(\boldsymbol{s}(\boldsymbol{x}^{(r)}) - \boldsymbol{s}(\boldsymbol{x}^{(0)}))\right\|_2 \leq \frac{1}{10}$.

where $\boldsymbol{e}^{(r)} = \boldsymbol{e}(\boldsymbol{\tau}^{(r)}, \boldsymbol{x}^{(r)})$.

Proof. Let $\eta = \|\nabla p_{\boldsymbol{e}^{(0)}}(\boldsymbol{x}^{(0)})\|_{\boldsymbol{Q}^{-1}}$. First, we use induction to prove that $\|\boldsymbol{x}^{(r)} - \boldsymbol{x}^{(0)}\|_{\boldsymbol{Q}} \leq 8\eta$, $\|\nabla p_{\boldsymbol{e}^{(r)}}(\boldsymbol{x}^{(r)})\|_{\boldsymbol{Q}^{-1}} \leq \left(1 - \frac{1}{64}\right)^r \eta$ and $\|\boldsymbol{e}^{(r)} - \boldsymbol{e}^{(0)}\|_2 \leq \frac{1}{10} c_\Delta$ for all r.

Clearly the claims hold for $r = 0$. We now suppose they hold for all $r \leq t$ and show that they hold for $r = t + 1$. Now, since $\|\boldsymbol{x}^{(t)} - \boldsymbol{x}^{(0)}\|_{\boldsymbol{Q}} \leq 8\eta$, $\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} - \frac{1}{8}\boldsymbol{Q}^{-1}\nabla p_{\boldsymbol{e}^{(t)}}(\boldsymbol{x}^{(t)})$, and $\|\nabla p_{\boldsymbol{e}^{(t)}}(\boldsymbol{x}^{(t)})\|_{\boldsymbol{Q}^{-1}} \leq \left(1 - \frac{1}{64}\right)^t \eta \leq \eta$, we have

$$\|\boldsymbol{x}^{(t+1)} - \boldsymbol{x}^{(0)}\|_{\boldsymbol{Q}} \leq \|\boldsymbol{x}^{(t)} - \boldsymbol{x}^{(0)}\|_{\boldsymbol{Q}} + \frac{1}{8}\|\nabla p_{\boldsymbol{e}^{(t)}}(\boldsymbol{x}^{(t)})\|_{\boldsymbol{Q}^{-1}} \leq 9\eta.$$

We will improve this estimate later in the proof to finish the induction on $\|\boldsymbol{x}^{(t+1)} - \boldsymbol{x}^{(0)}\|_{\boldsymbol{Q}}$, but using this, $\eta \leq 0.01\sqrt{c_e + \mu(\boldsymbol{x}^{(0)})}$, and $\|\boldsymbol{e}^{(t)}\|_\infty \leq \|\boldsymbol{e}^{(t)} - \boldsymbol{e}^{(0)}\|_\infty + \|\boldsymbol{e}^{(0)}\|_\infty \leq \frac{c_e}{2}$, we can invoke Lemma 8.3.6 and Lemma 8.3.7 and therefore

$$\|\nabla p_{\boldsymbol{e}^{(t)}}(\boldsymbol{x}^{(t+1)})\|_{\boldsymbol{Q}^{-1}} \leq \left(1 - \frac{1}{32}\right)\|\nabla p_{\boldsymbol{e}^{(t)}}(\boldsymbol{x}^{(t)})\|_{\boldsymbol{Q}^{-1}}.$$

By Lemma 8.3.8 we have

$$\|\nabla p_{\boldsymbol{e}^{(t+1)}}(\boldsymbol{x}^{(t+1)})\|_{\boldsymbol{Q}^{-1}} \leq \left(1 - \frac{1}{32}\right)\|\nabla p_{\boldsymbol{e}^{(t)}}(\boldsymbol{x}^{(t)})\|_{\boldsymbol{Q}^{-1}} + \frac{1}{\sqrt{c_e + \mu(\boldsymbol{x}^{(0)})}}\|\boldsymbol{e}^{(t+1)} - \boldsymbol{e}^{(t)}\|_2. \tag{8.4}$$

To bound $\|\boldsymbol{e}^{(t+1)} - \boldsymbol{e}^{(t)}\|_2$, we use the definition of $\boldsymbol{\Delta}$ to shows that

$$\begin{aligned}\|\boldsymbol{e}^{(t+1)} - \boldsymbol{e}^{(t)}\|_2 &= \left\|\left(\boldsymbol{\tau}^{(t+1)} - \boldsymbol{\psi}(\boldsymbol{x}^{(t+1)})\right) - \left(\boldsymbol{\tau}^{(t)} - \boldsymbol{\psi}(\boldsymbol{x}^{(t)})\right)\right\|_2 \\ &= \left\|\boldsymbol{\Delta}^{(t+1)} - \left(\boldsymbol{\psi}(\boldsymbol{x}^{(t+1)}) - \boldsymbol{\psi}(\boldsymbol{x}^{(t)})\right)\right\|_2 \\ &\leq c_\Delta \left\|\boldsymbol{S}_{x^{(t)}}^{-1}(\boldsymbol{s}_{x^{(t+1)}} - \boldsymbol{s}_{x^{(t)}})\right\|_2\end{aligned}$$

with high probability in n. Now, we note that Lemma 8.3.5 and the induction hypothesis $\|\boldsymbol{x}^{(t)} - \boldsymbol{x}^{(0)}\|_{\boldsymbol{H}(\boldsymbol{x}^{(0)})} \leq \|\boldsymbol{x}^{(t)} - \boldsymbol{x}^{(0)}\|_{\boldsymbol{Q}} \leq 8\eta$ shows that $(1-0.1)\boldsymbol{S}_{x^{(0)}} \preceq \boldsymbol{S}_{x^{(t)}} \preceq (1+0.1)\boldsymbol{S}_{x^{(0)}}$ and therefore

$$\begin{aligned}\|\boldsymbol{e}^{(t+1)} - \boldsymbol{e}^{(t)}\|_2 &\leq c_\Delta \|\boldsymbol{S}_{x^{(t)}}^{-1}(\boldsymbol{s}_{x^{(t)}} - \boldsymbol{s}_{x^{(t+1)}})\|_2 \\ &\leq \frac{c_\Delta}{1-0.1}\|\boldsymbol{S}_{x^{(0)}}^{-1}\boldsymbol{A}\left(\boldsymbol{x}^{(t)} - \boldsymbol{x}^{(t+1)}\right)\|_2 \\ &= \frac{c_\Delta}{1-0.1}\left\|\frac{1}{8}\boldsymbol{Q}^{-1}\nabla p_{\boldsymbol{e}^{(t)}}(\boldsymbol{x}^{(t)})\right\|_{\boldsymbol{A}^T\boldsymbol{S}_{x^{(0)}}^{-2}\boldsymbol{A}} \\ &\leq \frac{c_\Delta}{8\,(1-0.1)\sqrt{c_e + \mu(\boldsymbol{x}^{(0)})}}\|\nabla p_{\boldsymbol{e}^{(t)}}(\boldsymbol{x}^{(t)})\|_{\boldsymbol{Q}^{-1}} \qquad (8.5)\end{aligned}$$

where in the last line we used $\min_{i\in[m]} w_i \geq \mu(\boldsymbol{x}^{(0)})$. Since $c_\Delta < 0.01 c_e \leq 0.01$, by (8.4), we have

$$\begin{aligned}\|\nabla p_{\boldsymbol{e}^{(t+1)}}(\boldsymbol{x}^{(t+1)})\|_{\boldsymbol{Q}^{-1}} &\leq \left(1 - \frac{1}{32}\right)\|\nabla p_{\boldsymbol{e}^{(t)}}(\boldsymbol{x}^{(t)})\|_{\boldsymbol{Q}^{-1}} + \frac{0.01}{8\,(1-0.1)}\|\nabla p_{\boldsymbol{e}^{(t)}}(\boldsymbol{x}^{(t)})\|_{\boldsymbol{Q}^{-1}} \\ &\leq \left(1 - \frac{1}{64}\right)\|\nabla p_{\boldsymbol{e}^{(t)}}(\boldsymbol{x}^{(t)})\|_{\boldsymbol{Q}^{-1}} \quad .\end{aligned}$$

Furthermore, this implies that

$$\|\boldsymbol{x}^{(t+1)} - \boldsymbol{x}^{(0)}\|_{\boldsymbol{Q}} \leq \left\|\sum_{k=0}^{t}\frac{1}{8}\boldsymbol{Q}^{-1}\nabla p_{\boldsymbol{e}^{(k)}}(\boldsymbol{x}^{(k)})\right\|_{\boldsymbol{Q}^{-1}} \leq \frac{1}{8}\sum_{i=0}^{\infty}\left(1 - \frac{1}{64}\right)^k \eta \leq \frac{64}{8}\eta = 8\eta \quad .$$

Similarly, we have that

$$\begin{aligned}\|\boldsymbol{e}^{(t+1)} - \boldsymbol{e}^{(0)}\|_2 &\leq \sum_{k=0}^{t}\frac{c_\Delta}{8\,(1-0.1)\sqrt{c_e + \mu(\boldsymbol{x}^{(0)})}}\left(1 - \frac{1}{64}\right)^k \|\nabla p_{\boldsymbol{e}^{(0)}}(\boldsymbol{x}^{(0)})\|_{\boldsymbol{Q}^{-1}} \\ &\leq \frac{8c_\Delta\eta}{(1-0.1)\sqrt{c_e + \mu(\boldsymbol{x}^{(0)})}} \leq \frac{8c_\Delta}{(1-0.1)\sqrt{c_e + \mu(\boldsymbol{x}^{(0)})}}\delta_{\boldsymbol{e}^{(0)}}(\boldsymbol{x}^{(0)}) \leq \frac{1}{10}c_\Delta\end{aligned}$$

where we used $\eta = \|\nabla p_{\boldsymbol{e}^{(0)}}(\boldsymbol{x}^{(0)})\|_{\boldsymbol{Q}^{-1}} \leq \|\nabla p_{\boldsymbol{e}^{(0)}}(\boldsymbol{x}^{(0)})\|_{\boldsymbol{H}^{-1}} = \delta_{\boldsymbol{e}^{(0)}}(\boldsymbol{x}^{(0)})$ and this finishes the induction on $\|\nabla p_{\boldsymbol{e}^{(t)}}(\boldsymbol{x}^{(t)})\|_{\boldsymbol{Q}^{-1}}$, $\|\boldsymbol{x}^{(t)} - \boldsymbol{x}^{(0)}\|_{\boldsymbol{Q}}$ and $\|\boldsymbol{e}^{(t)} - \boldsymbol{e}^{(0)}\|_2$.

Hence, for all r, Lemma 8.3.6 shows that

$$\begin{aligned}\delta_{\boldsymbol{e}^{(r)}}(\boldsymbol{x}^{(r)}) &= \|\nabla p_{\boldsymbol{e}^{(r)}}(\boldsymbol{x}^{(r)})\|_{\boldsymbol{H}(\boldsymbol{x}^{(r)})^{-1}} \leq \sqrt{2}\|\nabla p_{\boldsymbol{e}^{(r)}}(\boldsymbol{x}^{(r)})\|_{\boldsymbol{H}(\boldsymbol{x}^{(0)})^{-1}} \\ &\leq \sqrt{\frac{8}{3}}\|\nabla p_{\boldsymbol{e}^{(r)}}(\boldsymbol{x}^{(r)})\|_{\boldsymbol{Q}^{-1}} \leq \sqrt{\frac{8}{3}}\left(1 - \frac{1}{64}\right)^r \|\nabla p_{\boldsymbol{e}^{(0)}}(\boldsymbol{x}^{(0)})\|_{\boldsymbol{Q}^{-1}} \\ &\leq 2\left(1 - \frac{1}{64}\right)^r \delta_{\boldsymbol{e}^{(0)}}(\boldsymbol{x}^{(0)}).\end{aligned}$$

Using that $\mathbf{E}\boldsymbol{e}^{(t+1)} = \boldsymbol{e}^{(t)}$, we see that the expected change in function value is only due to the change

while taking centering steps and therefore Lemma 8.3.7 shows that

$$\mathbf{E}[p_{\boldsymbol{e}^{(r)}}(\boldsymbol{x}^{(r)})] \geq p_{\boldsymbol{e}^{(0)}}(\boldsymbol{x}^{(0)}) - \frac{1}{8}\sum_{k=0}^{\infty}\left(1-\frac{1}{64}\right)^{2k}\|\nabla p_{\boldsymbol{e}^{(0)}}(\boldsymbol{x}^{(0)})\|^2_{\boldsymbol{Q}^{-1}} \geq p_{\boldsymbol{e}^{(0)}}(\boldsymbol{x}^{(0)}) - 8\left(\delta_{\boldsymbol{e}^{(0)}}(\boldsymbol{x}^{(0)})\right)^2.$$

Finally, for (4), we note that

$$\left\|\frac{s(\boldsymbol{x}^{(r)})-s(\boldsymbol{x}^{(0)})}{s(\boldsymbol{x}^{(0)})}\right\|_2 = \left\|\boldsymbol{x}^{(r)}-\boldsymbol{x}^{(0)}\right\|_{\boldsymbol{A}^T\boldsymbol{S}^{-2}_{x^{(0)}}\boldsymbol{A}} \leq \frac{1}{\sqrt{\mu(\boldsymbol{x}^{(0)})+c_e}}\left\|\boldsymbol{x}^{(r)}-\boldsymbol{x}^{(0)}\right\|_{\boldsymbol{Q}^{-1}} \leq \frac{1}{10}.$$

□

8.3.2 Changing Constraints

Here we bound the effect that adding or a removing a constraint has on the hybrid barrier function. Much of the analysis in this section follows from the following lemma which follows easily from the Sherman Morrison Formula.

Lemma 8.3.10 (Sherman Morrison Formula Implications)**.** *Let $\boldsymbol{B} \in \mathbb{R}^{n\times n}$ be an invertible symmetric matrix and let $\boldsymbol{a} \in \mathbb{R}^n$ be arbitrary vector satisfying $\boldsymbol{a}^T\boldsymbol{B}^{-1}\boldsymbol{a} < 1$. The following hold:*

1. $\left(\boldsymbol{B} \pm \boldsymbol{a}\boldsymbol{a}^T\right)^{-1} = \boldsymbol{B}^{-1} \mp \frac{\boldsymbol{B}^{-1}\boldsymbol{a}\boldsymbol{a}^T\boldsymbol{B}^{-1}}{1\pm\boldsymbol{a}^T\boldsymbol{B}^{-1}\boldsymbol{a}}$.
2. $\mathbf{0} \preceq \frac{\boldsymbol{B}^{-1}\boldsymbol{a}\boldsymbol{a}^T\boldsymbol{B}^{-1}}{1\pm\boldsymbol{a}^T\boldsymbol{B}^{-1}\boldsymbol{a}} \preceq \frac{\boldsymbol{a}^T\boldsymbol{B}^{-1}\boldsymbol{a}}{1\pm\boldsymbol{a}^T\boldsymbol{B}^{-1}\boldsymbol{a}}\boldsymbol{B}^{-1}$.
3. $\log\det\left(\boldsymbol{B}\pm\boldsymbol{a}\boldsymbol{a}^T\right) = \ln\det\boldsymbol{B} + \ln\left(1\pm\boldsymbol{a}^T\boldsymbol{B}^{-1}\boldsymbol{a}\right)$.

Proof. (1) follows immediately from Sherman Morrison Formula (Lemma 2.3.7). (2) follows since $\boldsymbol{a}\boldsymbol{a}^T$ is PSD,

$$\frac{\boldsymbol{B}^{-1}\boldsymbol{a}\boldsymbol{a}^T\boldsymbol{B}^{-1}}{1\pm\boldsymbol{a}^T\boldsymbol{B}^{-1}\boldsymbol{a}} = \boldsymbol{B}^{-1/2}\left[\frac{\boldsymbol{B}^{-1/2}\boldsymbol{a}\boldsymbol{a}^T\boldsymbol{B}^{-1/2}}{1\pm\boldsymbol{a}^T\boldsymbol{B}^{-1}\boldsymbol{a}}\right]\boldsymbol{B}^{-1/2} \quad ,$$

and $\boldsymbol{y}\boldsymbol{y}^T \preceq \|\boldsymbol{y}\|_2^2\boldsymbol{I}$ for any vector $\boldsymbol{y}$. (3) follows immediately from the Matrix Determinant Lemma. □

We also make use of the following technical helper lemma.

Lemma 8.3.11. *For $\boldsymbol{A} \in \mathbb{R}^{n\times m}$ and all $\boldsymbol{a} \in \mathbb{R}^n$ we have*

$$\sum_{i\in[m]}\frac{1}{\psi_{\boldsymbol{A}}[i]}\left(\boldsymbol{A}\left(\boldsymbol{A}^T\boldsymbol{A}+\lambda\boldsymbol{I}\right)^{-1}\boldsymbol{a}\right)_i^4 \leq \left(\boldsymbol{a}^T\left(\boldsymbol{A}^T\boldsymbol{A}+\lambda\boldsymbol{I}\right)^{-1}\boldsymbol{a}\right)^2$$

Proof. We have by Cauchy Schwarz that

$$\left(\mathbf{1}_i^T\boldsymbol{A}\left(\boldsymbol{A}^T\boldsymbol{A}+\lambda\boldsymbol{I}\right)^{-1}\boldsymbol{a}\right)^2 \leq \psi_{\boldsymbol{A}}[i]\cdot\boldsymbol{a}^T\left(\boldsymbol{A}^T\boldsymbol{A}+\lambda\boldsymbol{I}\right)^{-1}\boldsymbol{a}$$

and consequently

$$\sum_{i\in[m]}\frac{\left(\mathbf{1}_i^T\boldsymbol{A}\left(\boldsymbol{A}^T\boldsymbol{A}+\lambda\boldsymbol{I}\right)^{-1}\boldsymbol{a}\right)^4}{\psi_{\boldsymbol{A}}[i]} \leq \left(\boldsymbol{a}^T\left(\boldsymbol{A}^T\boldsymbol{A}+\lambda\boldsymbol{I}\right)^{-1}\boldsymbol{a}\right)\sum_{i\in[m]}\left(\mathbf{1}_i\boldsymbol{A}\left(\boldsymbol{A}^T\boldsymbol{A}+\lambda\boldsymbol{I}\right)^{-1}\boldsymbol{a}\right)^2.$$

Since

$$\sum_{i\in[m]}\left(\mathbf{1}_i^T\boldsymbol{A}\left(\boldsymbol{A}^T\boldsymbol{A}+\lambda\boldsymbol{I}\right)^{-1}\boldsymbol{a}\right)^2=\boldsymbol{a}^T\left(\boldsymbol{A}^T\boldsymbol{A}+\lambda\boldsymbol{I}\right)^{-1}\boldsymbol{A}^T\boldsymbol{A}\left(\boldsymbol{A}^T\boldsymbol{A}+\lambda\boldsymbol{I}\right)^{-1}\boldsymbol{a}$$
$$\leq\boldsymbol{a}^T\left(\boldsymbol{A}^T\boldsymbol{A}+\lambda\boldsymbol{I}\right)^{-1}\boldsymbol{a},$$

we have the desired result. □

We now bound the effect of adding a constraint.

Lemma 8.3.12. *Let $\boldsymbol{A}\in\mathbb{R}^{m\times n}$, $\boldsymbol{b}\in\mathbb{R}^m$, $\boldsymbol{\tau}\in\mathbb{R}^m$, and $\boldsymbol{x}\in P\stackrel{\text{def}}{=}\{\boldsymbol{y}\in\mathbb{R}^n:\boldsymbol{A}\boldsymbol{y}\geq\boldsymbol{b}\}$. Let $\overline{\boldsymbol{A}}\in\mathbb{R}^{(m+1)\times n}$ be $\boldsymbol{A}$ with a row $\boldsymbol{a}_{m+1}$ added, let $\overline{b}\in\mathbb{R}^{m+1}$ be the vector $\boldsymbol{b}$ with an entry b_{m+1} added, and let $\overline{P}\stackrel{\text{def}}{=}\left\{\boldsymbol{y}\in\mathbb{R}^n:\overline{\boldsymbol{A}}\boldsymbol{y}\geq\overline{b}\right\}$. Let $s_{m+1}=\boldsymbol{a}_{m+1}^T\boldsymbol{x}-b_{m+1}>0$, $\psi_a=\frac{\boldsymbol{a}_{m+1}^T(\boldsymbol{A}_x^T\boldsymbol{A}_x+\lambda\boldsymbol{I})^{-1}\boldsymbol{a}_{m+1}}{s_{m+1}^2}$.*

Now, let $\boldsymbol{\upsilon}\in\mathbb{R}^{m+1}$ be defined so that $\upsilon_{m+1}=\frac{\psi_a}{1+\psi_a}$ and for all $i\in[m]$

$$\upsilon_i=\tau_i-\frac{1}{1+\psi_a}\left[\boldsymbol{A}_x\left(\boldsymbol{A}_x^T\boldsymbol{A}_x+\lambda\boldsymbol{I}\right)^{-1}\frac{\boldsymbol{a}_{m+1}}{s_{m+1}}\right]_i^2.$$

Then, the following hold

- [Leverage Score Estimation] $e_{\overline{P}}(\vec{v},\boldsymbol{x})_{m+1}=0$ and $e_{\overline{P}}(\vec{v},\boldsymbol{x})_i=e_P(\boldsymbol{\tau},\boldsymbol{x})_i$ for all $i\in[m]$.
- [Function Value Increase] $p_{\boldsymbol{e}_{\overline{P}}(\boldsymbol{\upsilon},\boldsymbol{x})}(\boldsymbol{x})=p_{\boldsymbol{e}_P(\boldsymbol{\tau},\boldsymbol{x})}(\boldsymbol{x})-c_e\ln\left(s(\boldsymbol{x})_{m+1}/R\right)+\ln(1+\psi_a)$.
- [Centrality Increase] $\delta_{\boldsymbol{e}_{\overline{P}}(\boldsymbol{\upsilon},\boldsymbol{x})}(\boldsymbol{x})\leq\sqrt{1+\psi_a}\delta_{\boldsymbol{e}_P(\boldsymbol{\tau},\boldsymbol{x})}(\boldsymbol{x})+\left(c_e\sqrt{1+\psi_a}+\psi_a\right)\sqrt{\frac{\psi_a}{\mu(\boldsymbol{x})}}+\psi_a$.

Proof. By (1) in Lemma 8.3.10, we have that for all $i\in[m]$

$$\psi_{\overline{P}}(\boldsymbol{x})_i=\psi_P(\boldsymbol{x})_i-\frac{1}{1+\psi_a}\left[\boldsymbol{A}_x\left(\boldsymbol{A}_x^T\boldsymbol{A}_x+\lambda\boldsymbol{I}\right)^{-1}\frac{\boldsymbol{a}_{m+1}}{s_{m+1}}\right]_i^2$$

and that

$$\psi_{\overline{P}}(\boldsymbol{x})_{m+1}=\psi_a-\frac{\psi_a^2}{1+\psi_a}=\frac{\psi_a}{1+\psi_a}.$$

Consequently [Leverage Score Estimation] holds.

By (3) in Lemma 8.3.10 we have that [Function Value Change] holds.

To bound the change in centrality, we first note that

$$\begin{aligned}\psi_{\overline{P}}(\boldsymbol{x})_i&\geq\psi_P(\boldsymbol{x})_i-\frac{1}{1+\psi_a}\left[\frac{\boldsymbol{a}_i^T}{s_i}\left(\boldsymbol{A}_x^T\boldsymbol{A}_x+\lambda\boldsymbol{I}\right)^{-1}\frac{\boldsymbol{a}_i}{s_i}\right]\left[\frac{\boldsymbol{a}_{m+1}^T}{s_{m+1}}\left(\boldsymbol{A}_x^T\boldsymbol{A}_x+\lambda\boldsymbol{I}\right)^{-1}\frac{\boldsymbol{a}_{m+1}}{s_{m+1}}\right]\\&=\frac{\psi_P(\boldsymbol{x})_i}{1+\psi_a}.\end{aligned}$$

Therefore, we have that the approximate Hessian for $\overline{P}$, denoted $\overline{\boldsymbol{H}}(\boldsymbol{x})$, is bounded by $\overline{\boldsymbol{H}}^{-1}\preceq(1+\psi_a)\boldsymbol{H}^{-1}$.

To bound the change in centrality note that by (2) in Lemma 8.3.10 we have that $\overline{\boldsymbol{H}}^{-1}\preceq\boldsymbol{H}^{-1}$. Therefore if let $\boldsymbol{v}'\in\mathbb{R}^m$ be defined so that $\boldsymbol{v}'_i=\boldsymbol{v}_i$ for all $i\in[m]$ then by triangle inequality we have

$$\begin{aligned}\delta_{e_{\overline{p}}(\boldsymbol{v},\boldsymbol{x})}(\boldsymbol{x}) &= \|\overline{\boldsymbol{A}}_x^T(c_e\mathbf{1}+\boldsymbol{v})\|_{\overline{\boldsymbol{H}}^{-1}} \leq \sqrt{1+\psi_a}\|\overline{\boldsymbol{A}}_x^T(c_e\mathbf{1}+\boldsymbol{v})\|_{\boldsymbol{H}^{-1}} \\ &\leq \sqrt{1+\psi_a}\left(\|\boldsymbol{A}_x^T(c_e\mathbf{1}+\boldsymbol{\tau})\|_{\boldsymbol{H}^{-1}} + \left\|\frac{\boldsymbol{a}_{m+1}}{s_{m+1}}(c_e+\upsilon_{m+1})\right\|_{\boldsymbol{H}^{-1}} + \|\boldsymbol{A}_x^T(\boldsymbol{v}'-\boldsymbol{\tau})\|_{\boldsymbol{H}^{-1}}\right) \\ &= \sqrt{1+\psi_a}\left(\delta_{\boldsymbol{e}_P(\boldsymbol{\tau},\boldsymbol{x})}(\boldsymbol{x}) + \left(c_e+\frac{\psi_a}{1+\psi_a}\right)\left\|\frac{\boldsymbol{a}_{m+1}}{s_{m+1}}\right\|_{\boldsymbol{H}^{-1}} + \|\boldsymbol{A}_x^T(\boldsymbol{v}'-\boldsymbol{\tau})\|_{\boldsymbol{H}^{-1}}\right)\end{aligned}$$

Now, since $\boldsymbol{H}^{-1} \preceq \frac{1}{\mu(\boldsymbol{x})}\left(\boldsymbol{A}_x^T\boldsymbol{A}_x+\lambda\boldsymbol{I}\right)^{-1}$, we have that

$$\left\|\frac{\boldsymbol{a}_{m+1}}{s_{m+1}}\right\|_{\boldsymbol{H}^{-1}} \leq \frac{1}{\sqrt{\mu(\boldsymbol{x})}}\left\|\frac{\boldsymbol{a}_{m+1}}{s_{m+1}}\right\|_{\left(\boldsymbol{A}_x^T\boldsymbol{A}_x+\lambda\boldsymbol{I}\right)^{-1}} = \sqrt{\frac{\psi_a}{\mu(\boldsymbol{x})}}.$$

Since $\boldsymbol{\Psi}^{1/2}\boldsymbol{A}_x\left(\boldsymbol{A}_x^T\boldsymbol{\Psi}\boldsymbol{A}_x\right)^{-1}\boldsymbol{A}_x^T\boldsymbol{\Psi}^{1/2}$ is a projection matrix, we have $\boldsymbol{\Psi}^{-1} \succeq \boldsymbol{A}_x\left(\boldsymbol{A}_x^T\boldsymbol{\Psi}\boldsymbol{A}_x\right)^{-1}\boldsymbol{A}_x^T \succeq \boldsymbol{A}_x\boldsymbol{H}^{-1}\boldsymbol{A}_x^T$. By Lemma 8.3.11, we have

$$\begin{aligned}\|\boldsymbol{A}_x^T\left(\boldsymbol{\tau}'-\boldsymbol{v}\right)\|_{\boldsymbol{H}^{-1}}^2 &\leq \|\boldsymbol{\tau}'-\boldsymbol{v}\|_{\boldsymbol{\Psi}^{-1}}^2 \\ &= \sum_{i\in[m]}\frac{1}{\psi(\boldsymbol{x})_i}\left(\frac{1}{1+\psi_a}\left(\mathbf{1}_i\boldsymbol{A}_x\left(\boldsymbol{A}_x^T\boldsymbol{A}_x+\lambda\boldsymbol{I}\right)^{-1}\frac{\boldsymbol{a}_{m+1}}{s_{m+1}}\right)^2\right)^2 \\ &\leq \left(\frac{1}{1+\psi_a}\right)^2\left(\frac{\boldsymbol{a}_{m+1}^T\left(\boldsymbol{A}_x^T\boldsymbol{A}_x+\lambda\boldsymbol{I}\right)^{-1}\boldsymbol{a}_{m+1}}{s_{m+1}^2}\right)^2 = \left(\frac{\psi_a}{1+\psi_a}\right)^2\end{aligned}$$

Combining, we have that

$$\begin{aligned}\delta_{e_{\overline{P}}(\boldsymbol{v},\boldsymbol{x})}(\boldsymbol{x}) &\leq \sqrt{1+\psi_a}\left(\delta_{\boldsymbol{e}_P(\boldsymbol{\tau},\boldsymbol{x})}(\boldsymbol{x}) + \left(c_e+\frac{\psi_a}{1+\psi_a}\right)\sqrt{\frac{\psi_a}{\mu(\boldsymbol{x})}} + \frac{\psi_a}{1+\psi_a}\right) \\ &\leq \sqrt{1+\psi_a}\delta_{\boldsymbol{e}_P(\boldsymbol{\tau},\boldsymbol{x})}(\boldsymbol{x}) + \left(c_e\sqrt{1+\psi_a}+\psi_a\right)\sqrt{\frac{\psi_a}{\mu(\boldsymbol{x})}} + \psi_a.\end{aligned}$$

□

We now bound the effect of removing a constraint.

Lemma 8.3.13 (Removing a Constraint). *Let $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, $\boldsymbol{b} \in \mathbb{R}^m$, $\boldsymbol{\tau} \in \mathbb{R}^m$, and $\boldsymbol{x} \in P \stackrel{\text{def}}{=} \{\boldsymbol{y} \in \mathbb{R}^n : \boldsymbol{A}\boldsymbol{y} \geq \boldsymbol{b}\}$. Let $\overline{\boldsymbol{A}} \in \mathbb{R}^{(m-1)\times n}$ be $\boldsymbol{A}$ with row m removed, let $\overline{b} \in \mathbb{R}^{m-1}$ denote the first $m-1$ coordinates of $\boldsymbol{b}$, and let $\overline{P} \stackrel{\text{def}}{=} \{\boldsymbol{y} \in \mathbb{R}^n : \overline{\boldsymbol{A}}\boldsymbol{y} \geq \overline{b}\}$. Let $\psi_d = \psi_P(\boldsymbol{x})_m$.*

Now, let $\boldsymbol{v} \in \mathbb{R}^{m-1}$ be defined so that for all $i \in [m-1]$

$$\upsilon_i = \tau_i + \frac{1}{1-\psi_d}\left(\boldsymbol{A}_x\left(\boldsymbol{A}_x^T\boldsymbol{A}_x+\lambda\boldsymbol{I}\right)^{-1}\boldsymbol{A}_x^T\mathbf{1}_m\right)_i^2.$$

Assume $\psi_d \leq 1.1\mu(\boldsymbol{x}) \leq \frac{1}{10}$ and $\|\boldsymbol{e}_P(\boldsymbol{\tau},\boldsymbol{x})\|_\infty \leq c_e \leq \frac{1}{2}$, we have the following:

- [Leverage Score Estimation] $e_{\overline{P}}(\vec{v},\boldsymbol{x})_i = e_P(\boldsymbol{\tau},\boldsymbol{x})_i$ for all $i \in [m-1]$.
- [Function Value Decrease] $p_{\boldsymbol{e}_{\overline{p}}(\boldsymbol{v},\boldsymbol{x})}(\boldsymbol{x}) = p_{\boldsymbol{e}_P(\boldsymbol{\tau},\boldsymbol{x})}(\boldsymbol{x}) + [c_e + e_P(\boldsymbol{\tau},\boldsymbol{x})_m]\ln\left(s(\boldsymbol{x})_m/R\right) + \ln(1-\psi_d)$
- [Centrality Increase] $\delta_{\boldsymbol{e}_{\overline{p}}(\boldsymbol{v},\boldsymbol{x})}(\boldsymbol{x}) \leq \frac{1}{\sqrt{1-2\mu(\boldsymbol{x})}}\delta_{\boldsymbol{e}_P(\boldsymbol{\tau},\boldsymbol{x})}(\boldsymbol{x}) + 3(c_e+\mu(\boldsymbol{x}))$.

Proof. By (1) in Lemma 8.3.10, we have that for all $i \in [m-1]$

$$\psi_{\overline{P}}(\boldsymbol{x})_i = \psi_P(\boldsymbol{x})_i + \frac{1}{1-\psi_d}\left(\mathbf{1}_i^T \boldsymbol{A}_x \left(\boldsymbol{A}_x^T \boldsymbol{A}_x + \lambda \boldsymbol{I}\right)^{-1} \boldsymbol{A}_x^T \mathbf{1}_m\right)^2.$$

Consequently, [Leverage Score Estimation] holds. Furthermore, by (3) in Lemma 8.3.10, this then implies that [Function Value Change] holds.

To bound the change in centrality, we first note that by (1) and (2) in Lemma 8.3.10 and the fact $\psi_{\overline{P}}(\boldsymbol{x})_i \geq \psi_P(\boldsymbol{x})_i$, we have that the approximate Hessian for $\overline{P}$, denoted $\overline{\boldsymbol{H}}(\boldsymbol{x})$, is bounded by

$$\begin{aligned}\overline{\boldsymbol{H}}(\boldsymbol{x})^{-1} &\preceq \left(\boldsymbol{H}(\boldsymbol{x}) - \boldsymbol{A}_x^T \left(c_e \boldsymbol{I} + \boldsymbol{\Psi}_x\right)^{1/2} \mathbf{1}_m \mathbf{1}_m^T \left(c_e \boldsymbol{I} + \boldsymbol{\Psi}_x\right)^{1/2} \boldsymbol{A}_x\right)^{-1} \\ &\preceq \left(1 + \frac{\alpha}{1-\alpha}\right) \boldsymbol{H}(\boldsymbol{x})^{-1} = \left(\frac{1}{1-\alpha}\right) \boldsymbol{H}(\boldsymbol{x})^{-1}\end{aligned}$$

where $\alpha \stackrel{\text{def}}{=} \mathbf{1}_m^T \left(c_e \boldsymbol{I} + \boldsymbol{\Psi}_x\right)^{1/2} \boldsymbol{A}_x \boldsymbol{H}(\boldsymbol{x})^{-1} \boldsymbol{A}_x^T \left(c_e \boldsymbol{I} + \boldsymbol{\Psi}_x\right)^{1/2} \mathbf{1}_m$. Using $c_e + \mu(\boldsymbol{x}) \leq \frac{1}{2} + \frac{1}{10} \leq 1$, we have

$$\boldsymbol{H}(\boldsymbol{x})^{-1} \preceq \left(\boldsymbol{A}_x^T (c_e + \mu(\boldsymbol{x})) \boldsymbol{A}_x + \lambda \boldsymbol{I}\right)^{-1} \preceq (c_e + \mu(\boldsymbol{x}))^{-1} \left(\boldsymbol{A}_x^T \boldsymbol{A}_x + \lambda \boldsymbol{I}\right)^{-1}. \tag{8.6}$$

Using this, we have

$$\alpha \leq \left(\frac{c_e + \psi_d}{c_e + \mu(\boldsymbol{x})}\right) \mathbf{1}_m^T \boldsymbol{A}_x \left(\boldsymbol{A}_x^T \boldsymbol{A}_x + \lambda \boldsymbol{I}\right)^{-1} \boldsymbol{A}_x^T \mathbf{1}_m = \left(\frac{c_e + \psi_d}{c_e + \mu(\boldsymbol{x})}\right) \psi_d \,. \tag{8.7}$$

Now let $\boldsymbol{\tau}' \in \mathbb{R}^{m-1}$ be defined so that $\tau_i' = \tau_i$ for all $i \in [m-1]$. We have by above that

$$\delta_{\boldsymbol{e}_{\overline{p}}(\boldsymbol{v},\boldsymbol{x})}(\boldsymbol{x}) = \|\overline{\boldsymbol{A}}_x^T (c_e \mathbf{1} + \boldsymbol{v})\|_{\overline{\boldsymbol{H}}^{-1}} \leq \frac{1}{\sqrt{1-\alpha}} \|\overline{\boldsymbol{A}}_x^T (c_e \mathbf{1} + \boldsymbol{v})\|_{\boldsymbol{H}^{-1}}$$

and therefore, by triangle inequality

$$\begin{aligned}\|\overline{\boldsymbol{A}}_x^T (c_e \mathbf{1} + \boldsymbol{v})\|_{\boldsymbol{H}^{-1}} &\leq \|\boldsymbol{A}_x^T (c_e \mathbf{1} + \boldsymbol{\tau})\|_{\boldsymbol{H}^{-1}} + \|\boldsymbol{A}_x^T \mathbf{1}_m (c_e + \tau_m)\|_{\boldsymbol{H}^{-1}} + \|\boldsymbol{A}_x^T (\boldsymbol{\tau}' - \boldsymbol{v})\|_{\boldsymbol{H}^{-1}} \\ &= \delta_{\boldsymbol{e}_P(\boldsymbol{\tau},\boldsymbol{x})}(\boldsymbol{x}) + (c_e + \tau_m) \|\boldsymbol{A}_x^T \mathbf{1}_m\|_{\boldsymbol{H}^{-1}} + \|\boldsymbol{A}_x^T (\boldsymbol{\tau}' - \boldsymbol{v})\|_{\boldsymbol{H}^{-1}}.\end{aligned}$$

Now, (8.6) shows that

$$\|\boldsymbol{A}_x^T \mathbf{1}_m\|_{\boldsymbol{H}^{-1}} \leq \frac{1}{\sqrt{c_e + \mu(\boldsymbol{x})}} \|\boldsymbol{A}_x^T \mathbf{1}_m\|_{\left(\boldsymbol{A}_x^T \boldsymbol{A}_x + \lambda \boldsymbol{I}\right)^{-1}} \leq \sqrt{\frac{\psi_d}{c_e + \mu(\boldsymbol{x})}}$$

Furthermore, since $\boldsymbol{\Psi}^{-1} \succeq \boldsymbol{A}_x \left(\boldsymbol{A}_x^T \boldsymbol{\Psi} \boldsymbol{A}_x\right)^{-1} \boldsymbol{A}_x^T \succeq \boldsymbol{A}_x \boldsymbol{H}^{-1} \boldsymbol{A}_x^T$, by Lemma 8.3.11 we have

$$\begin{aligned}\|\boldsymbol{A}_x^T \left(\boldsymbol{\tau}' - \boldsymbol{v}\right)\|_{\boldsymbol{H}^{-1}}^2 &\leq \|\boldsymbol{\tau}' - \boldsymbol{v}\|_{\boldsymbol{\Psi}^{-1}}^2 \\ &= \sum_{i \in [m]} \frac{1}{\psi(\boldsymbol{x})_i} \left(\frac{1}{1-\psi_d} \left(\mathbf{1}_i^T \boldsymbol{A}_x \left(\boldsymbol{A}_x^T \boldsymbol{A}_x + \lambda \boldsymbol{I}\right)^{-1} \boldsymbol{A}_x^T \mathbf{1}_m\right)^2\right)^2 \\ &\leq \left(\frac{1}{1-\psi_d}\right)^2 \left(\mathbf{1}_m^T \boldsymbol{A}_x \left(\boldsymbol{A}_x^T \boldsymbol{A}_x + \lambda \boldsymbol{I}\right)^{-1} \boldsymbol{A}_x^T \mathbf{1}_m\right)^2 = \left(\frac{\psi_d}{1-\psi_d}\right)^2\end{aligned}$$

Combining, we have that

$$\delta_{\boldsymbol{e}_{\overline{p}}(\boldsymbol{v},\boldsymbol{x})}(\boldsymbol{x}) \leq \frac{1}{\sqrt{1-\alpha}} \left[\delta_{\boldsymbol{e}_P(\boldsymbol{\tau},\boldsymbol{x})}(\boldsymbol{x}) + (c_e + \tau_m) \sqrt{\frac{\psi_d}{c_e + \mu(\boldsymbol{x})}} + \frac{\psi_d}{1-\psi_d}\right].$$

Using the assumption $\psi_d \le 1.1\mu(\boldsymbol{x}) \le \frac{1}{10}$, $\|\boldsymbol{e}_P(\boldsymbol{\tau},\boldsymbol{x})\|_\infty \le c_e$ and (8.7), we have $\alpha \le 1.1\psi_d \le 1.21\mu(\boldsymbol{x})$ and $\tau_m \le \psi_d + c_e$, and

$$\begin{aligned}\delta_{\boldsymbol{e}_{\overline{P}}(\boldsymbol{v},\boldsymbol{x})}(\boldsymbol{x}) &\le \frac{1}{\sqrt{1-1.3\mu(\boldsymbol{x})}}\left[\delta_{\boldsymbol{e}_P(\boldsymbol{\tau},\boldsymbol{x})}(\boldsymbol{x}) + (c_e+\tau_m)\sqrt{1.1} + 1.2\psi_d\right] \\ &\le \frac{1}{\sqrt{1-2\mu(\boldsymbol{x})}}\delta_{\boldsymbol{e}_P(\boldsymbol{\tau},\boldsymbol{x})}(\boldsymbol{x}) + \frac{1}{\sqrt{1-\frac{1.3}{10}}}\left(\sqrt{1.1}\cdot 2c_e + 1.1(\sqrt{1.1}+1.2)\mu(\boldsymbol{x})\right)\end{aligned}$$

□

8.3.3 Hybrid Center Properties

Here we prove properties of points near the hybrid center. First we bound the distance between points in the $\boldsymbol{H}(\boldsymbol{x})$ norm in terms of the ℓ_2 norm of the points.

Lemma 8.3.14. *For $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ and $\boldsymbol{b} \in \mathbb{R}^m$ suppose that $\boldsymbol{x} \in P = \{\boldsymbol{y} : \boldsymbol{A}\boldsymbol{y} \ge \boldsymbol{b}\}$ and $\boldsymbol{e} \in \mathbb{R}^m$ such that $\|\boldsymbol{e}\|_\infty \le \frac{1}{2}c_e < \frac{1}{20}$ and $\delta_{\boldsymbol{e}} \le 0.1\sqrt{c_e+\mu(\boldsymbol{x})}$. Then for all $\boldsymbol{y} \in P$ we have*

$$\|\boldsymbol{x}-\boldsymbol{y}\|_{\boldsymbol{H}(\boldsymbol{x})} \le \frac{12mc_e + 6n + 2\lambda\|\boldsymbol{y}\|_2^2}{\sqrt{c_e+\mu(\boldsymbol{x})}} \tag{8.8}$$

and

$$\|\boldsymbol{x}\|_2^2 \le 4\lambda^{-1}(mc_e+n) + 2\|\boldsymbol{y}\|_2^2 \quad .$$

Proof. For notational simplicity let $\boldsymbol{t} \stackrel{\text{def}}{=} c_e\boldsymbol{1} + \boldsymbol{e} + \boldsymbol{\psi}_x$, $\boldsymbol{T} \stackrel{\text{def}}{=} \boldsymbol{Diag}(\boldsymbol{t})$, and $\boldsymbol{M} \stackrel{\text{def}}{=} \boldsymbol{A}_x^T(c_e\boldsymbol{I} + \boldsymbol{\Psi}_x)\boldsymbol{A}_x$. We have

$$\|\boldsymbol{x}-\boldsymbol{y}\|_{\boldsymbol{A}_x^T\boldsymbol{T}\boldsymbol{A}_x}^2 = \sum_{i\in[m]} t_i\frac{[\boldsymbol{s}_x-\boldsymbol{s}_y]_i^2}{[\boldsymbol{s}_x]_i^2} = \sum_{i\in[m]} t_i\left(1 - 2\frac{[\boldsymbol{s}_y]_i}{[\boldsymbol{s}_x]_i} + \frac{[\boldsymbol{s}_y]_i^2}{[\boldsymbol{s}_x]_i^2}\right) \tag{8.9}$$

and

$$\sum_{i\in[m]} t_i\frac{[\boldsymbol{s}_y]_i^2}{[\boldsymbol{s}_x]_i^2} \le \left(\sum_{i\in[m]} t_i\frac{[\boldsymbol{s}_y]_i}{[\boldsymbol{s}_x]_i}\right)\max_{i\in[m]}\frac{[\boldsymbol{s}_y]_i}{[\boldsymbol{s}_x]_i} \le \left(\sum_{i\in[m]} t_i\frac{[\boldsymbol{s}_y]_i}{[\boldsymbol{s}_x]_i}\right)\left(1 + \|\boldsymbol{S}_x^{-1}(\boldsymbol{s}_y-\boldsymbol{s}_x)\|_\infty\right) \tag{8.10}$$

and

$$\begin{aligned}\|\boldsymbol{S}_x^{-1}(\boldsymbol{s}_x-\boldsymbol{s}_y)\|_\infty = \max_{i\in[m]}\left|\boldsymbol{1}_i\boldsymbol{S}_x^{-1}\boldsymbol{A}(\boldsymbol{x}-\boldsymbol{y})\right| &\le \|\boldsymbol{x}-\boldsymbol{y}\|_{\boldsymbol{H}(\boldsymbol{x})}\sqrt{\max_{i\in[m]}\left[\boldsymbol{S}_x^{-1}\boldsymbol{A}\boldsymbol{H}(\boldsymbol{x})^{-1}\boldsymbol{A}^T\boldsymbol{S}_x^{-1}\right]_{ii}} \\ &\le (c_e+\mu(\boldsymbol{x}))^{-1/2}\|\boldsymbol{x}-\boldsymbol{y}\|_{\boldsymbol{H}(\boldsymbol{x})} \quad .\end{aligned} \tag{8.11}$$

Now, clearly $\sum_{i\in[m]} t_i[\boldsymbol{s}_y]_i/[\boldsymbol{s}_x]_i$ is positive and since $\|\boldsymbol{e}\|_\infty \le \frac{1}{2}c_e$ we know that $\frac{1}{2}\boldsymbol{M} \preceq \boldsymbol{A}_x^T\boldsymbol{T}\boldsymbol{A}_x$. Therefore, by combining, (8.9), (8.10), and (8.11) we have

$$\begin{aligned}\frac{1}{2}\|\boldsymbol{x}-\boldsymbol{y}\|_{\boldsymbol{M}}^2 &\le \|\boldsymbol{t}\|_1 - \sum_{i\in[m]} t_i\frac{[\boldsymbol{s}_y]_i}{[\boldsymbol{s}_x]_i} + \left(\sum_{i\in[m]} t_i\frac{[\boldsymbol{s}_y]_i}{[\boldsymbol{s}_x]_i}\right)\frac{\|\boldsymbol{x}-\boldsymbol{y}\|_{\boldsymbol{H}(\boldsymbol{x})}}{\sqrt{c_e+\mu(\boldsymbol{x})}} \\ &\le \|\boldsymbol{t}\|_1 + \left(\sum_{i\in[m]} t_i\frac{[\boldsymbol{s}_y]_i}{[\boldsymbol{s}_x]_i}\right)\frac{\|\boldsymbol{x}-\boldsymbol{y}\|_{\boldsymbol{H}(\boldsymbol{x})}}{\sqrt{c_e+\mu(\boldsymbol{x})}}\end{aligned} \tag{8.12}$$

Now since $\nabla p_{\boldsymbol{e}}(\boldsymbol{x}) = -\boldsymbol{A}^T \boldsymbol{S}_x^{-1} \boldsymbol{T}\mathbf{1} + \lambda \boldsymbol{x}$ we have

$$\langle \boldsymbol{x} - \boldsymbol{y}, \nabla p_e(\boldsymbol{x}) \rangle = -\sum_{i \in [m]} t_i \frac{[\boldsymbol{s}_x - \boldsymbol{s}_y]_i}{[\boldsymbol{s}_x]_i} + \lambda \boldsymbol{x}^T (\boldsymbol{x} - \boldsymbol{y})$$

and therefore by Cauchy Schwarz and $\boldsymbol{x}^T \boldsymbol{y} \leq \|\boldsymbol{x}\|_2^2 + \frac{1}{4}\|\boldsymbol{y}\|_2^2$,

$$\sum_{i \in [m]} t_i \frac{[\boldsymbol{s}_y]_i}{[\boldsymbol{s}_x]_i} = \|\boldsymbol{t}\|_1 - \lambda \|\boldsymbol{x}\|_2^2 + \lambda \boldsymbol{x}^T \boldsymbol{y} + \langle \boldsymbol{x} - \boldsymbol{y}, \nabla p_e(\boldsymbol{x}) \rangle \tag{8.13}$$

$$\leq \|t\|_1 + \frac{\lambda}{4}\|\boldsymbol{y}\|_2^2 + \|\boldsymbol{x} - \boldsymbol{y}\|_{\boldsymbol{H}(\boldsymbol{x})} \delta_e(\boldsymbol{x}) \quad . \tag{8.14}$$

Now, using (8.12), (8.14) and the definition of $\boldsymbol{H}(\boldsymbol{x})$, we have

$$\begin{aligned}
\frac{1}{2}\|\boldsymbol{x} - \boldsymbol{y}\|_{\boldsymbol{H}(\boldsymbol{x})}^2 &= \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{y}\|_M^2 + \frac{\lambda}{2}\|\boldsymbol{x} - \boldsymbol{y}\|_2^2 \\
&\leq \|\boldsymbol{t}\|_1 + \left(\sum_{i \in [m]} t_i \frac{[\boldsymbol{s}_y]_i}{[\boldsymbol{s}_x]_i} \right) \frac{\|\boldsymbol{x} - \boldsymbol{y}\|_{\boldsymbol{H}(\boldsymbol{x})}}{\sqrt{c_e + \mu(\boldsymbol{x})}} + \frac{\lambda}{2}\|\boldsymbol{x} - \boldsymbol{y}\|_2^2 \\
&\leq \|\boldsymbol{t}\|_1 + \frac{\|t\|_1 + \frac{\lambda}{4}\|\boldsymbol{y}\|_2^2}{\sqrt{c_e + \mu(\boldsymbol{x})}} \|\boldsymbol{x} - \boldsymbol{y}\|_{\boldsymbol{H}(\boldsymbol{x})} + \delta_e(\boldsymbol{x}) \frac{\|\boldsymbol{x} - \boldsymbol{y}\|_{\boldsymbol{H}(\boldsymbol{x})}^2}{\sqrt{c_e + \mu(\boldsymbol{x})}} + \frac{\lambda}{2}\|\boldsymbol{x} - \boldsymbol{y}\|_2^2.
\end{aligned}$$

Using the fact that $\delta_e(\boldsymbol{x}) \leq 0.1\sqrt{c_e + \mu(\boldsymbol{x})}$, we have

$$\frac{1}{4}\|\boldsymbol{x} - \boldsymbol{y}\|_{\boldsymbol{H}(\boldsymbol{x})}^2 \leq \|\boldsymbol{t}\|_1 + \frac{\lambda}{2}\|\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \frac{\|t\|_1 + \frac{\lambda}{4}\|\boldsymbol{y}\|_2^2}{\sqrt{c_e + \mu(\boldsymbol{x})}} \|\boldsymbol{x} - \boldsymbol{y}\|_{\boldsymbol{H}(\boldsymbol{x})}. \tag{8.15}$$

Furthermore, since $\sum_{i \in [m]} t_i [\boldsymbol{s}_y]_i / [\boldsymbol{s}_x]_i$ is positive, (8.13) shows that

$$\lambda \boldsymbol{x}^T (\boldsymbol{x} - \boldsymbol{y}) = \lambda \|\boldsymbol{x}\|_2^2 - \lambda \boldsymbol{x}^T \boldsymbol{y} \leq \|\boldsymbol{t}\|_1 + \langle \boldsymbol{x} - \boldsymbol{y}, \nabla p_e(\boldsymbol{x}) \rangle \leq \|\boldsymbol{t}\|_1 + \|\boldsymbol{x} - \boldsymbol{y}\|_{\boldsymbol{H}(\boldsymbol{x})} \delta_e(\boldsymbol{x})$$

and hence

$$\begin{aligned}
\frac{\lambda}{2}\|\boldsymbol{x} - \boldsymbol{y}\|_2^2 \leq \frac{\lambda}{2}\|\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \frac{\lambda}{2}\|\boldsymbol{x}\|_2^2 &= \lambda \boldsymbol{x}^T (\boldsymbol{x} - \boldsymbol{y}) + \frac{\lambda}{2}\|\boldsymbol{y}\|_2^2 \\
&\leq \|\boldsymbol{t}\|_1 + \frac{\lambda}{2}\|\boldsymbol{y}\|_2^2 + \|\boldsymbol{x} - \boldsymbol{y}\|_{\boldsymbol{H}(\boldsymbol{x})} \delta_e(\boldsymbol{x}) \quad .
\end{aligned} \tag{8.16}$$

Putting (8.16) into (8.15) and using the fact that $\delta_e(\boldsymbol{x}) \leq 0.1\sqrt{c_e + \mu(\boldsymbol{x})}$, we have

$$\frac{1}{4}\|\boldsymbol{x} - \boldsymbol{y}\|_{\boldsymbol{H}(\boldsymbol{x})}^2 \leq 2\|\boldsymbol{t}\|_1 + \frac{\lambda}{2}\|\boldsymbol{y}\|_2^2 + \left(0.1 + \frac{\|\boldsymbol{t}\|_1 + \frac{\lambda}{4}\|\boldsymbol{y}\|_2^2}{\sqrt{c_e + \mu(\boldsymbol{x})}} \right) \|\boldsymbol{x} - \boldsymbol{y}\|_{\boldsymbol{H}(\boldsymbol{x})}.$$

Now, using $\|\boldsymbol{t}\|_1 \leq 2mc_e + n$ and $\sqrt{c_e + \mu(\boldsymbol{x})} \leq 1.05$, we have

$$\frac{1}{4}\|\boldsymbol{x} - \boldsymbol{y}\|_{\boldsymbol{H}(\boldsymbol{x})}^2 \leq 2.2\alpha + (0.1 + \alpha) \|\boldsymbol{x} - \boldsymbol{y}\|_{\boldsymbol{H}(\boldsymbol{x})} \quad \text{for} \quad \alpha = \frac{2mc_e + n + \frac{\lambda}{4}\|\boldsymbol{y}\|_2^2}{\sqrt{c_e + \mu(\boldsymbol{x})}} \quad .$$

Since $\alpha \geq 1/\sqrt{1.1}$, we have that

$$\|\boldsymbol{x} - \boldsymbol{y}\|_{\boldsymbol{H}(\boldsymbol{x})} \leq \frac{0.1 + \alpha + \sqrt{(\alpha + 0.1)^2 + 2.2\alpha}}{2 \cdot \frac{1}{4}} \leq 6\alpha$$

yielding (8.8).

We also have by (8.13) and the fact that $\delta_e(\boldsymbol{x}) \leq 0.1\sqrt{c_e + \mu(\boldsymbol{x})}$,

$$\begin{aligned}\lambda\|\boldsymbol{x}\|_2^2 &= \|t\|_1 + \lambda \boldsymbol{x}^T\boldsymbol{y} + \langle \boldsymbol{x}-\boldsymbol{y}, \nabla p_{\boldsymbol{e}}(\boldsymbol{x})\rangle - \sum_{i\in[m]} t_i \frac{[\boldsymbol{s}_y]_i}{[\boldsymbol{s}_x]_i} \\ &\leq \|t\|_1 + \frac{\lambda}{2}\|\boldsymbol{x}\|_2^2 + \frac{\lambda}{2}\|\boldsymbol{y}\|_2^2 + \|\boldsymbol{x}-\boldsymbol{y}\|_{\boldsymbol{H}(\boldsymbol{x})}\delta_e(\boldsymbol{x}) \\ &\leq \|t\|_1 + \frac{\lambda}{2}\|\boldsymbol{x}\|_2^2 + \frac{\lambda}{2}\|\boldsymbol{y}\|_2^2 + 0.1\sqrt{c_e+\mu(\boldsymbol{x})}\|\boldsymbol{x}-\boldsymbol{y}\|_{\boldsymbol{H}(\boldsymbol{x})}\end{aligned}$$

Hence, using $\|\boldsymbol{t}\|_1 \leq 2mc_e + n$ and $\|\boldsymbol{x}-\boldsymbol{y}\|_{\boldsymbol{H}(\boldsymbol{x})} \leq 6\alpha$, we have

$$\begin{aligned}\frac{\lambda}{2}\|\boldsymbol{x}\|_2^2 &\leq \|t\|_1 + \frac{\lambda}{2}\|\boldsymbol{y}\|_2^2 + 0.6\left(2mc_e + n + \frac{\lambda}{4}\|\boldsymbol{y}\|_2^2\right) \\ &\leq \lambda\|\boldsymbol{y}\|_2^2 + 2(mc_e+n).\end{aligned}$$

□

In the following lemma we show how we can write one hyperplane in terms of the others provided that we are nearly centered and show there is a constraint that the central point is close to.

Lemma 8.3.15. *Let $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ and $\boldsymbol{b} \in \mathbb{R}^m$ such that $\|a_i\|_2 = 1$ for all i. Suppose that $\boldsymbol{x} \in P = \{\boldsymbol{y} : \boldsymbol{A}\boldsymbol{y} \geq \boldsymbol{b}\}$ and $\boldsymbol{e} \in \mathbb{R}^m$ such that $\|\boldsymbol{e}\|_\infty \leq \frac{1}{2}c_e \leq \frac{1}{2}$. Furthermore, let $\varepsilon = \min_{j\in[m]} s_j(\boldsymbol{x})$ and suppose that $i = \arg\min_{j\in[m]} s_j(\boldsymbol{x})$ then*

$$\left\|\boldsymbol{a}_i + \sum_{j\neq i}\left(\frac{s(x)_i}{s(x)_j}\right)\left(\frac{c_e + e_j + \psi_j(\boldsymbol{x})}{c_e + e_i + \psi_i(\boldsymbol{x})}\right)\boldsymbol{a}_j\right\|_2 \leq \frac{2\varepsilon}{(c_e+\mu(\boldsymbol{x}))}\left[\lambda\|\boldsymbol{x}\|_2 + \delta_e(\boldsymbol{x})\sqrt{\frac{mc_e+n}{\varepsilon^2}+\lambda}\right].$$

Proof. We know that

$$\begin{aligned}\nabla p_e(\boldsymbol{x}) &= -\boldsymbol{A}^T\boldsymbol{S}_x^{-1}(c_e\boldsymbol{1} + \boldsymbol{e} + \boldsymbol{\psi}_x) + \lambda\boldsymbol{x} \\ &= \lambda\boldsymbol{x} - \sum_{i\in[m]}\frac{(c_e + e_i + \psi_i)}{s(\boldsymbol{x})_i}\boldsymbol{a}_i\end{aligned}$$

Consequently, by $\|\boldsymbol{e}\|_\infty \leq \frac{1}{2}c_e$, and $\psi_i(\boldsymbol{x}) \geq \mu(\boldsymbol{x})$, we have

$$\begin{aligned}\left\|\boldsymbol{a}_i + \sum_{j\neq i}\left(\frac{s(x)_i}{s(x)_j}\right)\left(\frac{c_e + e_j + \psi_j(\boldsymbol{x})}{c_e + e_i + \psi_i(\boldsymbol{x})}\right)\boldsymbol{a}_j\right\|_2 &= \frac{s_i(\boldsymbol{x})}{c_e + e_i + \psi_i(\boldsymbol{x})}\left\|\boldsymbol{A}^T\boldsymbol{S}_x^{-1}(c_e\boldsymbol{1} + \boldsymbol{e} + \boldsymbol{\psi}_x)\right\|_2 \\ &\leq \frac{2\varepsilon}{c_e + \mu(\boldsymbol{x})}\left[\lambda\|\boldsymbol{x}\|_2 + \|\nabla p_e(\boldsymbol{x})\|_2\right].\end{aligned}$$

Using $\|\boldsymbol{a}_i\| = 1$, $\sum_i \psi_i \leq n$, and $s_i(\boldsymbol{x}) \geq \varepsilon$, we have

$$\begin{aligned}\mathrm{Tr}(\boldsymbol{A}_x^T(c_e\boldsymbol{I} + \boldsymbol{\Psi}_x)\boldsymbol{A}_x) &= \mathrm{Tr}(\boldsymbol{A}_x\boldsymbol{A}_x^T(c_e\boldsymbol{I} + \boldsymbol{\Psi}_x)) \\ &= \sum_i (c_e + \psi_i)\frac{\|a_i\|_2^2}{s_i^2(\boldsymbol{x})} \leq \frac{mc_e+n}{\varepsilon^2}. \qquad (8.17)\end{aligned}$$

Hence, we have $\boldsymbol{H}(\boldsymbol{x}) \preceq \left(\frac{mc_e+n}{\varepsilon^2} + \lambda\right)\boldsymbol{I}$ and $\|\nabla p_e(\boldsymbol{x})\|_2 \leq \delta_e(\boldsymbol{x})\sqrt{\frac{mc_e+n}{\varepsilon^2}+\lambda}$ yielding the result. □

8.3.4 The Algorithm

Here, we put all the results in the previous sections to get our ellipsoid algorithm. Below is a sketch of the pseudocode; we use c_a, c_d, c_e, c_Δ to denote parameters we decide later.

Algorithm 22: Our Cutting Plane Method

Input: $\boldsymbol{A}^{(0)} \in \mathbb{R}^{m\times n}$, $\boldsymbol{b}^{(0)} \in \mathbb{R}^m$, $\varepsilon > 0$, and radius $R > 0$.
Input: A separation oracle for a non-empty set $K \subset B_\infty(R)$.
Check: Throughout the algorithm, if $s_i(\boldsymbol{x}^{(k)}) < \varepsilon$ output $P^{(k)}$.
Check: Throughout the algorithm, if $\boldsymbol{x}^{(k)} \in K$, output $\boldsymbol{x}^{(k)}$.
Set $P^{(0)} = B_\infty(R)$.
Set $\boldsymbol{x}^{(0)} := \mathbf{0}$ and compute $\tau_i^{(0)} = \psi_{P^{(0)}}(\boldsymbol{x}^{(0)})_i$ for all $i \in [m]$ exactly.
for $k = 0$ *to* ∞ **do**
  Let $m^{(k)}$ be the number of constraints in $P^{(k)}$.
  Compute $\boldsymbol{w}^{(k)}$ such that $\boldsymbol{\Psi}_{P^{(k)}}(\boldsymbol{x}^{(k)}) \preceq \boldsymbol{W}^{(k)} \preceq (1+c_\Delta)\boldsymbol{\Psi}_{P^{(k)}}(\boldsymbol{x}^{(k)})$.
  Let $i^{(k)} \in \arg\max_{i\in[m^{(k)}]} \left|w_i^{(k)} - \tau_i^{(k)}\right|$.
  Set $\tau_{i^{(k)}}^{(k+\frac{1}{3})} = \psi_{P^{(k)}}(\boldsymbol{x}^{(k)})_{i^{(k)}}$ and $\tau_j^{(k+\frac{1}{3})} = \tau_j^{(k)}$ for all $j \neq i^{(k)}$.
  if $\min_{i\in[m^{(k)}]} w_i^{(k)} \leq c_d$ **then**
    Remove constraint with minimum $w_i^{(k)}$ yielding polytope $P^{(k+1)}$.
    Update $\boldsymbol{\tau}$ according to Lemma 8.3.13 to get $\tau_j^{(k+\frac{2}{3})}$.
  else
    Use separation oracle at $\boldsymbol{x}^{(k)}$ to get a constraint $\{\boldsymbol{x} : \boldsymbol{a}^T\boldsymbol{x} \geq \boldsymbol{a}^T\boldsymbol{x}^{(k)}\}$ with $\|\boldsymbol{a}\|_2 = 1$.
    Add constraint $\{\boldsymbol{x} : \boldsymbol{a}^T\boldsymbol{x} \geq \boldsymbol{a}^T\boldsymbol{x}^{(k)} - c_a^{-1/2}\sqrt{\boldsymbol{a}^T(\boldsymbol{A}^T\boldsymbol{S}_{\boldsymbol{x}^{(k)}}^{-2}\boldsymbol{A} + \lambda\boldsymbol{I})^{-1}\boldsymbol{a}}\}$ yielding polytope $P^{(k+1)}$.
    Update $\boldsymbol{\tau}$ according to Lemma 8.3.12 to get $\tau_j^{(k+\frac{2}{3})}$.
  $(\boldsymbol{x}^{(k+1)}, \boldsymbol{\tau}^{(k+1)}) = \texttt{Centering}(\boldsymbol{x}^{(k)}, \boldsymbol{\tau}^{(k+\frac{2}{3})}, 200, c_\Delta)$.
end

In the algorithm, there are two main invariants we maintain. First, we maintain that the centrality $\delta_{P,\boldsymbol{e}}(\boldsymbol{x})$, which indicates how close $\boldsymbol{x}$ is to the minimum point of $p_{\boldsymbol{e}}$, is small. Second, we maintain that $\|\boldsymbol{e}(\boldsymbol{\tau},\boldsymbol{x})\|_\infty$, which indicates how accurate the leverage score estimate $\boldsymbol{\tau}$ is, is small. In the following lemma we show that we maintain both invariants throughout the algorithm.

Lemma 8.3.16. *Assume that $c_e \leq c_d \leq \frac{1}{10^6}$, $c_a\sqrt{c_a} \leq \frac{c_d}{10^3}$, $c_d \leq c_a$, and $c_\Delta \leq Cc_e/\log(n\log(R/\varepsilon))$ for some small enough universal constant C. During our cutting plane method, for all k, with high probability in n, we have*

1. $\|\boldsymbol{e}(\boldsymbol{\tau}^{(k+\frac{1}{3})}, \boldsymbol{x}^{(k)})\|_\infty \leq \frac{1}{1000}c_e$, $\|\boldsymbol{e}(\boldsymbol{\tau}^{(k+\frac{2}{3})}, \boldsymbol{x}^{(k)})\|_\infty \leq \frac{1}{1000}c_e$, $\|\boldsymbol{e}(\boldsymbol{\tau}^{(k+1)}, \boldsymbol{x}^{(k+1)})\|_\infty \leq \frac{1}{400}c_e$.
2. $\delta_{P^{(k)},\boldsymbol{e}(\boldsymbol{\tau}^{(k+\frac{2}{3})},\boldsymbol{x}^{(k)})}(\boldsymbol{x}^{(k)}) \leq \frac{1}{100}\sqrt{c_e + \min\left(\mu(\boldsymbol{x}^{(k)}), c_d\right)}$.
3. $\delta_{P^{(k+1)},\boldsymbol{e}(\boldsymbol{\tau}^{(k+1)},\boldsymbol{x}^{(k+1)})}(\boldsymbol{x}^{(k+1)}) \leq \frac{1}{400}\sqrt{c_e + \min\left(\mu(\boldsymbol{x}^{(k+1)}), c_d\right)}$.

Proof. Some statements of the proof hold only with high probability in n; we omit mentioning this for simplicity.

We prove by induction on k. Note that the claims are written in order consistent with the algorithm and proving the statement for k involves bounding centrality at the point $\boldsymbol{x}^{(k+1)}$. Trivially we define, $\boldsymbol{\tau}^{(-1)} = \boldsymbol{\tau}^{(-\frac{2}{3})} = \boldsymbol{\tau}^{(-\frac{1}{3})} = \boldsymbol{\tau}^{(0)}$, $\boldsymbol{x}^{(-1)} = \boldsymbol{x}^{(0)}$ and note that the claims then hold for $k = -1$ as we compute the initial leverage scores, $\boldsymbol{\tau}^{(0)}$, exactly and since the polytope is symmetric we have $\delta_{\boldsymbol{e}(\boldsymbol{\tau}^{(0)},\boldsymbol{x}^{(0)})}(\mathbf{0}) = 0$. We now suppose they hold for all $t < k$ and show that they hold for $t = k$.

We first bound δ. For notational simplicity, let $\eta_t \stackrel{\text{def}}{=} \sqrt{c_e + \min(\mu(\boldsymbol{x}^{(t)}), c_d)}$. By the induction hypothesis we know that $\delta_{P^{(t)},\boldsymbol{e}(\boldsymbol{\tau}^{(t)},\boldsymbol{x}^{(t)})}(\boldsymbol{x}^{(t)}) \leq \frac{1}{400}\eta_t$. Now, when we update $\tau^{(t)}$ to $\tau^{(t+\frac{1}{3})}$, we set $\boldsymbol{e}_{i^{(t)}}$ to 0. Consequently, Lemma 8.3.8 and the induction hypothesis $\|\boldsymbol{e}(\boldsymbol{\tau}^{(t)}, \boldsymbol{x}^{(t)})\|_\infty \leq \frac{1}{400}c_e$ show that

$$\begin{aligned}\delta_{P^{(t)},\boldsymbol{e}(\boldsymbol{\tau}^{(t+\frac{1}{3})},\boldsymbol{x}^{(t)})}(\boldsymbol{x}^{(t)}) &\leq \delta_{P^{(t)},\boldsymbol{e}(\boldsymbol{\tau}^{(t)},\boldsymbol{x}^{(t)})}(\boldsymbol{x}^{(t)}) + \frac{1}{\sqrt{c_e + \mu(\boldsymbol{x}^{(t)})}} e_{i^{(t)}}(\boldsymbol{\tau}^{(t)}, \boldsymbol{x}^{(t)}) \\ &\leq \frac{1}{400}\eta_t + \frac{\sqrt{c_e}}{400} \leq \frac{\eta_t}{200} \end{aligned} \tag{8.18}$$

Next, we estimate the δ changes when we remove or add a constraint.

For the case of removal, we note that it happens only if $\mu(\boldsymbol{x}^{(t)}) \leq \min_i w_i \leq c_d \leq \frac{1}{10^6}$. Also, the row we remove has leverage score at most $1.1\mu(\boldsymbol{x}^{(t)})$ because we pick the row with minimum w. Hence, Lemma 8.3.13 show that

$$\begin{aligned}\delta_{P^{(t+1)},\boldsymbol{e}(\boldsymbol{\tau}^{(t+\frac{2}{3})},\boldsymbol{x}^{(t)})}(\boldsymbol{x}^{(t)}) &\leq \frac{1}{\sqrt{1 - 2\mu(\boldsymbol{x}^{(t)})}} \delta_{P^{(t)},\boldsymbol{e}(\boldsymbol{\tau}^{(t+\frac{1}{3})},\boldsymbol{x}^{(t)})}(\boldsymbol{x}^{(t)}) + 3(c_e + \mu(\boldsymbol{x}^{(t)})) \\ &\leq \frac{1}{\sqrt{1 - 2 \cdot 10^{-6}}} \left(\frac{\eta_t}{200}\right) + 3(c_e + \mu(\boldsymbol{x}^{(t)})) \leq \frac{\eta_t}{100}\end{aligned}$$

where we used the fact $\mu(\boldsymbol{x}^{(t)}) \leq c_d$ and hence $c_e + \mu(\boldsymbol{x}^{(t)}) \leq \sqrt{2c_d}\eta_t \leq \frac{\sqrt{2}}{1000}\eta_t$.

For the case of addition, we note that it happens only if $2\mu(\boldsymbol{x}^{(t)}) \geq \min_i w_i \geq c_d$. Furthermore, in this case the hyperplane we add is chosen precisely so that $\psi_a = c_a$. Furthermore, since $c_e \leq c_d \leq c_a \leq \frac{1}{100}$ by Lemma 8.3.12 we have that

$$\begin{aligned}\delta_{P^{(t+1)},\boldsymbol{e}(\boldsymbol{\tau}^{(t+\frac{2}{3})},\boldsymbol{x}^{(t)})}(\boldsymbol{x}^{(t)}) &\leq \sqrt{1 + c_a}\delta_{P^{(t)},\boldsymbol{e}(\boldsymbol{\tau}^{(t+\frac{1}{3})},\boldsymbol{x}^{(t)})} + \left(c_e\sqrt{1 + c_a} + c_a\right)\sqrt{\frac{c_a}{\mu(\boldsymbol{x}^{(t)})}} + c_a \\ &\leq \frac{\eta_t}{190} + 4c_a\sqrt{\frac{c_a}{c_d}} \quad .\end{aligned}$$

Furthermore, since $c_a\sqrt{c_a} \leq \frac{c_d}{1000}$, $\mu(\boldsymbol{x}^{(t)}) \geq c_d/2$, and $c_d \leq 10^{-6}$ we know that $4c_a\sqrt{c_a/c_d} \leq \frac{1}{250}\eta_t$ and consequently in both cases we have $\delta_{P^{(t+1)},\boldsymbol{e}(\boldsymbol{\tau}^{(t+\frac{2}{3})},\boldsymbol{x}^{(t)})}(\boldsymbol{x}^{(t)}) \leq \frac{1}{100}\eta_t$.

Now, note that Lemmas 8.3.12 and 8.3.13 show that $\boldsymbol{e}$ does not change during the addition or removal of an constraint. Hence, we have $\|\boldsymbol{e}(\boldsymbol{\tau}^{(t+\frac{2}{3})}, \boldsymbol{x}^{(t)})\|_\infty \leq \|\boldsymbol{e}(\boldsymbol{\tau}^{(t+\frac{1}{3})}, \boldsymbol{x}^{(t)})\|_\infty$. Furthermore, we know the step "$\boldsymbol{\tau}_{i^{(k)}}^{(k+\frac{1}{3})} = \psi_{P^{(k)}}(\boldsymbol{x}^{(k)})_{i^{(k)}}$" only decreases $\|\boldsymbol{e}\|_\infty$ and hence we have $\|\boldsymbol{e}(\boldsymbol{\tau}^{(t+\frac{2}{3})}, \boldsymbol{x}^{(t)})\|_\infty \leq \|\boldsymbol{e}(\boldsymbol{\tau}^{(t)}, \boldsymbol{x}^{(t)})\|_\infty \leq \frac{c_e}{400}$. Thus, we have all the conditions needed for Lemma 8.3.9 and consequently

$$\delta_{P^{(t+1)},\boldsymbol{e}(\boldsymbol{\tau}^{(t+1)},\boldsymbol{x}^{(t+1)})}(\boldsymbol{x}^{(t+1)}) \leq 2\left(1 - \frac{1}{64}\right)^{200} \delta_{P^{(t+1)},\boldsymbol{e}(\boldsymbol{\tau}^{(t+\frac{2}{3})},\boldsymbol{x}^{(t)})}(\boldsymbol{x}^{(t)}) \leq \frac{1}{1000}\eta_t \quad .$$

Lemma 8.3.9 also shows that that $\left\|\frac{s(\boldsymbol{x}^{(t+1)}) - s(\boldsymbol{x}^{(t)})}{s(\boldsymbol{x}^{(t)})}\right\|_2 \leq \frac{1}{10}$ and hence $\psi_i(\boldsymbol{x}^{(t)}) \leq 2\psi_i(\boldsymbol{x}^{(t+1)})$ for all i.

Therefore, $\eta_t \leq 2\eta_{t+1}$ and thus

$$\delta_{P^{(t+1)},\boldsymbol{e}(\boldsymbol{\tau}^{(t+1)},\boldsymbol{x}^{(t+1)})}(\boldsymbol{x}^{(t+1)}) \leq \frac{\sqrt{c_e + \min\left(c_d, \mu(\boldsymbol{x}^{(t+1)})\right)}}{400}.$$

completing the induction case for δ.

Now, we bound $\|\boldsymbol{e}\|_\infty$. Lemma 8.3.12 and 8.3.13 show that $\boldsymbol{e}$ does not change during the addition or removal of an constraint. Hence, $\boldsymbol{e}$ is affected by only the update step "$\tau_{i^{(k)}}^{(k+\frac{1}{2})} = \psi_{P^{(k)}}(\boldsymbol{x}^{(k)})_{i^{(k)}}$" and the centering step. Using the induction hypothesis $\delta_{P^{(t)},\boldsymbol{e}(\boldsymbol{\tau}^{(t+\frac{2}{3})},\boldsymbol{x}^{(t)})}(\boldsymbol{x}^{(t)}) \leq \frac{1}{100}\eta_t$ and Lemma 8.3.9 shows that $\mathbf{E}\boldsymbol{e}(\boldsymbol{\tau}^{(t+1)},\boldsymbol{x}^{(t+1)}) = \boldsymbol{e}(\boldsymbol{\tau}^{(t+\frac{2}{3})},\boldsymbol{x}^{(t)})$ and $\|\boldsymbol{e}(\boldsymbol{\tau}^{(t+1)},\boldsymbol{x}^{(t+1)}) - \boldsymbol{e}(\boldsymbol{\tau}^{(t+\frac{2}{3})},\boldsymbol{x}^{(t)})\|_2 \leq \frac{1}{10}c_\Delta$. The goal for the update step is to decrease $\boldsymbol{e}$ by updating $\boldsymbol{\tau}$. In Section 8.4.2, we give a self-contained analysis of the effect of this step as a game. In each round, the vector $\boldsymbol{e}$ is corrupted by some mean 0 and bounded variance noise and the problem is how to update $\boldsymbol{e}$ such that $\|\boldsymbol{e}\|_\infty$ is bounded. Theorem 8.4.3 shows that we can do this by setting the $\boldsymbol{e}_i = 0$ for the almost maximum coordinate in each iteration. This is exactly what the update step is doing. Since our algorithm would run only $O(n\log(nR/\varepsilon))$ many iterations, Theorem 8.4.3 shows that this strategy guarantees that after the update step, we have

$$\left\|\boldsymbol{e}(\tau^{(t+\frac{1}{3})},\boldsymbol{x}^{(t)})\right\|_\infty = O\left(c_\Delta \log\left(n\log(R/\varepsilon)\right)\right).$$

Now, by our choice of c_Δ, we have $\|\boldsymbol{e}(\boldsymbol{\tau}^{(t+\frac{1}{3})},\boldsymbol{x}^{(t)})\|_\infty \leq \frac{1}{1000}c_e$. Lemma 8.3.12 and 8.3.13 show that $\boldsymbol{e}$ does not change during the addition or removal of an constraint. Hence, we have $\|\boldsymbol{e}(\boldsymbol{\tau}^{(t+\frac{2}{3})},\boldsymbol{x}^{(t)})\|_\infty \leq \frac{1}{1000}c_e$. Now, we note that again Lemma 8.3.9 shows $\|\boldsymbol{e}(\boldsymbol{\tau}^{(t+1)},\boldsymbol{x}^{(t+1)}) - \boldsymbol{e}(\boldsymbol{\tau}^{(t+\frac{2}{3})},\boldsymbol{x}^{(t)})\|_2 \leq \frac{1}{10}c_\Delta \leq \frac{1}{1000}c_e$, and we have $\|\boldsymbol{e}(\boldsymbol{\tau}^{(t+1)},\boldsymbol{x}^{(t+1)})\|_\infty \leq \frac{c_e}{400}$. This finishes the induction case for $\|\boldsymbol{e}\|_\infty$ and proves this lemma. □

Next, we show the number of constraints is always linear to n.

Lemma 8.3.17. *Throughout our cutting plane method, there are at most* $1 + \frac{2n}{c_d}$ *constraints.*

Proof. We only add a constraint if $\min_i w_i \geq c_d$. Since $2\psi_i \geq w_i$, we have $\psi_i \geq \frac{c_d}{2}$ for all i. Letting m denote the number of constraints after we add that row, we have $n \geq \sum_i \psi_i \geq (m-1)(c_d/2)$. □

Using $K \neq \emptyset$ and $K \subset B_\infty(R)$, here we show that the points are bounded.

Lemma 8.3.18. *During our Cutting Plane Method, for all k, we have* $\|\boldsymbol{x}^{(k)}\|_2 \leq 6\sqrt{n/\lambda} + 2\sqrt{n}R$.

Proof. By Lemma 8.3.16 and Lemma 8.3.14 we know that $\|\boldsymbol{x}^{(k)}\|_2^2 \leq 4\lambda^{-1}(mc_e + n) + 2\|\boldsymbol{y}\|_2^2$ for any $\boldsymbol{y} \in P^{(k)}$. Since our method never cuts out any point in K and since K is nonempty, there is some $\boldsymbol{y} \in K \subset P^{(k)}$. Since $K \subset B_\infty(R)$, we have $\|\boldsymbol{y}\|_2^2 \leq nR^2$. Furthermore, by Lemma 8.3.17 we have that $mc_e \leq c_e + 2n \leq 3n$ yielding the result. □

Lemma 8.3.19. $s_i\left(\boldsymbol{x}^{(k)}\right) \leq 12\sqrt{n/\lambda} + 4\sqrt{n}R + \sqrt{\frac{1}{c_a\lambda}}$ *for all i and k in the our cutting plane method.*

Proof. Let $\boldsymbol{x}^{(j)}$ be the current point at the time that the constraint corresponding to s_i, denoted $\{\boldsymbol{x} : \boldsymbol{a}_i^T\boldsymbol{x} \geq a_i^T\boldsymbol{x}^{(j)} - s_i(\boldsymbol{x}^{(j)})\}$, was added. Clearly

$$s_i(\boldsymbol{x}^{(k)}) = \boldsymbol{a}_i^T\boldsymbol{x}^{(k)} - a_i^T\boldsymbol{x}^{(j)} + s_i(\boldsymbol{x}^{(j)}) \leq \|\boldsymbol{a}_i\| \cdot \|\boldsymbol{x}^{(k)}\| + \left|\boldsymbol{a}_i^T\boldsymbol{x}^{(j)} - s_i(\boldsymbol{x}^{(j)})\right| \quad .$$

On the one hand, if the constraint for s_i comes from the initial symmetric polytope $P^{(0)} = B_\infty(R)$, we know $\left|\boldsymbol{a}^T\boldsymbol{x}^{(j)} - \boldsymbol{s}_i(\boldsymbol{x}^{(j)})\right| \le R$. On the other hand, if the constraint was added later then we know that

$$s_i(\boldsymbol{x}^{(j)}) = c_a^{-1/2}\sqrt{\boldsymbol{a}^T(\boldsymbol{A}^T\boldsymbol{S}_{\boldsymbol{x}^{(j)}}^{-2}\boldsymbol{A} + \lambda\boldsymbol{I})^{-1}\boldsymbol{a}} \le (c_a\lambda)^{-1/2}$$

and $\left|\boldsymbol{a}^T\boldsymbol{x}^{(j)} - s_i(\boldsymbol{x}^{(j)})\right| \le \|\boldsymbol{a}_i\| \cdot \|\boldsymbol{x}^{(j)}\| + \left|s_i(\boldsymbol{x}^{(j)})\right|$. Since $\|\boldsymbol{a}_i\|_2 = 1$ by design and $\|\boldsymbol{x}^{(j)}\|_2$ and $\|\boldsymbol{x}^{(k)}\|_2$ are upper bounded by $6\sqrt{n/\lambda} + 2\sqrt{n}R$ by Lemma 8.3.18, in either case the result follows. □

Now, we have everything we need to prove that the potential function is increasing in expectation.

Lemma 8.3.20. *Under the assumptions of Lemma 8.3.16 if* $\lambda = \frac{1}{c_a R^2}$, $c_e = \frac{c_d}{4\ln(6nR/\varepsilon)}$, *and* $24c_d \le c_a \le \frac{1}{1000}$ *then for all* k *we have*

$$\mathbb{E}p_{\boldsymbol{e}(\boldsymbol{\tau}^{(k+1)},\boldsymbol{x}^{(k+1)})}(\boldsymbol{x}^{(k+1)}) \quad \ge \quad p_{\boldsymbol{e}(\boldsymbol{\tau}^{(k)},\boldsymbol{x}^{(k)})}(\boldsymbol{x}^{(k)}) - c_d + \ln(1+\beta)$$

where $\beta = c_a$ *for the case of adding a constraint and* $\beta = -c_d$ *for the case of removal.*

Proof. Note that there are three places which affect the function value, namely the update step for $\tau^{(k+\frac{1}{3})}$, the addition/removal of constraints, and the centering step. We bound the effect of each separately.

First, for the update step, we have

$$p_{\boldsymbol{e}(\boldsymbol{\tau}^{(k+\frac{1}{3})},\boldsymbol{x}^{(k)})}(\boldsymbol{x}^{(k)}) = (c + e_{i^{(k)}})\log(s_{i^{(k)}}(\boldsymbol{x}^{(k)})/R) + p_{\boldsymbol{e}(\boldsymbol{\tau}^{(k)},\boldsymbol{x}^{(k)})}(\boldsymbol{x}^{(k)}).$$

Lemma 8.3.19, the termination condition, $\lambda = \frac{1}{c_a R^2}$ and $c_a < \frac{1}{1000}$ ensure that

$$\varepsilon \le s_{i^{(k)}}(\boldsymbol{x}^{(k)}) \le 12\sqrt{n/\lambda} + 4\sqrt{n}R + \sqrt{\frac{1}{c_a\lambda}} \le 6\sqrt{n}R \tag{8.19}$$

and Lemma 8.3.16 shows that $|e_{i^{(k)}}| \le c_e$. Hence, we have

$$p_{\boldsymbol{e}(\boldsymbol{\tau}^{(k+\frac{1}{3})},\boldsymbol{x}^{(k)})}(\boldsymbol{x}^{(k)}) \ge p_{\boldsymbol{e}(\boldsymbol{\tau}^{(k)},\boldsymbol{x}^{(k)})}(\boldsymbol{x}^{(k)}) + 2c_e\log(\varepsilon/R).$$

For the addition step, Lemma 8.3.12 shows that

$$\begin{aligned} p_{\boldsymbol{e}(\boldsymbol{\tau}^{(k+\frac{2}{3})},\boldsymbol{x}^{(k)})}(\boldsymbol{x}^{(k)}) &= p_{\boldsymbol{e}(\boldsymbol{\tau}^{(k+\frac{1}{3})},\boldsymbol{x}^{(k)})}(\boldsymbol{x}^{(k)}) - c_e\ln\left(s(\boldsymbol{x})_{m+1}/R\right) + \ln(1+c_a) \\ &\ge p_{\boldsymbol{e}(\boldsymbol{\tau}^{(k+\frac{1}{3})},\boldsymbol{x}^{(k)})}(\boldsymbol{x}^{(k)}) - c_e\log(6\sqrt{n}) + \ln(1+c_a) \end{aligned}$$

and for the removal step, Lemma 8.3.13 and $|e_i| \le c_e$ shows that

$$\begin{aligned} p_{\boldsymbol{e}(\boldsymbol{\tau}^{(k+\frac{2}{3})},\boldsymbol{x}^{(k)})}(\boldsymbol{x}^{(k)}) &\ge p_{\boldsymbol{e}(\boldsymbol{\tau}^{(k+\frac{1}{3})},\boldsymbol{x}^{(k)})}(\boldsymbol{x}^{(k)}) + [c_e + e_P(\boldsymbol{\tau},\boldsymbol{x})_m]\ln\left(s(\boldsymbol{x})_m/R\right) + \ln(1-c_d) \\ &\ge p_{\boldsymbol{e}(\boldsymbol{\tau}^{(k+\frac{1}{3})},\boldsymbol{x}^{(k)})}(\boldsymbol{x}^{(k)}) - 2c_e\log(R/\varepsilon) + \ln(1-c_d) \end{aligned}$$

After the addition or removal of a constraint, Lemma 8.3.16 shows that

$$\delta_{P^{(k)},\boldsymbol{e}(\boldsymbol{\tau}^{(k+\frac{2}{3})},\boldsymbol{x}^{(k)})}(\boldsymbol{x}^{(k)}) \le \frac{1}{100}\sqrt{c_e + \min\left(\mu(\boldsymbol{x}^{(k)}), c_d\right)}$$

and therefore Lemma 8.3.9 and $c_e \leq c_d$ show that

$$\begin{aligned}\mathbb{E}p_{\boldsymbol{e}(\boldsymbol{\tau}^{(k+1)},\boldsymbol{x}^{(k+1)})}(\boldsymbol{x}^{(k+1)}) &\geq p_{\boldsymbol{e}(\boldsymbol{\tau}^{(k+\frac{2}{3})},\boldsymbol{x}^{(k)})}(\boldsymbol{x}^{(k)}) - 8\left(\frac{\sqrt{c_e + \min\left(\mu(\boldsymbol{x}^{(k)}), c_d\right)}}{100}\right)^2 \\ &\geq p_{\boldsymbol{e}(\boldsymbol{\tau}^{(k+\frac{2}{3})},\boldsymbol{x}^{(k)})}(\boldsymbol{x}^{(k)}) - \frac{c_d}{625}.\end{aligned}$$

Combining them with $c_e = \frac{c_d}{4\ln(6nR/\varepsilon)}$, we have

$$\begin{aligned}\mathbb{E}p_{\boldsymbol{e}(\boldsymbol{\tau}^{(k+1)},\boldsymbol{x}^{(k+1)})}(\boldsymbol{x}^{(k+1)}) &\geq p_{\boldsymbol{e}(\boldsymbol{\tau}^{(k)},\boldsymbol{x}^{(k)})}(\boldsymbol{x}^{(k)}) - 2c_e\log(6nR/\varepsilon) - \frac{c_d}{625} + \ln(1+\beta) \\ &\geq p_{\boldsymbol{e}(\boldsymbol{\tau}^{(k)},\boldsymbol{x}^{(k)})}(\boldsymbol{x}^{(k)}) - c_d + \ln(1+\beta)\end{aligned}$$

where $\beta = c_a$ for the case of addition and $\beta = -c_d$ for the case of removal. □

Theorem 8.3.21. *For $c_a = \frac{1}{10^{10}}$, $c_d = \frac{1}{10^{12}}$, $c_e = \frac{c_d}{4\ln(6nR/\varepsilon)}$, $c_\Delta = \frac{Cc_e}{\log(n\log(R/\varepsilon))}$ and $\lambda = \frac{1}{c_a R^2}$ for some small enough universal constant C, then we have*

$$\mathbb{E}p_{\boldsymbol{e}(\boldsymbol{\tau}^{(k+1)},\boldsymbol{x}^{(k+1)})}(\boldsymbol{x}^{(k+1)}) \geq p_{\boldsymbol{e}(\boldsymbol{\tau}^{(k)},\boldsymbol{x}^{(k)})}(\boldsymbol{x}^{(k)}) - \frac{1}{10^{11}} + \frac{9\beta}{10^{11}}$$

where $\beta = 1$ for the case of addition and $\beta = 0$ for the case of removal.

Proof. It is easy to see that these parameters satisfy the requirements of Lemma 8.3.20. □

8.3.5 Guarantees of the Algorithm

In this section we put everything together to prove Theorem 8.3.26, the main result of this section, providing the guarantees of our cutting plane method.

For the remainder of this section we assume that $c_a = \frac{1}{10^{10}}$, $c_d = \frac{1}{10^{12}}$, $c_e = \frac{c_d}{4\ln(6nR/\varepsilon)}$, $c_\Delta = \frac{Cc_e}{\log(n\ln(R/\varepsilon))}$ and $\lambda = \frac{1}{c_a R^2}$. Consequently, throughout the algorithm we have

$$\|\boldsymbol{x}\|_2 \leq 6\sqrt{n/\lambda} + 2\sqrt{n}R \leq 3\sqrt{n}R. \tag{8.20}$$

Lemma 8.3.22. *If $s_i(\boldsymbol{x}^{(k)}) < \varepsilon$ for some i and k during our Cutting Plane Method then*

$$\max_{\boldsymbol{y}\in P^{(k)}\cap B_\infty(R)} \langle \boldsymbol{a}_i, \boldsymbol{y}\rangle - \min_{\boldsymbol{y}\in P^{(k)}\cap B_\infty(R)} \langle \boldsymbol{a}_i, \boldsymbol{y}\rangle \leq \frac{8n\varepsilon}{c_a c_e}.$$

Proof. Let $\boldsymbol{y} \in P^{(k)} \cap B_\infty(R)$ be arbitrary. Since $\boldsymbol{y} \in B_\infty(R)$ clearly $\|\boldsymbol{y}\|_2^2 \leq nR^2$. Furthermore, by Lemma 8.3.17 and the choice of parameters $mc_e + n \leq 3n$. Consequently, by Lemma 8.3.14 and the fact that $\lambda = \frac{1}{c_a R^2}$ and $c_a < 1$ we have

$$\|\boldsymbol{x} - \boldsymbol{y}\|_{\boldsymbol{H}(\boldsymbol{x})} \leq \frac{12mc_e + 6n + 2\lambda\|\boldsymbol{y}\|_2^2}{\sqrt{c_e + \mu(\boldsymbol{x})}} \leq \frac{30n + 2\frac{n}{c_a}}{\sqrt{c_e + \mu(\boldsymbol{x})}} \leq \frac{4n}{c_a\sqrt{c_e}}$$

and therefore

$$\left\|\boldsymbol{S}_{x^{(k)}}^{-1}(s(\boldsymbol{x}^{(k)}) - s(\boldsymbol{y}))\right\|_\infty \leq \frac{1}{\sqrt{c_e}}\left\|\boldsymbol{S}_{x^{(k)}}^{-1}(s(\boldsymbol{x}^{(k)}) - s(\boldsymbol{y}))\right\|_{c_e\boldsymbol{I}+\boldsymbol{\Psi}} \leq \frac{4n}{c_a c_e}.$$

Consequently, we have $(1 - \frac{4n}{c_a c_e})s_i(\boldsymbol{x}^{(k)}) \leq s_i(\boldsymbol{y}) \leq (1 + \frac{4n}{c_a c_e})s_i(\boldsymbol{x}^{(k)})$ for all $\boldsymbol{y} \in P^{(k)} \cap B_\infty(R)$. □

Now let us show how to compute a proof (or certificate) that the feasible region has small width on the direction $\boldsymbol{a}_i$.

Lemma 8.3.23. *Suppose that during some iteration k for $i = \arg\min_j s_j(\boldsymbol{x}^{(k)})$ we have $s_i(\boldsymbol{x}^{(k)}) \leq \varepsilon$. Let $(\boldsymbol{x}_*, \boldsymbol{\tau}_*) = \texttt{Centering}(\boldsymbol{x}^{(k)}, \boldsymbol{\tau}^{(k)}, 64\log(2R/\varepsilon), c_\Delta)$ where $\boldsymbol{\tau}^{(k)}$ is the τ at that point in the algorithm and let*

$$\boldsymbol{a}^* = \sum_{j \neq i} t_j \boldsymbol{a}_j \text{ where } t_j = \left(\frac{s(\boldsymbol{x}_*)_i}{s(\boldsymbol{x}_*)_j}\right)\left(\frac{c_e + e_j(\boldsymbol{x}_*, \boldsymbol{\tau}_*) + \psi_j(\boldsymbol{x}_*)}{c_e + e_i(\boldsymbol{x}_*, \boldsymbol{\tau}_*) + \psi_i(\boldsymbol{x}_*)}\right).$$

Then, we have that $\|\boldsymbol{a}_i + \boldsymbol{a}^*\|_2 \leq \frac{8\sqrt{n}\varepsilon}{c_a c_e R}$ and $t_j \geq 0$ for all j. Furthermore, we have

$$\left(\sum_{j \neq i}^{O(n)} t_j a_j\right)^T \boldsymbol{x}_* - \sum_{j \neq i}^{O(n)} t_j b_j \leq \frac{4n\varepsilon}{c_e} \quad .$$

Proof. By Lemma 8.3.9 and Lemma 8.3.16 we know that $\boldsymbol{e}(\boldsymbol{x}_*, \boldsymbol{\tau}_*) \leq \frac{1}{2}c_e$ and $\delta_{\boldsymbol{e}(\boldsymbol{x}_*, \boldsymbol{\tau}_*)} \leq \frac{\varepsilon}{R}\sqrt{c_e + \mu(\boldsymbol{x}_*)}$. Since $\boldsymbol{e}(\boldsymbol{x}_*, \boldsymbol{\tau}_*) \leq \frac{1}{2}c_e$, we have $t_j \geq 0$ for all j. Furthermore, by Lemma 8.3.15 and (8.20), we then have that with high probability in n

$$\begin{aligned}
\|\boldsymbol{a}_i + \boldsymbol{a}^*\|_2 &\leq \frac{2\varepsilon}{(c_e + \mu(\boldsymbol{x}_*))}\left[\lambda\|\boldsymbol{x}_*\|_2 + \delta_e(\boldsymbol{x}_*)\sqrt{\frac{mc_e + n}{\varepsilon^2} + \lambda}\right] \\
&\leq \frac{2\varepsilon}{c_e}\left[\frac{1}{c_a R^2}(3\sqrt{n}R) + \frac{2\varepsilon}{R}\sqrt{\frac{3n}{\varepsilon^2} + \frac{1}{c_a R^2}}\right] \\
&\leq \frac{2\varepsilon}{c_e}\left[\frac{3\sqrt{n}}{c_a R} + \frac{2\sqrt{3n}}{R} + \frac{2}{\sqrt{c_a}R}\right] \leq \frac{8\sqrt{n}\varepsilon}{c_a c_e R}.
\end{aligned}$$

By Lemma 8.3.16 we know that $\boldsymbol{e}(\boldsymbol{x}_*, \boldsymbol{\tau}_*) \leq \frac{1}{2}c_e$ and hence

$$\begin{aligned}
\left(\sum_{j \neq i}^{O(n)} t_j a_j\right)^T \boldsymbol{x}_* - \sum_{j \neq i}^{O(n)} t_j b_j &= \sum_{j \neq i}^{O(n)} t_j s(\boldsymbol{x}_*)_j = s_i(\boldsymbol{x}_*) \sum_{j \neq i}^{O(n)} \left(\frac{c_e + e_j(\boldsymbol{x}_*, \boldsymbol{\tau}_*) + \psi_j(\boldsymbol{x}_*)}{c_e + e_i(\boldsymbol{x}_*, \boldsymbol{\tau}_*) + \psi_i(\boldsymbol{x}_*)}\right) \\
&\leq s_i(\boldsymbol{x}_*) \sum_{j \neq i}^{O(n)} \left(\frac{\frac{3}{2}c_e + \psi_j(\boldsymbol{x}_*)}{\frac{1}{2}c_e + \psi_i(\boldsymbol{x}_*)}\right) \leq s_i(\boldsymbol{x}_*)\frac{3mc_e + 2n}{c_e} \\
&\leq \frac{3n}{c_e} s_i(\boldsymbol{x}_*) \leq \frac{4n\varepsilon}{c_e}
\end{aligned}$$

where the last line follows the fact that $s_i(\boldsymbol{x}_*) \leq 1.1 s_i(\boldsymbol{x}^{(k)}) \leq 1.1\varepsilon$ (Lemma 8.3.9). □

Lemma 8.3.24. *During our Cutting Plane Method, if $p_{\boldsymbol{e}}(\boldsymbol{x}^{(k)}) \geq 2n\log(\frac{nR}{c_a\varepsilon}) + \frac{6n}{c_a}$, then we have $s_i(\boldsymbol{x}^{(k)}) \leq \varepsilon$ for some i.*

Proof. Recall that

$$p_{\boldsymbol{e}}(\boldsymbol{x}^{(k)}) = -\sum_{i \in [m]} (c_e + e_i)\log\left(s_i(\boldsymbol{x}^{(k)})_i / R\right) + \frac{1}{2}\log\det\left(R^2\left(\boldsymbol{A}^T \boldsymbol{S}_{x^{(k)}}^{-2}\boldsymbol{A} + \lambda \boldsymbol{I}\right)\right) + \frac{\lambda}{2}\|\boldsymbol{x}^{(k)}\|_2^2.$$

Using $\|\boldsymbol{x}^{(k)}\| \leq 3\sqrt{n}R$ (8.20) and $\lambda = \frac{1}{c_a R^2}$, we have

$$p_{\boldsymbol{e}}(\boldsymbol{x}^{(k)}) \leq -\sum_{i\in[m]} (c_e + e_i)\log(s(\boldsymbol{x}^{(k)})_i/R) + \frac{1}{2}\log\det\left(R^2\left(\boldsymbol{A}^T\boldsymbol{S}_{x^{(k)}}^{-2}\boldsymbol{A} + \lambda\boldsymbol{I}\right)\right) + \frac{5n}{c_a}.$$

Next, we note that $\|e_i\|_\infty \leq c_e \leq \frac{1}{4\ln(6nR/\varepsilon)}$ and $s_i\left(\boldsymbol{x}^{(k)}\right) \leq 12\sqrt{n/\lambda} + 4\sqrt{n}R + \sqrt{\frac{1}{c_a\lambda}} \leq 6\sqrt{n}R$ (Lemma 8.3.19). Hence, we have

$$p_{\boldsymbol{e}}(\boldsymbol{x}^{(k)}) \leq \frac{1}{2}\log\det\left(R^2\left(\boldsymbol{A}^T\boldsymbol{S}_{x^{(k)}}^{-2}\boldsymbol{A} + \lambda\boldsymbol{I}\right)\right) + \frac{6n}{c_a} - \frac{n}{2}\log(\frac{\min_i s_i}{R}).$$

We prove $s_i(\boldsymbol{x}^{(k)}) \leq \varepsilon$ by contradiction. Since $p_{\boldsymbol{e}}(\boldsymbol{x}^{(k)}) \geq 2n\log(\frac{nR}{c_a\varepsilon}) + \frac{6n}{c_a}$ and $s_i(\boldsymbol{x}^{(k)}) > \varepsilon$ for all i, we have that $\frac{1}{2}\log\det\left(R^2\left(\boldsymbol{A}^T\boldsymbol{S}_{x^{(k)}}^{-2}\boldsymbol{A} + \lambda\boldsymbol{I}\right)\right) \geq n\log(\frac{nR}{c_a\varepsilon})$. Using $\varepsilon < R$, we have that

$$\sum_i \log\lambda_i\left(R^2\left(\boldsymbol{A}^T\boldsymbol{S}_{x^{(k)}}^{-2}\boldsymbol{A} + \lambda\boldsymbol{I}\right)\right) \geq n\log\left(\frac{n^2R^2}{c_a^2\varepsilon^2}\right) \geq n\log\left(\frac{nR^2}{2c_a^2\varepsilon^2} + R^2\lambda\right).$$

Therefore, we have $\log\lambda_{\max}\left(R^2\left(\boldsymbol{A}^T\boldsymbol{S}_{x^{(k)}}^{-2}\boldsymbol{A} + \lambda\boldsymbol{I}\right)\right) \geq \log\left(\frac{nR^2}{2c_a^2\varepsilon^2} + R^2\lambda\right)$. Hence, we have some unit vector $\boldsymbol{v}$ such that $\boldsymbol{v}\boldsymbol{A}^T\boldsymbol{S}_x^{-2}\boldsymbol{A}\boldsymbol{v} + \lambda\boldsymbol{v}^T\boldsymbol{v} \geq \frac{n}{2c_a^2\varepsilon^2} + \lambda$. Thus,

$$\sum_i \frac{(\boldsymbol{A}\boldsymbol{v})_i^2}{s(\boldsymbol{x}^{(k)})_i^2} \geq \frac{n}{2c_a^2\varepsilon^2}.$$

Lemma 8.3.17 shows that the number of constraints is bounded by $1+2n/c_d \leq n/(2c_a^2)$ and hence there is some i such that $\frac{(\boldsymbol{A}\boldsymbol{v})_i^2}{s(\boldsymbol{x}^{(k)})_i^2} \geq \frac{1}{\varepsilon^2}$. Since $\boldsymbol{a}_i$ and $\boldsymbol{v}$ are unit vectors, we have $1 \geq \langle\boldsymbol{a}_i, \boldsymbol{v}\rangle^2 \geq s(\boldsymbol{x}^{(k)})_i^2/\varepsilon^2$ and hence $s(\boldsymbol{x}^{(k)})_i \leq \varepsilon$ (contradiction). □

Lemma 8.3.25. *With constant probability, the algorithm ends in* $10^{24}n\log(\frac{nR}{\varepsilon})$ *iterations.* [1]

Proof. Theorem 8.3.21 shows that for all k

$$\mathbb{E}p_{\boldsymbol{e}(\boldsymbol{\tau}^{(k+1)},\boldsymbol{x}^{(k+1)})}(\boldsymbol{x}^{(k+1)}) \quad \geq \quad p_{\boldsymbol{e}(\boldsymbol{\tau}^{(k)},\boldsymbol{x}^{(k)})}(\boldsymbol{x}^{(k)}) - \frac{1}{10^{11}} + \frac{9\beta}{10^{11}} \tag{8.21}$$

where $\beta = 1$ for the case of adding a constraint and $\beta = 0$ for the case of removing a constraint. Now, for all t consider the random variable

$$\boldsymbol{X}_t = p_{\boldsymbol{e}(\boldsymbol{\tau}^{(t)},\boldsymbol{x}^{(t)})}(\boldsymbol{x}^{(t)}) - \frac{4.5m^{(t)}}{10^{11}} - \frac{3.5t}{10^{11}}$$

where $m^{(t)}$ is the number of constraints in iteration t of the algorithm. Then, since $m^{(t+1)} = m^{(t)} - 1 + 2\beta$, (8.21) shows that

$$\begin{aligned}\mathbf{E}\boldsymbol{X}_{t+1} &\geq p_{\boldsymbol{e}(\boldsymbol{\tau}^{(t)},\boldsymbol{x}^{(t)})}(\boldsymbol{x}^{(t)}) - \frac{1}{10^{11}} + \frac{9\beta}{10^{11}} - \frac{4.5m^{(t+1)}}{10^{11}} - \frac{3.5(t+1)}{10^{11}} \\ &= \boldsymbol{X}_t - \frac{1}{10^{11}} + \frac{9\beta}{10^{11}} - \frac{4.5(-1+2\beta)}{10^{11}} - \frac{3.5}{10^{11}} = \boldsymbol{X}_t.\end{aligned}$$

[1] We have made no effort on improving this constant and we believe it can be improved to less than 300 using techniques in [11, 14].

Hence, it is a sub-martingale. Let T be the iteration the algorithm outputs $\boldsymbol{x}^{(k)}$ or $P^{(k)}$. Optional stopping theorem shows that

$$\mathbb{E}\boldsymbol{X}_{\min(T,t)} \geq \mathbb{E}\boldsymbol{X}_0. \tag{8.22}$$

Since the polytope is $B_\infty(R)$, we have

$$\begin{aligned} p_{\mathbf{0}}(\mathbf{0}) &= -\sum_{i\in[m^{(0)}]} c_e \log\left(s_i(\mathbf{0})/R\right) + \frac{1}{2}\log\det\left(R^2\left(\boldsymbol{A}^T\boldsymbol{S}_0^{-2}\boldsymbol{A}+\lambda\boldsymbol{I}\right)\right) + \frac{\lambda}{2}\|\mathbf{0}\|_2^2 \\ &\geq \frac{1}{2}\log\det\left(R^2\left(\frac{2}{R^2}\boldsymbol{I}+\frac{1}{c_a R^2}\boldsymbol{I}\right)\right) \\ &\geq -15n \end{aligned}$$

where we used $c_a = 1/10^{10}$ on the last line. Hence, we have

$$\begin{aligned} \boldsymbol{X}_0 &\geq -15n - \frac{4.5m^{(0)}}{10^{11}} \\ &\geq -20n. \end{aligned}$$

Therefore, (8.22) shows that for all t we have

$$\begin{aligned} -20n &\leq \mathbb{E}\boldsymbol{X}_{\min(T,t)} \\ &= p\mathbb{E}\left[\boldsymbol{X}_{\min(T,t)}|T<t\right] + (1-p)\mathbb{E}\left[\boldsymbol{X}_{\min(T,t)}|T\geq t\right] \end{aligned} \tag{8.23}$$

where $p \stackrel{\text{def}}{=} \mathbb{P}(T<t)$.

Note that

$$\begin{aligned} \mathbb{E}\left[\boldsymbol{X}_{\min(T,t)}|T\geq t\right] &\leq \mathbb{E}\left[p_{\boldsymbol{e}(\boldsymbol{\tau}^{(t)},\boldsymbol{x}^{(t)})}(\boldsymbol{x}^{(t)})|T\geq t\right] - \frac{4.5m^{(t)}}{10^{11}} - \frac{3.5t}{10^{11}}. \\ &\leq \mathbb{E}\left[p_{\boldsymbol{e}(\boldsymbol{\tau}^{(t)},\boldsymbol{x}^{(t)})}(\boldsymbol{x}^{(t)})|T\geq t\right] - \frac{3.5t}{10^{11}}. \end{aligned}$$

Furthermore, by Lemma 8.3.24 we know that when $p_{\boldsymbol{e}(\boldsymbol{\tau}^{(t)},\boldsymbol{x}^{(t)})}(\boldsymbol{x}^{(t)}) \geq 2n\log(\frac{nR}{c_a\varepsilon}) + \frac{6n}{c_a}$, there is a slack that is too small and the algorithm terminates. Hence, we have

$$\mathbf{E}\left[\boldsymbol{X}_{\min(T,t)}|T\geq t\right] \leq 2n\log(\frac{nR}{c_a\varepsilon}) + \frac{6n}{c_a} - \frac{3.5t}{10^{11}}.$$

The proof of Lemma 8.3.16 shows that the function value does not change by more than 1 in one iteration by changing $\boldsymbol{x}$ and can change by at most $mc_e\log(\frac{3nR}{\varepsilon})$ by changing τ. Since by Lemma 8.3.17 we know that $m \leq 1+\frac{2n}{c_a}$ and $c_e = \frac{c_d}{4\ln(6nR/\varepsilon)}$, we have that $p_{\boldsymbol{e}}(\boldsymbol{x}) \leq 2n\log(\frac{nR}{c_a\varepsilon}) + \frac{7n}{c_a}$ throughout the execution of the algorithm. Therefore, we have

$$\mathbf{E}\left[\boldsymbol{X}_{\min(T,t)}|T\leq t\right] \leq \mathbb{E}_{T<t}p_{\boldsymbol{e}(\boldsymbol{\tau}^{(t)},\boldsymbol{x}^{(t)})}(\boldsymbol{x}^{(t)}) \leq 2n\log(\frac{nR}{c_a\varepsilon}) + \frac{7n}{c_a}.$$

Therefore, (8.23) shows that

$$-20n \leq 2n\log\left(\frac{nR}{c_a\varepsilon}\right) + \frac{7n}{c_a} - (1-p)\frac{3.5t}{10^{11}}.$$

Hence, we have

$$\begin{aligned}(1-p)\frac{3.5t}{10^{11}} &\leq 2n\log\left(\frac{nR}{c_a\varepsilon}\right)+\frac{7n}{c_a}+20n\\ &\leq 2n\log\left(\frac{nR}{c_a\varepsilon}\right)+\frac{7n}{c_a}+20n\\ &= 2n\log\left(\frac{Rn}{c_a\varepsilon}\right)+8\cdot 10^{10}n.\end{aligned}$$

Thus, we have

$$\mathbb{P}(T<t)=p\geq 1-\frac{1}{t}\left(10^{11}n\log\left(\frac{nR}{\varepsilon}\right)+10^{22}n\right).$$

□

Now, we gather all the result as follows:

Theorem 8.3.26 (Our Cutting Plane Method)**.** *Let $K\subseteq\mathbb{R}^n$ be a non-empty set contained in a box of radius R, i.e. $K\subseteq B_\infty(R)$. For any $\varepsilon\in(0,R)$ in expected time $O(n\mathrm{SO}_{\Omega(\varepsilon/\sqrt{n})}(K)\log(nR/\varepsilon)+n^3\log^{O(1)}(nR/\varepsilon))$ our cutting plane method either outputs $\boldsymbol{x}\in K$ or finds a polytope $P=\{\boldsymbol{x}:\boldsymbol{A}\boldsymbol{x}\geq\boldsymbol{b}\}\supseteq K$ such that*

1. *P has $O(n)$ many constraints (i.e. $\boldsymbol{A}\in\mathbb{R}^{O(n)\times n}$ and $\boldsymbol{b}\in\mathbb{R}^{O(n)}$).*
2. *Each constraint of P is either an initial constraint from $B_\infty(R)$ or of the form $\langle\boldsymbol{a},\boldsymbol{x}\rangle\geq b-\delta$ where $\langle\boldsymbol{a},\boldsymbol{x}\rangle\geq b$ is a normalized hyperplane (i.e. $\|\boldsymbol{a}\|_2=1$) returned by the separation oracle and $\delta=\Omega\left(\frac{\varepsilon}{\sqrt{n}}\right)$.*
3. *The polytope P has small width with respect to some direction $\boldsymbol{a}_1$ given by one of the constraints, i.e.*
$$\max_{\boldsymbol{y}\in P\cap B_\infty(R)}\langle\boldsymbol{a}_1,\boldsymbol{y}\rangle-\min_{\boldsymbol{y}\in P\cap B_\infty(R)}\langle\boldsymbol{a}_1,\boldsymbol{y}\rangle\leq O\left(n\varepsilon\ln(nR/\varepsilon)\right)$$
4. *Furthermore, the algorithm produces a proof of the fact above involving convex combination of the constraints, namely, non-negatives $t_2,\ldots,t_{O(n)}$ and $\boldsymbol{x}\in P$ such that*
   - *(a) $\|\boldsymbol{x}\|_2=O\left(\sqrt{n}R\right)$,*
   - *(b) $\left\|\boldsymbol{a}_1+\sum_{i=2}^{O(n)}t_i\boldsymbol{a}_i\right\|_2=O\left(\frac{\varepsilon}{R}\sqrt{n}\log(nR/\varepsilon)\right)$,*
   - *(c) $\boldsymbol{a}_1^T\boldsymbol{x}-\boldsymbol{b}_1\leq\varepsilon$,*
   - *(d) $\left(\sum_{i=2}^{O(n)}t_ia_i\right)^T\boldsymbol{x}-\sum_{i=2}^{O(n)}t_ib_i\leq O(n\varepsilon\log(nR/\varepsilon))$.*

Proof. Our algorithm either finds $\boldsymbol{x}\in K$ or we have $s_i(\boldsymbol{x}^{(k)})<\varepsilon$. When $s_i(\boldsymbol{x}^{(k)})<\varepsilon$, we apply Lemma 8.3.23 to construct the polytope P and the linear combination $\sum_{i=2}^{O(n)}t_i\boldsymbol{a}_i$.

Notice that each iteration of our algorithm needs to solve constant number of linear systems and implements the sampling step to find $\boldsymbol{\Delta}^{(k)}\in\mathbb{R}^n$ s.t. $\mathbf{E}[\boldsymbol{\Delta}^{(k)}]=\boldsymbol{\psi}(\boldsymbol{x}^{(k)})-\boldsymbol{\psi}(\boldsymbol{x}^{(k-1)})$. Theorem 8.4.2 shows how to do the sampling in $\tilde{O}(1)$ many linear systems. Hence, in total, each iterations needs to solve $\tilde{O}(1)$ many linear systems plus nearly linear work. To output the proof for (4), we use Lemma 8.3.23.

Note that the linear systems the whole algorithm need to solve is of the form

$$(\boldsymbol{A}^T\boldsymbol{S}_x^{-2}\boldsymbol{A}+\lambda\boldsymbol{I})^{-1}\boldsymbol{x}=\boldsymbol{y}.$$

where the matrix $\boldsymbol{A}^T\boldsymbol{S}_x^{-2}\boldsymbol{A}+\lambda\boldsymbol{I}$ can be written as $\overline{\boldsymbol{A}}^T\boldsymbol{D}\overline{\boldsymbol{A}}$ for the matrix $\overline{\boldsymbol{A}}=[\boldsymbol{A}\ \boldsymbol{I}]$ and diagonal matrix

$$\boldsymbol{D}=\begin{bmatrix}\boldsymbol{S}^{-2} & \boldsymbol{0}\\ \boldsymbol{0} & \lambda\boldsymbol{I}\end{bmatrix}.$$

Note that Lemma 8.3.9 shows that $\|\left(\boldsymbol{S}^{(k)}\right)^{-1}(\boldsymbol{s}^{(k+1)}-\boldsymbol{s}^{(k)})\|_2\leq\frac{1}{10}$ for the k^{th} and $(k+1)^{th}$ linear systems we solved in the algorithm. Hence, we have $\|\left(\boldsymbol{D}^{(k)}\right)^{-1}(\boldsymbol{d}^{(k+1)}-\boldsymbol{d}^{(k)})\|_2\leq\frac{1}{10}$. In [164], they showed how to solve such sequence of systems in $\tilde{O}(n^2)$ amortized cost. Moreover, since our algorithm always changes the constraints by δ amount where $\delta=\Omega(\frac{\varepsilon}{\sqrt{n}})$ an inexact separation oracle $\mathrm{SO}_{\Omega(\varepsilon/\sqrt{n})}$ suffices. (see Def 2.3.5). Consequently, the total work $O(n\mathrm{SO}_{\Omega(\varepsilon/\sqrt{n})}(K)\log(nR/\varepsilon)+n^3\log^{O(1)}(nR/\varepsilon))$. Note that as the running time holds with only constant probability, we can restart the algorithm whenever the running time is too large.

To prove (2), we note that from the algorithm description, we know the constraints are either from $B_\infty(R)$ or of the form $\boldsymbol{a}^T\boldsymbol{x}\geq\boldsymbol{a}^T\boldsymbol{x}^{(k)}-\delta$ where

$$\delta=\sqrt{\frac{\boldsymbol{a}^T(\boldsymbol{A}^T\boldsymbol{S}_{\boldsymbol{x}^{(k)}}^{-2}\boldsymbol{A}+\lambda\boldsymbol{I})^{-1}\boldsymbol{a}}{c_a}}.$$

From the proof of Lemma 8.3.24, we know that if $\lambda_{\max}(\boldsymbol{A}^T\boldsymbol{S}_x^{-2}\boldsymbol{A}+\lambda\boldsymbol{I})\geq\frac{n}{c_a^2\varepsilon^2}$, then there is $s_i<\varepsilon$. Hence, we have $\lambda_{\min}((\boldsymbol{A}^T\boldsymbol{S}_x^{-2}\boldsymbol{A}+\lambda\boldsymbol{I})^{-1})\geq\frac{c_a^2\varepsilon^2}{n}$. Since $\boldsymbol{a}$ is a unit vector, we have

$$\sqrt{\frac{\boldsymbol{a}^T(\boldsymbol{A}^T\boldsymbol{S}_{\boldsymbol{x}^{(k)}}^{-2}\boldsymbol{A}+\lambda\boldsymbol{I})^{-1}\boldsymbol{a}}{c_a}}\geq\sqrt{\frac{c_a\varepsilon^2}{n}}.$$

□

8.4 Technical Tools

In this section we provide stand-alone technical tools we use in our cutting plane method in Section 8.3. In Section 8.4.1 we show how to efficiently compute accurate estimates of changes in leverage scores using access to a linear system solver. In Section 8.4.2 we study what we call the "Stochastic Chasing **0** Game" and show how to maintain that a vector is small in ℓ_∞ norm by making small coordinate updates while the vector changes randomly in ℓ_2.

8.4.1 Estimating Changes in Leverage Scores

In previous sections, we needed to compute leverage scores accurately and efficiently for use in our cutting plane method. Note that the leverage score definition we used was

$$\psi(\boldsymbol{w})_i=\boldsymbol{1}_i^T\sqrt{\boldsymbol{W}}\boldsymbol{A}\left(\boldsymbol{A}^T\boldsymbol{W}\boldsymbol{A}+\lambda\boldsymbol{I}\right)^{-1}\boldsymbol{A}^T\sqrt{\boldsymbol{W}}\boldsymbol{1}_i$$

for some $\lambda>0$ which is different from the standard definition

$$\sigma(\boldsymbol{w})_i=\boldsymbol{1}_i^T\sqrt{\boldsymbol{W}}\boldsymbol{A}\left(\boldsymbol{A}^T\boldsymbol{W}\boldsymbol{A}\right)^{-1}\boldsymbol{A}^T\sqrt{\boldsymbol{W}}\boldsymbol{1}_i.$$

However, note that the matrix $\boldsymbol{A}^T\boldsymbol{W}\boldsymbol{A} + \lambda\boldsymbol{I}$ can be written as $\overline{\boldsymbol{A}}^T\boldsymbol{D}\overline{\boldsymbol{A}}$ for the matrix $\overline{\boldsymbol{A}}^T = [\boldsymbol{A}^T\ \boldsymbol{I}]$ and diagonal matrix

$$\boldsymbol{D} = \begin{bmatrix} \boldsymbol{W} & \boldsymbol{0} \\ \boldsymbol{0} & \lambda\boldsymbol{I} \end{bmatrix}.$$

and therefore computing ψ is essentially same as computing typical leverage scores. Consequently, we use the standard definition σ to simplify notation.

We explained in Section 2.3.1 that leverage scores can be approximated efficiently using dimension reduction. Unfortunately, the error incurred by this approximation is too large to use inside the cutting point method. In this section, we show how to efficiently approximate the *change* of leverage score more accurately.

In particular, we show how to approximate $\sigma(\boldsymbol{w}) - \sigma(\boldsymbol{v})$ for any given $\boldsymbol{w}, \boldsymbol{v}$ with $\|\log(\boldsymbol{w}) - \log(\boldsymbol{v})\|_2 \ll 1$. Our algorithm breaks $\sigma(\boldsymbol{w})_i - \sigma(\boldsymbol{v})_i$ into the sum of the norm of small vectors and then uses the Johnson-Lindenstrauss dimension reduction to approximate the norm of each vector separately. Our algorithm makes use of the following version of Johnson-Lindenstrauss.

Lemma 8.4.1 ([2]). *Let $0 \leq \varepsilon \leq \frac{1}{2}$ and let $\boldsymbol{x}_1, ..., \boldsymbol{x}_m \in \mathbb{R}^n$ be arbitrary m points. For $k = O(\varepsilon^{-2}\log(m))$ let $\boldsymbol{Q}$ be a $k \times n$ random matrix with each entry sampled from $\{-\frac{1}{\sqrt{k}}, \frac{1}{\sqrt{k}}\}$ uniformly and independently. Then, $\mathbb{E}\|\boldsymbol{Q}\boldsymbol{x}_i\|^2 = \|\boldsymbol{x}_i\|^2$ for all $i \in [m]$ and with high probability in m we have that for all $i \in [m]$*

$$(1-\varepsilon)\|\boldsymbol{x}_i\|^2 \leq \|\boldsymbol{Q}\boldsymbol{x}_i\|^2 \leq (1+\varepsilon)\|\boldsymbol{x}_i\|^2 \quad .$$

Algorithm 23: $\widehat{h} = \texttt{LeverageChange}(\boldsymbol{A}, \boldsymbol{v}, \boldsymbol{w}, \varepsilon)$

Input: $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, $\boldsymbol{v}, \boldsymbol{w} \in \mathbb{R}^m_{>0}$, $\varepsilon \in (0, 0.5)$.
Given: $\|\boldsymbol{V}^{-1}(\boldsymbol{v}-\boldsymbol{w})\|_2 \leq \frac{1}{10}$; $\boldsymbol{A}^T\boldsymbol{V}\boldsymbol{A}$ and $\boldsymbol{A}^T\boldsymbol{W}\boldsymbol{A}$ are invertible.
Sample $\boldsymbol{Q}_d \in \mathbb{R}^{O(\varepsilon^{-2}\log(m))\times n}$ as in Lemma 8.4.1.
Let $\hat{d}_i = \|\boldsymbol{Q}_d\sqrt{\boldsymbol{W}}\boldsymbol{A}\left(\boldsymbol{A}^T\boldsymbol{W}\boldsymbol{A}\right)^{-1}\boldsymbol{A}^T\boldsymbol{1}_i\|_2^2$ for all $i \in [n]$.
Let $t = O\left(\log(\varepsilon^{-1})\right)$.
Sample $\boldsymbol{Q}_f \in \mathbb{R}^{O(\varepsilon^{-2}\log(mt))\times n}$ as in Lemma 8.4.1.
Pick positive integer u randomly such that $\Pr[u = i] = (\frac{1}{2})^i$.
for $j \in \{1, 2, \cdots, t\} \cup \{t+u\}$ **do**
  if j *is even* **then**
    Let $\hat{f}_i^{(j)} = \|\boldsymbol{Q}_f\sqrt{\boldsymbol{V}}\boldsymbol{A}\left(\boldsymbol{A}^T\boldsymbol{V}\boldsymbol{A}\right)^{-1}\left(\boldsymbol{A}^T\left(\boldsymbol{V}-\boldsymbol{W}\right)\boldsymbol{A}\left(\boldsymbol{A}^T\boldsymbol{V}\boldsymbol{A}\right)^{-1}\right)^{\frac{j}{2}}\boldsymbol{A}^T\boldsymbol{1}_i\|_2^2$.
  else
    Let $\Delta^+ \stackrel{\text{def}}{=} (\boldsymbol{V}-\boldsymbol{W})^+$, i.e. the matrix $\boldsymbol{V}-\boldsymbol{W}$ with negative entries set to 0.
    Let $\Delta^- \stackrel{\text{def}}{=} (\boldsymbol{W}-\boldsymbol{V})^+$, i.e. the matrix $\boldsymbol{W}-\boldsymbol{V}$ with negative entries set to 0.
    Let $\hat{\alpha}_i^{(j)} = \|\boldsymbol{Q}_f\sqrt{\Delta^+}\boldsymbol{A}(\boldsymbol{A}^T\boldsymbol{V}\boldsymbol{A})^{-1}(\boldsymbol{A}^T\left(\boldsymbol{V}-\boldsymbol{W}\right)\boldsymbol{A}\left(\boldsymbol{A}^T\boldsymbol{V}\boldsymbol{A}\right)^{-1})^{\frac{j-1}{2}}\boldsymbol{A}^T\boldsymbol{1}_i\|_2^2$.
    Let $\hat{\beta}_i^{(j)} = \|\boldsymbol{Q}_f\sqrt{\Delta^-}\boldsymbol{A}(\boldsymbol{A}^T\boldsymbol{V}\boldsymbol{A})^{-1}(\boldsymbol{A}^T(\boldsymbol{V}-\boldsymbol{W})\boldsymbol{A}(\boldsymbol{A}^T\boldsymbol{V}\boldsymbol{A})^{-1})^{\frac{j-1}{2}}\boldsymbol{A}^T\boldsymbol{1}_i\|_2^2$.
    Let $\hat{f}_i^{(j)} = \hat{\alpha}_i^{(j)} - \hat{\beta}_i^{(j)}$.
  end
end
Let $\hat{f}_i = 2^u\hat{f}_i^{(t+u)} + \sum_{j=1}^t \hat{f}_i^{(j)}$.
Output: $\hat{h}_i = (w_i - v_i)\hat{d}_i + v_i\hat{f}_i$. for all $i \in [m]$

Theorem 8.4.2. *Let $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ and $\boldsymbol{v}, \boldsymbol{w} \in \mathbb{R}^m_{>0}$ be such that $\alpha \stackrel{\text{def}}{=} \|\boldsymbol{V}^{-1}(\boldsymbol{v}-\boldsymbol{w})\|_2 \leq \frac{1}{10}$ and both $\boldsymbol{A}^T\boldsymbol{V}\boldsymbol{A}$ and $\boldsymbol{A}^T\boldsymbol{W}\boldsymbol{A}$ are invertible. For any $\varepsilon \in (0, 0.5)$, Algorithm 23 generates a random variable $\widehat{h}$*

such that $\mathbf{E}\hat{h} = \sigma(\boldsymbol{w}) - \sigma(\boldsymbol{v})$ *and with high probability in* m*, we have* $\|\widehat{h} - (\sigma(\boldsymbol{w}) - \sigma(\boldsymbol{v}))\|_2 \leq O(\alpha\varepsilon)$*. Furthermore, the expected running time is* $\widetilde{O}((\mathrm{nnz}(\boldsymbol{A}) + \mathrm{LO})/\varepsilon^2)$ *where* LO *is the amount of time needed to apply* $\left(\boldsymbol{A}^T\boldsymbol{V}\boldsymbol{A}\right)^{-1}$ *and* $\left(\boldsymbol{A}^T\boldsymbol{W}\boldsymbol{A}\right)^{-1}$ *to a vector.*

Proof. First we bound the running time. To compute $\hat{d}_i, \hat{f}_i^{(j)}, \hat{\alpha}_i^{(j)}, \hat{\beta}_i^{(j)}$, we simply perform matrix multiplications from the left and then consider the dot products with each of the rows of $\boldsymbol{A}$. Naively this would take time $\widetilde{O}((t+u)^2 \log(mt)(\mathrm{nnz}(\boldsymbol{A}) + \mathrm{LO}))$. However, we can reuse the computation in computing high powers of j to only take time $\widetilde{O}((t+u)\log(mt)(\mathrm{nnz}(\boldsymbol{A}) + \mathrm{LO}))$. Now since $\mathbf{E}[u]$ is constant we see that the total running time is as desired. It only remains to prove the desired properties of $\hat{h}$.

First we note that we can re-write leverage score differences using

$$\sigma(\boldsymbol{w})_i - \sigma(\boldsymbol{v})_i = (w_i - v_i)\left[\boldsymbol{A}\left(\boldsymbol{A}^T\boldsymbol{W}\boldsymbol{A}\right)^{-1}\boldsymbol{A}^T\right]_{ii} + v_i\left[\boldsymbol{A}\left(\left(\boldsymbol{A}^T\boldsymbol{W}\boldsymbol{A}\right)^{-1} - \left(\boldsymbol{A}^T\boldsymbol{V}\boldsymbol{A}\right)^{-1}\right)\boldsymbol{A}^T\right]_{ii} \quad .$$

Consequently, for all $i \in [m]$, if we let

$$\begin{aligned} d_i &\stackrel{\text{def}}{=} \mathbf{1}_i^T\boldsymbol{A}\left(\boldsymbol{A}^T\boldsymbol{W}\boldsymbol{A}\right)^{-1}\boldsymbol{A}^T\mathbf{1}_i, \\ f_i &\stackrel{\text{def}}{=} \mathbf{1}_i^T\boldsymbol{A}\left[\left(\boldsymbol{A}^T\boldsymbol{W}\boldsymbol{A}\right)^{-1} - \left(\boldsymbol{A}^T\boldsymbol{V}\boldsymbol{A}\right)^{-1}\right]\boldsymbol{A}^T\mathbf{1}_i. \end{aligned}$$

then

$$\sigma(\boldsymbol{w})_i - \sigma(\boldsymbol{v})_i = (w_i - v_i)d_i + (v_i)f_i \quad . \tag{8.24}$$

We show that $\hat{d}_i$ approximates d and $\hat{f}_i$ approximate $\hat{f}$ well enough to satisfy the statements in the Theorem.

First we bound the quality of $\hat{d}_i$. Note that $d_i = \|\sqrt{\boldsymbol{W}}\boldsymbol{A}\left(\boldsymbol{A}^T\boldsymbol{W}\boldsymbol{A}\right)^{-1}\boldsymbol{A}^T\mathbf{1}_i\|_2^2$. Consequently, Lemma 8.4.1 shows that $\mathbf{E}[\hat{d}_i] = d_i$ and that with high probability in m we have $(1-\varepsilon)d_i \leq \hat{d}_i \leq (1+\varepsilon)d_i$ for all $i \in [m]$. Therefore, with high probability in m, we have

$$\begin{aligned} \|(\boldsymbol{w}-\boldsymbol{v})\widehat{d} - (\boldsymbol{w}-\boldsymbol{v})\boldsymbol{d}\|_2^2 &= \sum_{i\in[m]}(w_i - v_i)^2\left(\widehat{d}_i - d_i\right)^2 \leq \varepsilon^2\sum_{i\in[m]}(w_i - v_i)^2 d_i^2 \\ &= \varepsilon^2\sum_{i\in[m]}(w_i - v_i)^2\left(\frac{\sigma(\boldsymbol{w})_i}{\boldsymbol{w}_i}\right)^2 \leq 2\varepsilon^2\sum_{i\in[m]}\left(\frac{w_i - v_i}{v_i}\right)^2 \quad . \end{aligned}$$

Next we show how to estimate f. Let $\boldsymbol{X} \stackrel{\text{def}}{=} \left(\boldsymbol{A}^T\boldsymbol{V}\boldsymbol{A}\right)^{-1/2}\boldsymbol{A}^T(\boldsymbol{V}-\boldsymbol{W})\boldsymbol{A}\left(\boldsymbol{A}^T\boldsymbol{V}\boldsymbol{A}\right)^{-1/2}$. By the assumption on α we know $-\frac{1}{2}\boldsymbol{V} \prec \boldsymbol{V}-\boldsymbol{W} \prec \frac{1}{2}\boldsymbol{V}$ and therefore $-\frac{1}{2}\boldsymbol{I} \prec \boldsymbol{X} \prec \frac{1}{2}\boldsymbol{I}$. Consequently we have that

$$\begin{aligned} \left(\boldsymbol{A}^T\boldsymbol{W}\boldsymbol{A}\right)^{-1} &= \left(\boldsymbol{A}^T\boldsymbol{V}\boldsymbol{A}\right)^{-1/2}(\boldsymbol{I}-\boldsymbol{X})^{-1}\left(\boldsymbol{A}^T\boldsymbol{V}\boldsymbol{A}\right)^{-1/2} \\ &= \sum_{j=0}^{\infty}\left(\boldsymbol{A}^T\boldsymbol{V}\boldsymbol{A}\right)^{-1/2}\boldsymbol{X}^j\left(\boldsymbol{A}^T\boldsymbol{V}\boldsymbol{A}\right)^{-1/2}. \end{aligned}$$

and therefore

$$\begin{aligned} f_i &= \mathbf{1}_i^T\boldsymbol{A}\left(\sum_{j=0}^{\infty}\left(\boldsymbol{A}^T\boldsymbol{V}\boldsymbol{A}\right)^{-1/2}\boldsymbol{X}^j\left(\boldsymbol{A}^T\boldsymbol{V}\boldsymbol{A}\right)^{-1/2} - \left(\boldsymbol{A}^T\boldsymbol{V}\boldsymbol{A}\right)^{-1}\right)\boldsymbol{A}^T\mathbf{1}_i \\ &= \sum_{j=1}^{\infty} f_i^{(j)} \quad \text{where} \quad f_i^{(j)} \stackrel{\text{def}}{=} \mathbf{1}_i^T\boldsymbol{A}\left(\boldsymbol{A}^T\boldsymbol{V}\boldsymbol{A}\right)^{-1/2}\boldsymbol{X}^j\left(\boldsymbol{A}^T\boldsymbol{V}\boldsymbol{A}\right)^{-1/2}\boldsymbol{A}^T\mathbf{1}_i \quad . \end{aligned}$$

Furthermore, using the definition of $\boldsymbol{X}$ we have that for even j

$$\begin{aligned} f_i^{(j)} &= \left\| \boldsymbol{X}^{\frac{j}{2}} \left(\boldsymbol{A}^T \boldsymbol{V} \boldsymbol{A}\right)^{-1/2} \boldsymbol{A}^T \mathbf{1}_i \right\|_2^2 \\ &= \left\| \left(\boldsymbol{A}^T \boldsymbol{V} \boldsymbol{A}\right)^{-1/2} \left(\boldsymbol{A}^T \left(\boldsymbol{V} - \boldsymbol{W}\right) \boldsymbol{A} \left(\boldsymbol{A}^T \boldsymbol{V} \boldsymbol{A}\right)^{-1} \right)^{\frac{j}{2}} \boldsymbol{A}^T \mathbf{1}_i \right\|_2^2 \\ &= \left\| \sqrt{\boldsymbol{V}} \boldsymbol{A} \left(\boldsymbol{A}^T \boldsymbol{V} \boldsymbol{A}\right)^{-1} \left(\boldsymbol{A}^T \left(\boldsymbol{V} - \boldsymbol{W}\right) \boldsymbol{A} \left(\boldsymbol{A}^T \boldsymbol{V} \boldsymbol{A}\right)^{-1} \right)^{\frac{j}{2}} \boldsymbol{A}^T \mathbf{1}_i \right\|_2^2 \end{aligned}$$

For odd j, using our definition of $\boldsymbol{\Delta}^+$ and $\boldsymbol{\Delta}^-$ we have that

$$\begin{aligned} f_i^{(j)} &= \mathbf{1}_i^T \boldsymbol{A} \left(\boldsymbol{A}^T \boldsymbol{V} \boldsymbol{A}\right)^{-1/2} \boldsymbol{X}^j \left(\boldsymbol{A}^T \boldsymbol{V} \boldsymbol{A}\right)^{-1/2} \boldsymbol{A}^T \mathbf{1}_i \\ &= \mathbf{1}_i^T \boldsymbol{A} \left(\left(\boldsymbol{A}^T \boldsymbol{V} \boldsymbol{A}\right)^{-1} \boldsymbol{A}^T \left(\boldsymbol{V} - \boldsymbol{W}\right) \boldsymbol{A} \right)^{\frac{j-1}{2}} \left(\boldsymbol{A}^T \boldsymbol{V} \boldsymbol{A}\right)^{-1} \boldsymbol{A}^T \left(\boldsymbol{V} - \boldsymbol{W}\right) \\ &\quad \times \boldsymbol{A} \left(\boldsymbol{A}^T \boldsymbol{V} \boldsymbol{A}\right)^{-1} \left(\boldsymbol{A}^T \left(\boldsymbol{V} - \boldsymbol{W}\right) \boldsymbol{A} \left(\boldsymbol{A}^T \boldsymbol{V} \boldsymbol{A}\right)^{-1} \right)^{\frac{j-1}{2}} \boldsymbol{A}^T \mathbf{1}_i \\ &= \alpha_i^{(j)} - \beta_i^{(j)} \end{aligned}$$

where

$$\begin{aligned} \alpha_i^{(j)} &\stackrel{\text{def}}{=} \left\| \sqrt{\Delta^+} \boldsymbol{A} \left(\boldsymbol{A}^T \boldsymbol{V} \boldsymbol{A}\right)^{-1} \left(\boldsymbol{A}^T \left(\boldsymbol{W} - \boldsymbol{V}\right) \boldsymbol{A} \left(\boldsymbol{A}^T \boldsymbol{V} \boldsymbol{A}\right)^{-1} \right)^{\frac{j-1}{2}} \boldsymbol{A}^T \mathbf{1}_i \right\|_2^2 , \\ \beta_i^{(j)} &\stackrel{\text{def}}{=} \left\| \sqrt{\Delta^-} \boldsymbol{A} \left(\boldsymbol{A}^T \boldsymbol{V} \boldsymbol{A}\right)^{-1} \left(\boldsymbol{A}^T \left(\boldsymbol{W} - \boldsymbol{V}\right) \boldsymbol{A} \left(\boldsymbol{A}^T \boldsymbol{V} \boldsymbol{A}\right)^{-1} \right)^{\frac{j-1}{2}} \boldsymbol{A}^T \mathbf{1}_i \right\|_2^2 . \end{aligned}$$

Consequently, by Lemma 8.4.1 and the construction, we see that

$$\mathbf{E}\hat{f}_i = \sum_{j=1}^{\infty} f_i^{(j)} = f_i$$

and therefore $\mathbf{E}\hat{h} = \sigma(\boldsymbol{w}) - \sigma(\boldsymbol{v})$ as desired. All that remains is to bound the variance of $\hat{f}_i$.

To bound the variance of $\hat{f}$, let $|\boldsymbol{X}| = \left(\boldsymbol{A}^T \boldsymbol{V} \boldsymbol{A}\right)^{-1/2} \boldsymbol{A}^T \left|\boldsymbol{W} - \boldsymbol{V}\right| \boldsymbol{A} \left(\boldsymbol{A}^T \boldsymbol{V} \boldsymbol{A}\right)^{-1/2}$. Note that $-\frac{1}{4}\boldsymbol{I} \preceq -|\boldsymbol{X}| \preceq \boldsymbol{X} \preceq |\boldsymbol{X}| \preceq \frac{1}{4}\boldsymbol{I}$ and consequently for all j

$$\begin{aligned} g_i^{(j)} &\stackrel{\text{def}}{=} \mathbf{1}_i^T \boldsymbol{A} \left(\boldsymbol{A}^T \boldsymbol{V} \boldsymbol{A}\right)^{-1/2} |\boldsymbol{X}|^j \left(\boldsymbol{A}^T \boldsymbol{V} \boldsymbol{A}\right)^{-1/2} \boldsymbol{A}^T \mathbf{1}_i \\ &\leq \frac{1}{4^{j-1}} \mathbf{1}_i^T \boldsymbol{A} \left(\boldsymbol{A}^T \boldsymbol{V} \boldsymbol{A}\right)^{-1/2} |\boldsymbol{X}| \left(\boldsymbol{A}^T \boldsymbol{V} \boldsymbol{A}\right)^{-1/2} \boldsymbol{A}^T \mathbf{1}_i \\ &\stackrel{\text{def}}{=} \frac{1}{v_i 4^{j-1}} \mathbf{1}_i^T \boldsymbol{P}_v \boldsymbol{\Delta} \boldsymbol{P}_v \mathbf{1}_i \end{aligned}$$

where $\boldsymbol{P}_v = \sqrt{\boldsymbol{V}} \boldsymbol{A} \left(\boldsymbol{A}^T \boldsymbol{V} \boldsymbol{A}\right)^{-1/2} \boldsymbol{A}^T \sqrt{\boldsymbol{V}}$ and $\boldsymbol{\Delta}$ is a diagonal matrix with $\boldsymbol{\Delta}_{ii} = \left|\frac{w_i - v_i}{v_i}\right|$. Using that $\mathbf{0} \preceq \boldsymbol{P}_v \preceq \boldsymbol{I}$, we have that for all j

$$\begin{aligned}(4^{j-1})^2\sum_{i=1}^{m}\left(v_i g_i^{(j)}\right)^2 &= \sum_{i=1}^{m}\left(\mathbf{1}_i^T\boldsymbol{P}_v\boldsymbol{\Delta}\boldsymbol{P}_v\mathbf{1}_i\right)^2 = \mathrm{Tr}\left(\boldsymbol{P}_v\boldsymbol{\Delta}\boldsymbol{P}_v\boldsymbol{P}_v\boldsymbol{\Delta}\boldsymbol{P}_v\right)\\ &\le \mathrm{Tr}\left(\boldsymbol{P}_v\boldsymbol{\Delta}\boldsymbol{\Delta}\boldsymbol{P}_v\right) = \mathrm{Tr}\left(\boldsymbol{\Delta}\boldsymbol{P}_v\boldsymbol{P}_v\boldsymbol{\Delta}\right)\\ &\le \mathrm{Tr}\left(\boldsymbol{\Delta}^2\right) = \sum_{i=1}^{m}\left(\frac{w_i-v_i}{v_i}\right)^2 \le \alpha^2\end{aligned}$$

and thus $\|\boldsymbol{V}\boldsymbol{g}^{(j)}\|_2 \le \frac{4\alpha}{4^j}$. Furthermore, since $\boldsymbol{\Delta}^+ \preceq |\boldsymbol{W}-\boldsymbol{V}|$ and $\boldsymbol{\Delta}^- \preceq |\boldsymbol{W}-\boldsymbol{V}|$ we have that $\left|\alpha_i^{(j)}\right| \le g_i^{(j)}$ and $\left|\beta_i^{(j)}\right| \le g_i^{(j)}$. Consequently, by Lemma 8.4.1 again, we have

$$\begin{aligned}\|\boldsymbol{V}\hat{f}^{(j)} - \boldsymbol{V}\boldsymbol{f}^{(j)}\|_2^2 &= \sum_i v_i^2\left(\hat{f}_i^{(j)} - f_i^{(j)}\right)^2\\ &\le 2\sum_i v_i^2\left(\hat{\alpha}_i^{(j)} - \alpha_i^{(j)}\right)^2 + 2\sum_i v_i^2\left(\hat{\beta}_i^{(j)} - \beta_i^{(j)}\right)^2\\ &\le 2\varepsilon^2\sum_i v_i^2\left(\left(\alpha_i^{(j)}\right)^2 + \left(\beta_i^{(j)}\right)^2\right)\\ &\le 4\varepsilon^2\sum_i\left(v_i g_i^{(j)}\right)^2 \le \frac{4\alpha^2\varepsilon^2}{\left(4^{j-1}\right)^2}\,.\end{aligned}$$

Putting this all together we have that

$$\begin{aligned}\|\boldsymbol{V}\hat{f} - \boldsymbol{V}\boldsymbol{f}\|_2 &\le \|2^u\boldsymbol{V}\hat{f}^{(t+u)} + \sum_{j=1}^{t}\boldsymbol{V}\hat{f}^{(j)} - \sum_{j=1}^{\infty}\boldsymbol{V}\boldsymbol{f}^{(j)}\|_2\\ &\le 2^u\|\boldsymbol{V}\hat{f}^{(t+u)}\|_2 + \sum_{j=1}^{t}\|\boldsymbol{V}\hat{f}^{(j)} - \boldsymbol{V}\boldsymbol{f}^{(j)}\|_2 + \sum_{j=t+1}^{\infty}\|\boldsymbol{V}\boldsymbol{f}^{(j)}\|_2\\ &\le 2^u\frac{4\alpha}{4^{t+u-1}} + \sum_{j=1}^{t}\frac{2\alpha\varepsilon}{4^{j-1}} + \sum_{j=t+1}^{\infty}\frac{2\alpha}{4^{j-1}}\\ &= O\left(\alpha\varepsilon + \frac{\alpha}{4^t}\right).\end{aligned}$$

Consequently, since $t = O(\log(\varepsilon^{-1}))$ we have the desired result. □

8.4.2 The Stochastic Chasing 0 Game

To avoid computing leverage scores exactly, in Section 8.4.1 we showed how to estimate the difference of leverage scores and use these to update the leverage scores. However, if we only applied this technique, the error of leverage scores would accumulate in the algorithm and we need to fix it. Naturally, one may wish to use dimension reduction to compute a multiplicative approximation to the leverage scores and update our computed value if the error is too large. However, this strategy would fail if there are too many rows with inaccurate leverage scores in the same iteration. In this case, we would change the central point too much that we are not able to recover. In this section, we present this update problem in a general form that we call *Stochastic Chasing 0 game* and provide an effective strategy for playing this game.

The *Stochastic chasing 0 game* is as follows. There is a player, a stochastic adversary, and a point

$\boldsymbol{x} \in \mathbb{R}^m$. The goal of the player is to keep the point close to $\mathbf{0} \in \mathbb{R}^m$ in ℓ_∞ norm and the goal of the stochastic adversary is to move $\boldsymbol{x}$ away from $\mathbf{0}$. The game proceeds for an infinite number of iterations where in each iteration the stochastic adversary moves the current point $\boldsymbol{x}^{(k)} \in \mathbb{R}^m$ to some new point $\boldsymbol{x}^{(k)} + \boldsymbol{\Delta}^{(k)} \in \mathbb{R}^m$ and the player needs to respond. The stochastic adversary cannot move the $\boldsymbol{\Delta}^{(k)}$ arbitrarily, instead he is only allowed to choose a probability distribution $\mathcal{D}^{(k)}$ and sample $\boldsymbol{\Delta}^{(k)}$ from it. Furthermore, it is required that $\mathbb{E}_{\mathcal{D}^{(k)}} \boldsymbol{\Delta} = \mathbf{0}$ and $\|\boldsymbol{\Delta}\|_2^2 \leq c$ for some fixed c and all $\boldsymbol{\Delta} \in \mathcal{D}^{(k)}$. The player does not know $\boldsymbol{x}^{(k)}$ or the distribution $\mathcal{D}^{(k)}$ or the move $\boldsymbol{\Delta}^{(k)}$ of the stochastic adversary. All the player knows is some $\boldsymbol{y}^{(k)} \in \mathbb{R}^n$ that is close to $\boldsymbol{x}^{(k)}$ in ℓ_∞ norm. With this information, the player is allowed to choose one coordinate i and set $x_i^{(k+1)}$ to be zero and for other j, we have $x_j^{(k+1)} = x_j^{(k)} + \Delta_j^{(k)}$.

The question we would like to address is, what strategy the player should choose to keep $\boldsymbol{x}^{(k)}$ close to $\mathbf{0}$ in ℓ_∞ norm? We show that there is a trivial strategy that performs well: simply pick the largest coordinate and set it to 0.

Algorithm 24: Stochastic chasing $\mathbf{0}$ game

Constant: $c > 0, R > 0$.
Let $\boldsymbol{x}^{(1)} = \mathbf{0} \in \mathbb{R}^m$.
for $k = 1$ *to* ∞ **do**
  Stochastic Adversary: Pick $\mathcal{D}^{(k)}$ such that $\mathbb{E}_{\mathcal{D}^{(k)}} \boldsymbol{\Delta} = \mathbf{0}$ and $\|\boldsymbol{\Delta}\|_2 \leq c$ all $\boldsymbol{\Delta} \in \mathcal{D}^{(k)}$.
  Stochastic Adversary: Pick $\boldsymbol{y}^{(k)} \in \mathbb{R}^m$ such that $\|\boldsymbol{y}^{(k)} - \boldsymbol{x}^{(k)}\|_\infty \leq R$.
  Player: Pick a coordinate $i^{(k)}$ using only $\boldsymbol{y}^{(k)}$.
  Sample $\boldsymbol{\Delta}^{(k)}$ from $\mathcal{D}^{(k)}$.
  Set $x_{i^{(k)}}^{(k+1)} = 0$ and $x_j^{(k+1)} = x_j^{(k)} + \Delta_j^{(k)}$ for all $j \neq i^{(k)}$.
end

Theorem 8.4.3. *Using the strategy* $i^{(k)} = \arg\max_i \left|y_i^{(k)}\right|$, *with probability at least* $1 - p$, *we have*

$$\|\boldsymbol{x}^{(k)}\|_\infty \leq 2(c + R) \log\left(4mk^2/p\right)$$

for all k *in the Stochastic Chasing* $\mathbf{0}$ *Game.*

Proof. Consider the potential function $\Phi(\boldsymbol{x}) = \sum_i e^{\alpha x_i} + \sum_i e^{-\alpha x_i}$ where α is to be determined. Now for all x we know that $e^x \leq 1 + x + \frac{x^2}{2} e^{|x|}$ and therefore for all $|\delta| \leq c$, x and α, we have

$$e^{\alpha x + \alpha\delta} \leq e^{\alpha x} + \alpha\delta e^{\alpha x} + \frac{1}{2}\alpha^2\delta^2 e^{\alpha x + |\alpha| c} \quad .$$

Consequently,

$$\mathbb{E}_{\boldsymbol{\Delta} \in \mathcal{D}^{(k)}} \Phi(\boldsymbol{x}^{(k)} + \boldsymbol{\Delta}) \leq \Phi(\boldsymbol{x}^{(k)}) + \alpha \mathbb{E}_{\boldsymbol{\Delta} \in \mathcal{D}^{(k)}} \left(\sum_{i \in [m]} e^{\alpha x_i^{(k)}} \Delta_i - \sum_{i \in [m]} e^{-\alpha x_i^{(k)}} \Delta_i \right)$$
$$+ \frac{\alpha^2}{2} e^{\alpha \|\boldsymbol{\Delta}\|_\infty} \mathbb{E}_{\boldsymbol{\Delta} \in \mathcal{D}^{(k)}} \left(\sum_{i \in [m]} e^{\alpha x_i^{(k)}} \Delta_i^2 + \sum_{i \in [m]} e^{-\alpha x_i^{(k)}} \Delta_i^2 \right) \quad .$$

Since $\mathbb{E}_{\mathcal{D}^{(k)}}\boldsymbol{\Delta} = \mathbf{0}$ and $\|\boldsymbol{\Delta}\|_2 \leq c$, we have $\mathbb{E}_{\boldsymbol{\Delta}\in\mathcal{D}^{(k)}}\left(\sum_i e^{\alpha x_i^{(k)}}\Delta_i - \sum_i e^{-\alpha x_i^{(k)}}\Delta_i\right) = 0$ and

$$\mathbb{E}_{\boldsymbol{\Delta}\in\mathcal{D}^{(k)}}\left(\sum_i e^{\alpha x_i^{(k)}}\Delta_i^2 + \sum_i e^{-\alpha x_i^{(k)}}\Delta_i^2\right) \leq \mathbb{E}_{\boldsymbol{\Delta}\in\mathcal{D}^{(k)}}\left(\sum_i \Delta_i^2\right)\left(\max_i e^{\alpha x_i^{(k)}} + \max_i e^{-\alpha x_i^{(k)}}\right)$$
$$\leq 2c^2 \max_i e^{\alpha\left|x_i^{(k)}\right|} \quad .$$

Letting $\eta^{(k)} = \max_i e^{\alpha\left|x_i^{(k)}\right|}$, we then have

$$\mathbb{E}_{\boldsymbol{\Delta}\in\mathcal{D}^{(k)}}\Phi(\boldsymbol{x}^{(k)} + \boldsymbol{\Delta}) \leq \Phi(\boldsymbol{x}^{(k)}) + \alpha^2 e^{\alpha c} c^2 \eta^{(k)}.$$

Since $i^{(k)} = \arg\max_i \left|y_i^{(k)}\right|$ and $\|\boldsymbol{y}^{(k)} - \boldsymbol{x}^{(k)}\|_\infty \leq R$, the player setting $x_{i^{(k)}}^{(k+1)} = 0$ decreases Φ by at least $e^{-\alpha(R+c)}\eta^{(k)}$. Hence, we have

$$\mathbb{E}_{\boldsymbol{\Delta}\in\mathcal{D}^{(k)}}\Phi(\boldsymbol{x}^{(k+1)}) \quad \leq \quad \Phi(\boldsymbol{x}^{(k)}) + \alpha^2 e^{\alpha c} c^2 \eta^{(k)} - e^{-\alpha(R+c)}\eta^{(k)}.$$

Picking $\alpha = \frac{1}{2(c+R)}$, we have $e^{2\alpha(c+R)}(\alpha(c+R))^2 \leq 1$ and hence $\alpha^2 e^{\alpha c} c^2 \leq e^{-\alpha(R+c)}$. Therefore, we have that

$$\mathbb{E}_{\boldsymbol{\Delta}\in\mathcal{D}^{(k)}}\Phi(\boldsymbol{x}^{(k+1)}) \leq \mathbf{E}\Phi(\boldsymbol{x}^{(k)}) \leq ... \leq \Phi(\boldsymbol{x}^{(1)}) = 2m\,.$$

Consequently, by Markov's inequality we have that $\Pr[\Phi(\boldsymbol{x}^{(k)}) \geq \lambda_k] \leq \frac{2m}{\lambda_k}$ for any λ_k. Furthermore, since clearly $\Phi(\boldsymbol{x}) \geq e^{\alpha\|\boldsymbol{x}\|_\infty}$ we have that $\Pr[\|\boldsymbol{x}^{(k)}\|_\infty \geq \log(\lambda_k)/\alpha] \leq \frac{2m}{\lambda_k}$ for all k. Choosing $\lambda_k = \frac{4mk^2}{p}$ and taking a union bound over all k, we have that

$$\|\boldsymbol{x}^{(k)}\|_\infty \leq 2(c+R)\log\left(4mk^2/p\right)$$

for all k with probability at least

$$1 - \sum_{i=1}^{\infty}\frac{2m}{\lambda_k} = 1 - \sum_{k=1}^{\infty}\frac{p}{2k^2} \geq 1 - p \quad .$$

□

■ 8.5 Glossary

Here we summarize problem specific notation we use throughout this section. For many quantities we included the typical order of magnitude as they appear during our algorithms.

- Our algorithm maintains a polytope $\{\boldsymbol{A}x \geq b\}$ which contains the set of solution K.
- Slacks $\boldsymbol{s}(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}$, $\boldsymbol{S}(\boldsymbol{x})$ is the diagonal matrix corresponds to $\boldsymbol{s}(\boldsymbol{x})$. Rescaled constraint matrix $\boldsymbol{A}_s = \boldsymbol{S}^{-1}\boldsymbol{A}$.
- Leverage Score: $\boldsymbol{\psi}(\boldsymbol{x}) = \boldsymbol{diag}\left(\boldsymbol{A}_x\left(\boldsymbol{A}_x^T\boldsymbol{A}_x + \lambda\boldsymbol{I}\right)^{-1}\boldsymbol{A}_x^T\right)$, $\mu(\boldsymbol{x}) = \min\psi_i(\boldsymbol{x})$, $\lambda = \frac{1}{c_a R^2}$ and $c_a = \frac{1}{10^{10}}$.
- Each iteration, if there is a constraint with leverage score less than $c_d = \frac{1}{10^{12}}$, then we delete that constraint. Otherwise, we add a constraint and put the leverage score c_a.
- Each iteration, our algorithm computes the change of leverage score with accuracy $c_\Delta = \frac{Cc_e}{\log(n\log(R/\varepsilon))}$ where $c_e = \frac{c_d}{4\ln(6nR/\varepsilon)}$ and R is the diameter of the box containing K.

- Potential function $p_{\boldsymbol{e}}(\boldsymbol{x}) = -\sum_{i\in[m]}(c_e + e_i)\log(s_i(\boldsymbol{x})/R) + \frac{1}{2}\log\det\left(R^2\left(\boldsymbol{A}^T\boldsymbol{S}_x^{-2}\boldsymbol{A} + \lambda\boldsymbol{I}\right)\right) + \frac{\lambda}{2}\|x\|_2^2$.
- The algorithm starts with $p_{\boldsymbol{e}}(\boldsymbol{x}) \sim -\Theta(1)n$ and ends when $p_{\boldsymbol{e}}(\boldsymbol{x}) \sim \Theta(1)n\log(nR/\varepsilon)$.
- Centrality $\delta_{\boldsymbol{e}}(\boldsymbol{x}) = \|\nabla p_{\boldsymbol{e}}(\boldsymbol{x})\|_{\boldsymbol{H}(\boldsymbol{x})^{-1}}$ where $\boldsymbol{H}(\boldsymbol{x}) = \boldsymbol{A}_x^T(c_e\boldsymbol{I} + \boldsymbol{\Psi}(\boldsymbol{x}))\boldsymbol{A}_x + \lambda\boldsymbol{I}$. $\delta_{\boldsymbol{e}}(\boldsymbol{x}) \approx \Theta(\sqrt{c_e + \mu(\boldsymbol{x})})$.
- Approximate Hessian $\boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{A}_x^T(c_e\boldsymbol{I} + \boldsymbol{W})\boldsymbol{A}_x + \lambda\boldsymbol{I}$.

Chapter 9

Effective Use of Cutting Plane Method

9.1 Introduction

Cutting plane methods have long been employed to obtain polynomial time algorithms for solving optimization problems. However, for many problems cutting plane methods are often regarded as inefficient both in theory and in practice. Here, in this chapter, we provide several techniques for applying cutting plane methods efficiently. Moreover, we illustrate the efficacy and versatility of these techniques by applying them to achieve improved running times for solving multiple problems including semidefinite programming, matroid intersection, and submodular flow.

We hope these results revive interest in ellipsoid and cutting plane methods. We believe these results demonstrate how cutting plan methods are often useful not just for showing that a problem is solvable in polynomial time, but in many yield substantial running time improvements. We stress that while some results in this chapter are problem-specific, the techniques introduced here are quite general and are applicable to a wide range of problems.

In the remainder of this introduction we survey the key techniques we use to apply our cutting plane method (Section 9.1.1) and the key results we obtain on improving the running time for solving various optimization problems (Section 9.1.2). We conclude in Section 9.1.3 by providing an overview of where to find additional technical result in this chapter.

9.1.1 Techniques

Although cutting plane methods are typically introduced as algorithms for finding a point in a convex set (as we did with the feasibility problem in Chapter 8), this is often not the easiest way to apply the methods. Moreover, improperly applying results on the feasibility problem to solve convex optimization problems can lead to vastly sub-optimal running times. Our central goal, here, in this chapter is to provide tools that allow cutting plane methods to be efficiently applied to solve complex optimization problems. Some of these tools are new and some are extensions of previously known techniques. Here we briefly survey the techniques we cover in Section 9.3 and Section 9.4.

Technique 0: From Feasibility to Optimization

In Section 9.3.1, we explain how to use our cutting plane method to solve convex optimization problems using an approximate subgradient oracle. Our result is based on a result of Nemirovski [202] in which he showed how to use a cutting plane method to solve convex optimization problems without smoothness assumptions on the function and with minimal assumptions on the size of the function's domain. We adopt his proof to accommodate for an approximate separation oracle, an extension which is essential for our applications. We use this result as the starting point for two new techniques we discuss below.

Technique 1: Dimension Reduction through Duality

In Section 9.3.2, we discuss how cutting plane methods can be applied to obtain both primal and dual solutions to convex optimization problems. Moreover, we show how this can be achieved while only applying the cutting plane method in the space, primal or dual, which has a fewer number of variables. Thus we show how to use duality to improve the convergence of cutting plane methods while still solving the original problem.

To illustrate this idea consider the following very simple linear program (LP)

$$\min_{x_i \geq 0, \sum x_i = 1} \sum_{i=1}^{n} w_i x_i$$

where $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{w} \in \mathbb{R}^n$. Although this LP has n variables, it should to be easy to solve purely on the grounds that it only has one equality constraint and thus dual linear program is simply

$$\max_{y \leq w_i \forall i} y\,,$$

i.e. a LP with only one variable. Consequently, we can apply our cutting plane method to solve it efficiently.

However, while this simple example demonstrates how we can use duality to decrease dimensions, it is not always obvious how to recover the optimal primal solution x variable given the optimal dual solution y. Indeed, for many problems their dual is significantly simpler than itself (primal), so some work is required to show that working in the space suffices to require a primal solution.

One such recent example of this approach proving successful is the linear program result explained in Chapter 6. In this result, the authors show how to take advantage of this observation and get a faster LP solver and maximum flow algorithm. It is interesting to study how far this technique can extend, that is, in what settings can one recover the solution to a more difficult dual problem from the solution to its easier primal problem?

There is in fact another precedent for such an approach. Grötschel, Lovász and Schrijver[109] showed how to obtain the primal solution for linear program by using a cutting plane method to solve the linear program exactly. This is based on the observation that cutting plane methods are able to find the active constraints of the optimal solution and hence one can take dual of the linear program to get the dual solution. This idea was further extended in [151] which also observed that cutting plane methods are incrementally building up a LP relaxation of the optimization problem. Hence, one can find a dual solution by taking the dual of that relaxation.

In Section 9.3.2, we provide a fairly general technique to recover a dual optimal solution from an approximately optimal primal solution. Unfortunately, the performance of this technique seems quite problem-dependent. We therefore only analyze this technique for semidefinite programming (SDP), a classic and popular convex optimization problem. As a result, we obtain a faster SDP solver in both the primal and dual formulations of the problem.

Technique 2: Using Optimization Oracles Directly

In the seminal works of Grötschel, Lovász, Schrijver and independently Karp and Papadimitriou [110, 138], they showed the equivalence between optimization oracles and separation oracles, and gave a general method to construct a separation oracle for a convex set given an optimization oracle for that set, that is an oracle for minimizing linear functionals over the set. This seminal result led to the first weakly polynomial time algorithm for many algorithms such as submodular function minimization. Since then, this idea has been used extensively in various settings [127, 44, 45, 65].

Unfortunately, while this equivalence of separation and optimization is a beautiful and powerful tool for polynomial time solvability of problems, in many case it may lead to inefficient algorithms. In order to use this reduction to get a separation oracle, the optimization oracle may need to be called multiple times – essentially the number of times needed to run a cutting plane method and hence may be detrimental to obtaining small asymptotic running times. Therefore, it is an interesting question of whether there is a way of using an optimization oracle more directly.

In Section 9.4 we provide a partial answer to this question for the case of a broad class of problems, that we call the *intersection problem*. For these problems we demonstrate how to achieve running time improvements by using optimization oracles directly. The problem we consider is as follows. We wish to solve the problem for some cost vector $\boldsymbol{c} \in \mathbb{R}^n$ and convex set K. We assume that the convex set K can be decomposed as $K = K_1 \cap K_2$ such that $\max_{\boldsymbol{x} \in K_1} \langle \boldsymbol{c}, \boldsymbol{x} \rangle$ and $\max_{\boldsymbol{x} \in K_2} \langle \boldsymbol{c}, \boldsymbol{x} \rangle$ can each be solved efficiently. Our goal is to obtain a running time for this problem comparable to that of minimizing K given only a separation oracle for it.

We show that by considering a carefully regularized variant, we obtain a problem such that optimization oracles for K_1 and K_2 immediately yield a separation oracle for this regularized problem. By analyzing the regularizer and bounding the domains of the problem we are able to show that this allows us to efficiently compute highly accurate solutions to the intersection problem by applying our cutting plane method once. In other words, we do not need to use a complicated iterative scheme or directly invoke the equivalence between separation and optimization and thereby save $O(\text{poly}(n))$ factors in our running times.

We note that this intersection problem can be viewed as a generalization of the matroid intersection problem and in Section 9.4.2, we show our reduction gives a faster algorithm in certain parameter regimes. As another example, in Section 9.4.3 we show our reduction gives a substantial polynomial improvement for the submodular flow problem. Furthermore, in Section 9.4.4 we show how our techniques allow us to minimize a linear function over the intersection of a convex set and an affine subspace in a number of iterations that depends only on the co-dimension of the affine space.

9.1.2 Applications

Our main goal in this chapter is to provide general techniques for efficiently using cutting plane methods for various problems. Hence, in this chapter, we use minimally problem-specific techniques to achieve the best possible running time. However, we also demonstrate the efficacy of our approach by showing how techniques improve upon the previous best known running times for solve several classic problems in combinatorial and continuous optimization. Here we provide a brief overview of these applications, previous work on these problems, and our results.

In order to avoid deviating from our main discussion, our coverage of previous methods and techniques is brief. Given the large body of prior works on SDP, matroid intersection and submodular flow, it would be impossible to have an in-depth discussion on all of them. Therefore, this section focuses on running time comparisons and explanations of relevant preivous techniques.

Semidefinite Programming

In Section 9.3.2 we consider the classic semidefinite programming (SDP) problem:

$$\max_{\boldsymbol{X} \succeq \boldsymbol{0}} \boldsymbol{C} \bullet \boldsymbol{X} \text{ s.t. } \boldsymbol{A}_i \bullet \boldsymbol{X} = b_i \text{ (primal)} \qquad \min_{\boldsymbol{y}} \boldsymbol{b}^T \boldsymbol{y} \text{ s.t. } \sum_{i=1}^{n} y_i \boldsymbol{A}_i \succeq \boldsymbol{C} \text{ (dual)}$$

where $\boldsymbol{X}$, $\boldsymbol{C}$, $\boldsymbol{A}_i$ are $m \times m$ symmetric matrices, $\boldsymbol{b}, \boldsymbol{y} \in \mathbb{R}^n$, and $\boldsymbol{A} \bullet \boldsymbol{B} \stackrel{\text{def}}{=} \text{Tr}(\mathbf{A}^T \mathbf{B})$. For many problems, $n \ll m^2$ and hence the dual problem has fewer variables than the primal. There are many

Years	Authors	Running times
1992	Nesterov, Nemirovsky[210]	$\tilde{O}(\sqrt{n}(nm^\omega + n^{\omega-1}m^2))$
2000	Anstreicher [13]	$\tilde{O}((mn)^{1/4}(nm^\omega + n^{\omega-1}m^2))$
2003	Krishnan, Mitchell [152]	$\tilde{O}(m(n^\omega + m^\omega + S))$ (dual SDP)
-	This Chapter	$\tilde{O}(m(n^2 + m^\omega + S))$

Table 9.1: Previous algorithms for solving a $n \times n$ SDP with m constraints and S non-zeros entries

results and applications of SDP; see [259, 248, 196] for a survey on this topic. Since our focus is on polynomial time algorithms, we do not discuss pseudo-polynomial algorithms such as the spectral bundle method [115], multiplicative weight update methods [17, 19, 126, 8], etc.

Currently, there are two competing approaches for solving SDP problems, namely interior point methods (IPM) and cutting plane methods. Typically, IPMs require fewer iterations than the cutting plane methods, however each iteration of these methods is more complicated and possibly more computationally expensive. For SDP problems, interior point methods require the computations of the Hessian of the function $-\log\det(\boldsymbol{C} - \sum_{i=1}^n y_i \boldsymbol{A}_i)$ whereas cutting plane methods usually only need to compute minimum eigenvectors of the slack matrix $\boldsymbol{C} - \sum_{i=1}^n y_i \boldsymbol{A}_i$.

In [13], Anstreicher provided the current fastest IPM for solving the dual SDP problem using a method based on the volumetric barrier function. This method takes $O((mn)^{1/4})$ iterations and each iteration is as cheap as usual IPMs. For general matrices $\boldsymbol{C}, \boldsymbol{X}, \boldsymbol{A}_i$, each iteration takes $O(nm^\omega + n^{\omega-1}m^2)$ time where ω is the fast matrix multiplication exponent. If the constraint matrices $\boldsymbol{A}_i$ are rank one matrices, the iteration cost can be improved to $O(m^\omega + nm^2 + n^2 m)$ [153]. If the matrices are sparse, then [97, 197] show how to use matrix completion inside the IPM. However, the running time depends on the extended sparsity patterns which can be much larger than the total number of non-zeros.

In [152], Krishnan and Mitchell observed that the separation oracle for dual SDP takes only $O(m^\omega + S)$ time, where $S = \sum_{i=1}^n \mathrm{nnz}(\boldsymbol{A}_i)$ be the total number of non-zeros in the constant matrix. Hence, the cutting plane method by [253] gives a faster algorithm for SDP for many regimes. For $\omega = 2.38$, the cutting plane method is faster when $\boldsymbol{A}_i$ is not rank 1 and the problem is not too dense, i.e. $\sum_{i=1}^n \mathrm{nnz}(\boldsymbol{A}_i) < n^{0.63}m^{2.25}$. While there are previous methods for using cutting plane methods to obtain primal solutions[151] , to the best of our knowledge, there are no worst case running time analysis for these techniques.

In Section 9.3.2, show how to alleviate this issue. We provide an improved algorithm for finding the dual solution and prove carefully how to obtain a comparable primal solution as well. See Figure 9.1 for a summary of the algorithms for SDP and their running times.

Matroid Intersection

In Section 9.4.2 we show how our optimization oracle technique can be used to improve upon the previous best known running times for matroid intersection. Matroid intersection is one of the most fundamental problems in combinatorial optimization. The first algorithm for matroid intersection is due to the seminal paper by Edmonds [81]. In Figures 9.2 and 9.3 we provide a summary of the previous algorithms for unweighted and weighted matroid intersection as well as the new running times we obtain in this chapter. While there is no total ordering on the running times of these algorithms due to the different dependence on various parameters, we would like to point out that our algorithms outperform the previous ones in regimes where r is close to n and/or the oracle query costs are relatively expensive. In particular, in terms of oracle query complexity our algorithms are the first to

Years	Authors	Running times
1968	Edmonds [81]	not stated
1971	Aigner, Dowling [6]	$O(nr^2\mathcal{T}_{\text{ind}})$
1974	Tomizawa, Iri [249]	not stated
1975	Lawler [157]	$O(nr^2\mathcal{T}_{\text{ind}})$
1979	Edmonds [80]	not stated
1986	Cunningham [60]	$O(nr^{1.5}\mathcal{T}_{\text{ind}})$
-	This Chapter	$O(n^2 \log n\mathcal{T}_{\text{ind}} + n^3 \log^{O(1)} n)$ $O(nr \log^2 n\mathcal{T}_{\text{rank}} + n^3 \log^{O(1)} n)$

Table 9.2: Previous algorithms for (unweighted) matroid intersection. Here n is the size of the ground set, $r = \max\{r_1, r_2\}$ is the maximum rank of the two matroids, $\mathcal{T}_{\text{ind}}$ is the time needed to check if a set is independent (independence oracle), and $\mathcal{T}_{\text{rank}}$ is the time needed to compute the rank of a given set (rank oracle).

Years	Authors	Running times
1968	Edmonds [81]	not stated
1974	Tomizawa, Iri [249]	not stated
1975	Lawler [157]	$O(nr^2\mathcal{T}_{\text{ind}} + nr^3)$
1979	Edmonds [80]	not stated
1981	Frank [90]	$O(n^2r(\mathcal{T}_{\text{circuit}} + n))$
1983	Orlin, Ahuja [219]	not stated
1986	Brezovec, Cornuéjols, Glover[39]	$O(nr(\mathcal{T}_{\text{circuit}} + r + \log n))$
1995	Fujishige, Zhang [96]	$O(n^2r^{0.5} \log rM \cdot \mathcal{T}_{\text{ind}})$
1995	Shigeno, Iwata [234]	$O((n + \mathcal{T}_{\text{circuit}})nr^{0.5} \log rM)$
-	This Chapter	$O((n^2 \log n\mathcal{T}_{\text{ind}} + n^3 \log^{O(1)} n) \log nM)$ $O((nr \log^2 n\mathcal{T}_{\text{rank}} + n^3 \log^{O(1)} n) \log nM)$

Table 9.3: Previous algorithms for weighted matroid intersection. In additions to the notations used in the unweighted table, $\mathcal{T}_{\text{circuit}}$ is the time needed to find a fundamental circuit and M is the bit complexity of the weights.

achieve the quadratic bounds of $\tilde{O}(n^2)$ and $\tilde{O}(nr)$ for independence and rank oracles. We hope our work will revive the interest in the problem of which progress has been mostly stagnated for the past 20-30 years.

Minimum-Cost Submodular Flow

In Section 9.4.3 we show how our optimization oracle technique can be used to improve upon the previous best known running times for (Minimum-cost) Submodular Flow. Submodular flow is a very general problem in combinatorial optimization which generalizes many problems such as minimum cost flow, the graph orientation, polymatroid intersection, directed cut covering [94]. In Figure 9.4 we provide an overview of the previous algorithms for submodular flow as well as the new running times we obtain in this chapter.

Many of the running times are in terms of a parameter h, which is the time required for computing an "exchange capacity". To the best of our knowledge, the most efficient way of computing an exchange capacity is to solve an instance of submodular minimization which previously took time $\tilde{O}(n^4\text{EO}+n^5)$ (and now takes $\tilde{O}(n^2\text{EO} + n^3)$ time using our result in Chapter 10). Readers may wish to substitute $h = \tilde{O}(n^2\text{EO} + n^3)$ when reading the table.

Years	Authors	Running times
1978	Fujishige [92]	not stated
1981	Grötschel, Lovász, Schrijver[110]	weakly polynomial
1982	Zimmermann [276]	not stated
1984	Barahona, Cunningham [30]	not stated
1985	Cunningham, Frank [62]	$\rightarrow O(n^4 h \log C)$
1987	Fujishige [93]	not stated
1987	Frank, Tardos [91]	strongly polynomial
1988	Cui, Fujishige [265]	not stated
1989	Fujishige, Röck, Zimmermann[95]	$\rightarrow O(n^6 h \log n)$
1991	Chung, Tcha [52]	not stated
1992	Zimmermann [277]	not stated
1993	McCormick, Ervolina [190]	$O(n^7 h^* \log nCU)$
1994	Wallacher, Zimmermann [266]	$O(n^8 h \log nCU)$
1997	Iwata [117]	$O(n^7 h \log U)$
1998	Iwata, McCormick, Shigeno [123]	$O\left(n^4 h \min\left\{\log nC, n^2 \log n\right\}\right)$
1999	Iwata, McCormick, Shigeno [124]	$O\left(n^6 h \min\left\{\log nU, n^2 \log n\right\}\right)$
1999	Fleischer, Iwata, McCormick[89]	$O\left(n^4 h \min\left\{\log U, n^2 \log n\right\}\right)$
1999	Iwata, McCormick, Shigeno [122]	$O\left(n^4 h \min\left\{\log C, n^2 \log n\right\}\right)$
2000	Fleischer, Iwata [88]	$O(mn^5 \log nU \cdot \mathrm{EO})$
-	This Chapter	$O(n^2 \log nCU \cdot \mathrm{EO} + n^3 \log^{O(1)} nCU)$

Figure 9-1: Previous algorithms for Submodular Flow with n vertices, maximum cost C and maximum capacity U. The factor h is the time for an exchange capacity oracle, h^* is the time for a "more complicated exchange capacity oracle" and EO is the time for evaluation oracle of the submodular function. The arrow,$\rightarrow$, indicates that it used currently best maximum submodular flow algorithm as subroutine which was non-existent at the time of the publication.

The previous fastest weakly polynomial algorithms for submodular flow are by [122, 88, 89], which take time $\tilde{O}(n^6\mathrm{EO} + n^7)$ and $O(mn^5 \log nU \cdot \mathrm{EO})$, assuming $h = \tilde{O}(n^2\mathrm{EO} + n^3)$. Our algorithm for submodular flow has a running time of $\tilde{O}(n^2\mathrm{EO} + n^3)$, which is significantly faster by roughly a factor of $\tilde{O}(n^4)$.

For strongly polynomial algorithms, our results do not yield a speedup but we remark that our faster strongly polynomial algorithm for submodular minimization in Chapter 10 improves the previous algorithms by a factor of $\tilde{O}(n^2)$ as a corollary (because h requires solving an instance of submodular minimization).

9.1.3 Overview

After providing covering some preliminaries on convex analysis in Section 9.2 we split the remainder of Chapter 9 into Section 9.3 and Section 9.4. In Section 9.3 we cover our algorithm for convex optimization using an approximate subgradient oracle (Section 9.3.1) as well as our technique on using duality to decrease dimensions and improve the running time of semidefinite programming (Section 9.3.2). In Section 9.4 we provide our technique for using minimization oracles to minimize functions over the intersection of convex sets and provide several applications including matroid intersection (Section 9.4.2), submodular flow (Section 9.4.3), and minimizing a linear function over the intersection of an affine subspace and a convex set (Section 9.4.4).

9.2 Preliminaries

In this section we review basic facts about convex functions that we use throughout Chapter 9. We also introduce two oracles that we use throughout Chapter 9, i.e. subgradient and optimization oracles, and provide some basic reductions between them. Note that we have slightly extended some definitions and facts to accommodate for the noisy separation oracles used in this chapter.

First we recall the definition of strong convexity

Definition 9.2.1 (Strong Convexity). A real valued function f on a convex set Ω is α-strongly convex if for any $\boldsymbol{x}, \boldsymbol{y} \in \Omega$ and $t \in [0,1]$, we have

$$f(t\boldsymbol{x} + (1-t)\boldsymbol{y}) + \frac{1}{2}\alpha t(1-t)\|\boldsymbol{x}-\boldsymbol{y}\|^2 \leq tf(\boldsymbol{x}) + (1-t)f(\boldsymbol{y}).$$

Next we define an approximate subgradient.

Definition 9.2.2 (Subgradient)**.** For any convex function f on a convex set Ω, the δ-subgradients of f at x are defined to be

$$\partial_\delta f(\boldsymbol{x}) \stackrel{\text{def}}{=} \{\boldsymbol{g} \in \Omega : \ f(\boldsymbol{y}) + \delta \geq f(\boldsymbol{x}) + \langle \boldsymbol{g}, \boldsymbol{y}-\boldsymbol{x}\rangle \ \text{ for all } \boldsymbol{y} \in \Omega\}.$$

Here we provide some basic facts regarding convexity and subgradients. These statements are natural extensions of well known facts regarding convex functions and their proof can be found in any standard textbook on convex optimization.

Fact 9.2.3. *For any convex set Ω and $\boldsymbol{x}$ be a point in the interior of Ω, we have the following:*

1. *If f is convex on Ω, then $\partial_0 f(\boldsymbol{x}) \neq \emptyset$ and $\partial_s f(\boldsymbol{x}) \subseteq \partial_t f(\boldsymbol{x})$ for all $0 \leq s \leq t$. Otherwise, we have $\|\boldsymbol{g}\|_2 > \frac{1}{2}\sqrt{\frac{\delta}{D}}$. For any $f(\boldsymbol{y}) \leq f(\boldsymbol{x})$, we have $\delta \geq \langle \boldsymbol{g}, \boldsymbol{y}-\boldsymbol{x}\rangle$ and hence*
2. *If f is a differential convex function on Ω, then $\nabla f(\boldsymbol{x}) \in \partial_0 f(\boldsymbol{x})$.*
3. *If f_1 and f_2 are convex function on Ω, $\boldsymbol{g}_1 \in \partial_{\delta_1} f_1(\boldsymbol{x})$ and $\boldsymbol{g}_2 \in \partial_{\delta_2} f_1(\boldsymbol{x})$, then $\alpha\boldsymbol{g}_1 + \beta\boldsymbol{g}_2 \in \partial_{\alpha\delta_1+\beta\delta_2}(\boldsymbol{g}_1+\boldsymbol{g}_2)(\boldsymbol{x})$.*
4. *If f is α-strongly convex on Ω with minimizer x^*, then for any $\boldsymbol{y}$ with $f(\boldsymbol{y}) \leq f(\boldsymbol{x}^*) + \varepsilon$, we have $\frac{1}{2}\alpha\|\boldsymbol{x}^* - \boldsymbol{y}\|^2 \leq \varepsilon$.*

Next we provide a reduction from subgradients to separation oracles. We will use this reduction several times in Chapter 9 to simplify our construction of separation oracles.

Lemma 9.2.4. *Let f be a convex function. Suppose we have $\boldsymbol{x}$ and $\boldsymbol{g} \in \partial_\delta f(\boldsymbol{x})$ with $\|\boldsymbol{x}\|_2 \leq 1 \leq D$ and $\delta \leq 1$. If $\|\boldsymbol{g}\|_2 \leq \frac{1}{2}\sqrt{\frac{\delta}{D}}$, then $f(\boldsymbol{x}) \leq \min_{\|\boldsymbol{y}_2\|_2 \leq D} f(\boldsymbol{y}) + 2\sqrt{\delta D}$ and if $\|\boldsymbol{g}\|_2 \leq \frac{1}{2}\sqrt{\frac{\delta}{D}}$ then*

$$\{\|\boldsymbol{y}\|_2 \leq D \ : \ f(\boldsymbol{y}) \leq f(\boldsymbol{x})\} \subset \{\boldsymbol{y} : \boldsymbol{d}^T\boldsymbol{y} \leq \boldsymbol{d}^T\boldsymbol{x} + 2\sqrt{\delta D}\}$$

with $\boldsymbol{d} = \boldsymbol{g}/\|\boldsymbol{g}\|_2$. Hence, this gives a $(2\sqrt{\delta D}, 2\sqrt{\delta D})$-separation oracle on the set $\{\|\boldsymbol{x}\|_2 \leq D\}$.

Proof. Let $\boldsymbol{y}$ such that $\|\boldsymbol{y}\|_2 \leq D$. By the definition of δ-subgradient, we have

$$f(\boldsymbol{y}) + \delta \geq f(\boldsymbol{x}) + \langle \boldsymbol{g}, \boldsymbol{y}-\boldsymbol{x}\rangle .$$

If $\|\boldsymbol{g}\| \leq \frac{1}{2}\sqrt{\frac{\delta}{D}}$, then, we have $|\langle \boldsymbol{g}, \boldsymbol{y} - \boldsymbol{x}\rangle| \leq \sqrt{\delta D}$ because $\|\boldsymbol{x}\| \leq D$ and $\|\boldsymbol{y}\|_2 \leq D$. Therefore,

$$\min_{\|\boldsymbol{y}\|_2 \leq D} f(\boldsymbol{y}) + 2\sqrt{\delta D} \geq f(\boldsymbol{x}).$$

Otherwise, we have $\|\boldsymbol{g}\|_2 > \frac{1}{2}\sqrt{\frac{\delta}{D}}$. For any $f(\boldsymbol{y}) \leq f(\boldsymbol{x})$, we have $\delta \geq \langle \boldsymbol{g}, \boldsymbol{y} - \boldsymbol{x}\rangle$ and hence

$$2\sqrt{\delta D} \geq \left\langle \frac{\boldsymbol{g}}{\|\boldsymbol{g}\|}, \boldsymbol{y} - \boldsymbol{x} \right\rangle.$$

□

At several times in Chapter 9 we will wish to construct subgradient oracles or separation oracles given only the ability to approximately maximize a linear function over a convex set. In the remainder of this section we formally define such a *optimization oracle* and prove this equivalence.

Definition 9.2.5 (Optimization Oracle)**.** Given a convex set K and $\delta > 0$ a δ-optimization oracle for K is a function on $\mathbb{R}^n$ such that for any input $\boldsymbol{c} \in \mathbb{R}^n$, it outputs $\boldsymbol{y}$ such that

$$\max_{\boldsymbol{x} \in K} \langle \boldsymbol{c}, \boldsymbol{x}\rangle \leq \langle \boldsymbol{c}, \boldsymbol{y}\rangle + \delta.$$

We denote by $\mathrm{OO}_\delta(K)$ the time complexity of this oracle.

Lemma 9.2.6. *Given a convex set K, any ε-optimization oracle for K is a ε-subgradient oracle for $f(\boldsymbol{c}) \stackrel{\text{def}}{=} \max_{\boldsymbol{x} \in K} \langle \boldsymbol{c}, \boldsymbol{x}\rangle$.*

Proof. Let $\boldsymbol{x}_c$ be the output of ε-optimization oracle on the cost vector $\boldsymbol{c}$. We have

$$\max_{\boldsymbol{x} \in K} \langle \boldsymbol{c}, \boldsymbol{x}\rangle \leq \langle \boldsymbol{c}, \boldsymbol{x}_c\rangle + \varepsilon.$$

Hence, for all $\boldsymbol{d}$, we have and therefore

$$\langle \boldsymbol{x}_c, \boldsymbol{d} - \boldsymbol{c}\rangle + f(\boldsymbol{c}) \leq f(\boldsymbol{d}) + \varepsilon.$$

Hence, $\boldsymbol{x}_c \in \partial_\delta f(\boldsymbol{c})$. □

Combining these lemmas shows that having an ε-optimization oracle for a convex set K contained in a ball of radius D yields a $O(\sqrt{D\varepsilon}, \sqrt{D\varepsilon})$ separation oracle for $\max_{x \in K}\langle \boldsymbol{c}, \boldsymbol{x}\rangle$. We use these ideas to construction separation oracles throughout Chapter 9.

9.3 Convex Optimization

In this section we show how to apply our cutting plane method to efficiently solve problems in convex optimization. First, in Section 9.3.1 we show how to use our result to minimize a convex function given an approximate subgradient oracle. Then, in Section 9.3.2 we illustrate how this result can be used to obtain both primal and dual solutions for a standard convex optimization problems. In particular, we show how our result can be used to obtain improved running times for semidefinite programming across a range of parameters.

9.3.1 From Feasibility to Optimization

In this section we consider the following standard optimization problem. We are given a convex function $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ and we want to find a point $\boldsymbol{x}$ that approximately solves the minimization

problem

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x})$$

given only a subgradient oracle for f.

Here we show how to apply the cutting plane method from Chapter 8 turning the small width guarantee of the output of that algorithm into a tool to find an approximate minimizer of f. Our result is applicable to any convex optimization problem armed with a separation or subgradient oracle. This result will serve as the foundation for many of our applications in this chapter.

Our approach is an adaptation of Nemiroski's method [202] which applies the cutting plane method to solve convex optimiziation problems, with only minimal assumption on the cutting plane method. The proof here is a generalization that accommodates for the noisy separation oracle. In the remainder of this subsection we provide a key definition we will use in our algorithm (Defintion 9.3.1), provide our main result (Theorem 9.3.2), and conclude with a brief discussion of this result.

Definition 9.3.1. For any compact set K, we define the *minimum width* by $\text{MinWidth}(K) \stackrel{\text{def}}{=} \min_{\|\boldsymbol{a}\|_2=1} \max_{\boldsymbol{x},\boldsymbol{y}\in K} \langle \boldsymbol{a}, \boldsymbol{x}-\boldsymbol{y}\rangle$.

Theorem 9.3.2. *Let f be a convex function on $\mathbb{R}^n$ and Ω be a convex set that contains a minimizer of f. Suppose we have a (η,δ)-separation oracle for f and Ω is contained inside $B_\infty(R)$. Using $B_\infty(R)$ as the initial polytope for our Cutting Plane Method, for any $0<\alpha<1$, we can compute $\boldsymbol{x}\in\mathbb{R}^n$ such that*

$$f(\boldsymbol{x}) - \min_{\boldsymbol{y}\in\Omega} f(\boldsymbol{y}) \le \eta + \alpha\left(\max_{\boldsymbol{y}\in\Omega} f(\boldsymbol{y}) - \min_{\boldsymbol{y}\in\Omega} f(\boldsymbol{y})\right) \quad . \tag{9.1}$$

with an expected running time of

$$O\left(n\text{SO}_{\eta,\delta}(f)\log\left(\frac{n\kappa}{\alpha}\right) + n^3\log^{O(1)}\left(\frac{n\kappa}{\alpha}\right)\right),$$

where $\delta = \Theta\left(\frac{\alpha\text{MinWidth}(\Omega)}{n^{3/2}\ln(\kappa)}\right)$ and $\kappa = \frac{R}{\text{MinWidth}(\Omega)}$. Furthermore, we only need the oracle defined on the set $B_\infty(R)$.

Proof. Let $\boldsymbol{x}^* \in \arg\min_{\boldsymbol{x}\in\Omega} f(\boldsymbol{x})$. Since $B_\infty(R)\supset\Omega$ contains a minimizer of f, by the definition of (η,δ)-separation oracles, our Cutting Plane Method (Theorem 8.3.26) either returns a point $\boldsymbol{x}$ that is almost optimal or returns a polytope P of small width. In the former case we have a point $\boldsymbol{x}$ such that $f(\boldsymbol{x}) \le \min_{\boldsymbol{y}} f(\boldsymbol{y}) + \eta$. Hence, the error is clearly at most $\eta + \alpha(\max_{\boldsymbol{z}\in\Omega} f(\boldsymbol{z}) - \min_{\boldsymbol{x}\in\Omega} f(\boldsymbol{x}))$ as desired. Consequently, we assume the latter case.

Theorem 8.3.26 shows $\text{MinWidth}(P) < Cn\varepsilon\ln(R/\varepsilon)$ for some universal constant C. Picking

$$\varepsilon = C'\frac{\alpha\text{MinWidth}(\Omega)}{n\ln\left(\frac{n\kappa}{\alpha}\right)} \tag{9.2}$$

for small enough constant C', we have $\text{MinWidth}(P^{(i)}) < \alpha\text{MinWidth}(\Omega)$. Let $\Omega^\alpha = \boldsymbol{x}^* + \alpha(\Omega - \boldsymbol{x}^*)$, namely, $\Omega^\alpha = \{\boldsymbol{x}^* + \alpha(\boldsymbol{z}-\boldsymbol{x}^*) : \boldsymbol{z}\in\Omega\}$. Then, we have

$$\text{MinWidth}(\Omega^\alpha) = \alpha\text{MinWidth}(\Omega) > \text{MinWidth}(P).$$

Therefore, Ω^α is not a subset of $P^{(i)}$ and hence there is some point $\boldsymbol{y}\in\Omega^\alpha\backslash P$. Since $\Omega^\alpha\subseteq\Omega\subseteq B_\infty(R)$, we know that $\boldsymbol{y}$ does not violate any of the constraints of $P^{(0)}$ and therefore must violate one of the constraints added by querying the separation oracle. Therefore, for some $j\le i$, we have

$$\left\langle \boldsymbol{c}^{(j-1)}, \boldsymbol{y}\right\rangle > \left\langle \boldsymbol{c}^{(j-1)}, \boldsymbol{x}^{(j-1)}\right\rangle + c_s\varepsilon/\sqrt{n} \quad .$$

By the definition of $(\eta, c_s\varepsilon/\sqrt{n})$-separation oracle (Definition 2.3.6), we have $f(\boldsymbol{y}) > f(\boldsymbol{x}^{(j-1)})$. Since $\boldsymbol{y} \in \Omega^\alpha$, we have $\boldsymbol{y} = (1-\alpha)\boldsymbol{x}^* + \alpha\boldsymbol{z}$ for some $\boldsymbol{z} \in \Omega$. Thus, the convexity of f implies that

$$f(\boldsymbol{y}) \leq (1-\alpha)f(\boldsymbol{x}^*) + \alpha f(\boldsymbol{z}).$$

Therefore, we have

$$\min_{1\leq k\leq i} f(\boldsymbol{x}^{(k)}) - \min_{\boldsymbol{x}\in\Omega} f(\boldsymbol{x}) < f(\boldsymbol{y}) - f(\boldsymbol{x}^*) \leq \alpha\left(\max_{\boldsymbol{z}\in\Omega} f(\boldsymbol{z}) - \min_{\boldsymbol{x}\in\Omega} f(\boldsymbol{x})\right).$$

Hence, we can simply output the best $\boldsymbol{x}$ among all $\boldsymbol{x}^{(j)}$ and in either case $\boldsymbol{x}$ satisfies (9.1).

Note that we need to call (η, δ)-separation oracle with $\delta = \Omega(\varepsilon/\sqrt{n})$ to ensure we do not cut out $\boldsymbol{x}^*$. Theorem 8.3.26 shows that the algorithm takes $O(n\mathrm{SO}_{\eta,\delta}(f)\log(nR/\varepsilon) + n^3\log^{O(1)}(nR/\varepsilon))$ expected time, as promised. Furthermore, the oracle needs only be defined on $B_\infty(R)$ as our cutting plane method guarantees $\boldsymbol{x}^{(k)} \in B_\infty(R)$ for all k (although if needed, an obvious separating hyperplane can be returned for a query point outside $B_\infty(R)$). □

Observe that this algorithm requires no information about Ω (other than that $\Omega \subseteq B_\infty(R)$) and does not guarantee that the output is in Ω. Hence, even though Ω can be complicated to describe, the algorithm still gives a guarantee related to the gap $\max_{\boldsymbol{x}\in\Omega} f(\boldsymbol{x}) - \min_{\boldsymbol{x}\in\Omega} f(\boldsymbol{x})$. For specific applications, it is therefore advantageous to pick a Ω as large as possible while the bound on function value is as small as possible.

Before indulging into specific applications, we remark on the dependence on κ. Using John's ellipsoid, it can be shown that any convex set Ω can be transformed linearly such that (1) $B_\infty(1)$ contains Ω and, (2) $\mathrm{MinWidth}(\Omega) = \Omega(n^{-3/2})$. In other words, κ can be effectively chosen as $O(n^{3/2})$. Therefore if we are able to find such a linear transformation, the running time is simply $O\left(n\mathrm{SO}(f)\log(n/\alpha) + n^3\log^{O(1)}(n/\alpha)\right)$. Often this can be done easily using the structure of the particular problem and the running time does not depend on the size of domain at all.

9.3.2 Duality and Semidefinite Programming

In this section we illustrate how our result in Section 9.3.1 can be used to obtain both primal and dual solutions for standard problems in convex optimization. In particular we show how to obtain improved running times for semidefinite programming.

To explain our approach, consider the following minimax problem

$$\min_{\boldsymbol{y}\in Y}\max_{\boldsymbol{x}\in X} \langle \boldsymbol{A}\boldsymbol{x}, \boldsymbol{y}\rangle + \langle \boldsymbol{c}, \boldsymbol{x}\rangle + \langle \boldsymbol{d}, \boldsymbol{y}\rangle \tag{9.3}$$

where $\boldsymbol{x} \in \mathbb{R}^m$ and $\boldsymbol{y} \in \mathbb{R}^n$. When $m \gg n$, solving this problem by directly using Chapter 8 could lead to an inefficient algorithm with running time at least m^3. In many situations, for any fixed $\boldsymbol{y}$, the problem $\max_{\boldsymbol{x}\in X}\langle \boldsymbol{A}\boldsymbol{x}, \boldsymbol{y}\rangle$ is very easy and hence one can use it as a separation oracle and apply Chapter 8 and this would gives a running time almost independent of m. However, this would only give us the $\boldsymbol{y}$ variable and it is not clear how to recover $\boldsymbol{x}$ variable from it.

In this section we show how to alleviate this issue and give semidefinite programming (SDP) as a concrete example of how to apply this general technique. We do not write down the general version as the running time of the technique seems to be problem specific and faster SDP is already an interesting application.

For the remainder of this section we focus on the semidefinite programming (SDP) problem:

$$\max_{\boldsymbol{X}\succeq\boldsymbol{0}} \boldsymbol{C}\bullet\boldsymbol{X} \text{ s.t. } \boldsymbol{A}_i\bullet\boldsymbol{X} = b_i \tag{9.4}$$

and its dual

$$\min_{\boldsymbol{y}} \boldsymbol{b}^T \boldsymbol{y} \text{ s.t. } \sum_{i=1}^{n} y_i \boldsymbol{A}_i \succeq \boldsymbol{C} \tag{9.5}$$

where $\boldsymbol{X}$, $\boldsymbol{C}$, $\boldsymbol{A}_i$ are $m \times m$ symmetric matrices and $\boldsymbol{b}, \boldsymbol{y} \in \mathbb{R}^n$. Our approach is partially inspired by one of the key ideas of [115, 152]. These results write down the dual SDP in the form

$$\min_{y} \boldsymbol{b}^T \boldsymbol{y} - K \min(\lambda_{\min}(\sum_{i=1}^{n} y_i \boldsymbol{A}_i - \boldsymbol{C}), 0) \tag{9.6}$$

for some large number K and use non-smooth optimization techniques to solve the dual SDP problem. Here, we follow the same approach but instead write it as a max-min problem $\min_{\boldsymbol{y}} f_K(\boldsymbol{y})$ where

$$f_K(\boldsymbol{y}) = \max_{\mathrm{Tr}\boldsymbol{X} \leq K, \boldsymbol{X} \succeq \boldsymbol{0}} \left(\boldsymbol{b}^T \boldsymbol{y} + \left\langle \boldsymbol{X}, \boldsymbol{C} - \sum_{i=1}^{n} y_i \boldsymbol{A}_i \right\rangle \right). \tag{9.7}$$

Thus the SDP problem in fact assumes the form (9.3) and many ideas in this section can be generalized to the minimax problem (9.3).

To get a dual solution, we notice that the cutting plane method maintains a subset of the primal feasible solution $\mathrm{conv}(\boldsymbol{X}_i)$ such that

$$\min_{\boldsymbol{y}} \boldsymbol{b}^T \boldsymbol{y} + \max_{\mathrm{Tr}\boldsymbol{X} \leq K, \boldsymbol{X} \succeq \boldsymbol{0}} \left\langle \boldsymbol{X}, \boldsymbol{C} - \sum_{i=1}^{n} y_i \boldsymbol{A}_i \right\rangle \sim \min_{\boldsymbol{y}} \boldsymbol{b}^T \boldsymbol{y} + \max_{\boldsymbol{X} \in \mathrm{conv}(\boldsymbol{X}_i)} \left\langle \boldsymbol{X}, \boldsymbol{C} - \sum_{i=1}^{n} y_i \boldsymbol{A}_i \right\rangle.$$

Applying minimax theorem, this shows that there exists an approximation solution $\boldsymbol{X}$ in $\mathrm{conv}(\boldsymbol{X}_i)$ for the primal problem. Hence, we can restrict the primal SDP on the polytope $\mathrm{conv}(\boldsymbol{X}_i)$, this reduces the primal SDP into a linear program which can be solved very efficiently. This idea of getting primal/dual solution from the cutting plane method is quite general and is the main purpose of this example. As a by-product, we have a faster SDP solver in both primal and dual! We remark that this idea has been used as a heuristic to obtain [151] for getting the primal SDP solution and our contribution here is mainly the asymptotic time analysis.

We first show how to construct the separation oracle for SDP. For that we need to compute smallest eigenvector of a matrix. Below, for completeness we provide a folklore result showing we can do this using fast matrix multiplication.

Lemma 9.3.3. *Given a $n \times n$ symmetric matrix $\boldsymbol{Y}$ such that $-R\boldsymbol{I} \preceq \boldsymbol{Y} \preceq R\boldsymbol{I}$, for any $\varepsilon > 0$, with high probability in n in time $O(n^{\omega+o(1)} \log^{O(1)}(R/\varepsilon))$ we can find a unit vector $\boldsymbol{u}$ such that $\boldsymbol{u}^T \boldsymbol{Y} \boldsymbol{u} \geq \lambda_{\max}(\boldsymbol{Y}) - \varepsilon$.*

Proof. Let $\boldsymbol{B} \stackrel{\text{def}}{=} \frac{1}{R}\boldsymbol{Y} + \boldsymbol{I}$. Note that $\boldsymbol{B} \succeq \boldsymbol{0}$. Now, we consider the repeated squaring $\boldsymbol{B}_0 = \boldsymbol{B}$ and $\boldsymbol{B}_{k+1} = \frac{\boldsymbol{B}_k^2}{\mathrm{Tr}\boldsymbol{B}_k^2}$. Let $0 \leq \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ be the eigenvalues of $\boldsymbol{B}$ and $\boldsymbol{v}_i$ be the corresponding eigenvectors. Then, it is easy to see the the eigenvalues of $\boldsymbol{B}_k$ are $\frac{\lambda_i^{2^k}}{\sum_{i=1}^n \lambda_i^{2^k}}$.

Let $\boldsymbol{q}$ be a random unit vector and $\boldsymbol{r} \stackrel{\text{def}}{=} \boldsymbol{B}_k \boldsymbol{q}$. Now $\boldsymbol{q} = \sum \alpha_i \boldsymbol{v}_i$ for some α_i such that $\sum \alpha_i^2 = 1$. Letting

$$\boldsymbol{p} = \frac{\sum_{\lambda_i > (1-\delta)\lambda_n} \alpha_i \lambda_i^{2^k} \boldsymbol{v}_i}{\sum_{i=1}^n \lambda_i^{2^k}}$$

we have

$$\|\boldsymbol{r}-\boldsymbol{p}\|_2 = \left\|\frac{\sum_{\lambda_i\le(1-\delta)\lambda_n}\alpha_i\lambda_i^{2^k}\boldsymbol{v}_i}{\sum_{i=1}^n\lambda_i^{2^k}}\right\|_2 \le \frac{\sum_{\lambda_i\le(1-\delta)\lambda_n}\lambda_i^{2^k}}{\sum_{i=1}^n\lambda_i^{2^k}} \le (1-\delta)^{2^k}n.$$

Letting $k=\log_2\left(\frac{\log(n^{3/2}/\delta)}{\delta}\right)$, we have $\|\boldsymbol{r}-\boldsymbol{p}\|_2\le\delta/\sqrt{n}$. Since $\mathbf{0}\preceq\boldsymbol{B}\preceq 2\boldsymbol{I}$, we have

$$\begin{aligned}\sqrt{\boldsymbol{r}^T\boldsymbol{B}\boldsymbol{r}} &\ge \sqrt{\boldsymbol{p}^T\boldsymbol{B}\boldsymbol{p}}-\sqrt{(\boldsymbol{r}-\boldsymbol{p})^T\boldsymbol{B}(\boldsymbol{r}-\boldsymbol{p})}\\ &\ge \sqrt{\boldsymbol{p}^T\boldsymbol{B}\boldsymbol{p}}-2\delta/\sqrt{n}.\end{aligned}$$

Note that $\boldsymbol{p}$ involves only eigenvectors between $(1-\delta)\lambda_n$ to λ_n. Hence, we have

$$\sqrt{\boldsymbol{r}^T\boldsymbol{B}\boldsymbol{r}}\ge\sqrt{(1-\delta)\lambda_n}\|\boldsymbol{p}\|_2-2\delta/\sqrt{n}.$$

With constant probability, we have $\alpha_n=\Omega(1/\sqrt{n})$. Hence, we have $\|\boldsymbol{p}\|_2=\Omega(1/\sqrt{n})$. Using $\boldsymbol{B}\preceq 2\boldsymbol{I}$ and $\|\boldsymbol{p}\|_2\ge\|\boldsymbol{r}\|_2-\delta/\sqrt{n}$ we have that so long as δ is a small enough universal constant

$$\begin{aligned}\frac{\sqrt{\boldsymbol{r}^T\boldsymbol{B}\boldsymbol{r}}}{\|\boldsymbol{r}\|_2} &\ge \frac{\sqrt{(1-\delta)\lambda_n}\|\boldsymbol{p}\|_2-2\delta/\sqrt{n}}{\|\boldsymbol{p}\|_2+\delta/\sqrt{n}}\\ &= (1-O(\delta))\sqrt{\lambda_n}-O(\delta)\\ &= \sqrt{\lambda_n}-O(\delta\sqrt{R}).\end{aligned}$$

Therefore, we have $\frac{\boldsymbol{r}^T\boldsymbol{Y}\boldsymbol{r}}{\|\boldsymbol{r}\|^2}\ge\lambda_{\max}(\boldsymbol{Y})-O(R\delta)$. Hence, we can find vector $\boldsymbol{r}$ by computing k matrix multiplications. [66] showed that fast matrix multiplication is stable under Frobenius norm, i.e., for any $\eta>0$, using $O(\log(n/b))$ bits, we can find $\boldsymbol{C}$ such that $\|\boldsymbol{C}-\boldsymbol{A}\boldsymbol{B}\|_F\le\frac{1}{b}\|\boldsymbol{A}\|\|\boldsymbol{B}\|$ in time $O(n^{\omega+\eta})$ where ω is the matrix multiplicative constant. Hence, this algorithm takes only $O(n^{\omega+o(1)}\log^{O(1)}(\delta^{-1}))$ time. The result follows from renormalizing the vector $\boldsymbol{r}$, repeating the algorithm $O(\log n)$ times to boost the probability and taking $\delta=\Omega(\varepsilon/R)$. □

The following lemma shows how to compute a separation for f_K defined in (9.7).

Lemma 9.3.4. *Suppose that $\|\boldsymbol{A}_i\|_F\le M$ and $\|\boldsymbol{C}\|_F\le M$. For any $0<\varepsilon<1$ and $\boldsymbol{y}$ with $\|\boldsymbol{y}\|_2=O(L)$, with high probability in m, we can compute a $(\varepsilon,\varepsilon)$-separation of f_K on $\{\|\boldsymbol{x}\|_2\le L\}$ at $\boldsymbol{y}$ in time $O(S+m^{\omega+o(1)}\log^{O(1)}(nKML/\varepsilon))$ where where S is the sparsity of the problem defined as $\operatorname{nnz}(\boldsymbol{C})+\sum_{i=1}^n\operatorname{nnz}(\boldsymbol{A}_i)$.*

Proof. Note that $-O(nML)\boldsymbol{I}\preceq\boldsymbol{C}-\sum_{i=1}^n y_i\boldsymbol{A}_i\preceq O(nML)\boldsymbol{I}$. Using Lemma 9.3.3, we can find a vector $\boldsymbol{v}$ with $\|\boldsymbol{v}\|_2=K$ in time $O(m^{\omega+o(1)}\log^{O(1)}(nKML/\delta))$ such that

$$\boldsymbol{v}^T\left(\boldsymbol{C}-\sum_{i=1}^n y_i\boldsymbol{A}_i\right)\boldsymbol{v}\ge\max_{\operatorname{Tr}\boldsymbol{X}\le K,\boldsymbol{X}\succeq\mathbf{0}}\left\langle\boldsymbol{X},\boldsymbol{C}-\sum_{i=1}^n y_i\boldsymbol{A}_i\right\rangle-\delta. \tag{9.8}$$

In other words, we have a δ-optimization oracle for the function f_K. Lemma 9.2.6 shows this yields a δ-subgradient oracle and Lemma 9.2.4 then shows this yields a $\left(O(\sqrt{\delta L}),O(\sqrt{\delta L})\right)$-separation oracle on the set $\{\|\boldsymbol{x}\|_2\le L\}$. By picking $\delta=\varepsilon^2/L$, we have the promised oracle. □

With the separation oracle in hand, we are ready to give the algorithm for SDP:

Theorem 9.3.5. *Given a primal-dual semidefinite programming problem in the form (9.4) and (9.5), suppose that for some $M\ge 1$ we have*

1. $\|b\|_2 \le M$, $\|\boldsymbol{C}\|_F \le M$ *and* $\|\boldsymbol{A}_i\|_F \le M$ *for all* i.
2. *The primal feasible set lies inside the region* $\mathrm{Tr}\boldsymbol{X} \le M$.
3. *The dual feasible set lies inside the region* $\|\boldsymbol{y}\|_\infty \le M$.

Let OPT *be the optimum solution of* (9.4) *and* (9.5). *Then, with high probability, we can find* $\boldsymbol{X}$ *and* $\boldsymbol{y}$ *such that*

1. $\boldsymbol{X} \succeq \boldsymbol{0}$, $\mathrm{Tr}\boldsymbol{X} = O(M)$, $\sum_i |b_i - \langle \boldsymbol{X}, \boldsymbol{A}_i\rangle| \le \varepsilon$ *for all* i *and* $\boldsymbol{C} \bullet \boldsymbol{X} \ge \mathrm{OPT} - \varepsilon$.
2. $\|\boldsymbol{y}\|_\infty = O(M)$, $\sum_{i=1}^n y_i \boldsymbol{A}_i \succeq \boldsymbol{C} - \varepsilon \boldsymbol{I}$ *and* $\boldsymbol{b}^T\boldsymbol{y} \le \mathrm{OPT} + \varepsilon$.

in expected time $O\left(\left(nS + n^3 + nm^{\omega+o(1)}\right)\log^{O(1)}\left(\frac{nM}{\varepsilon}\right)\right)$ *where* S *is the sparsity of the problem defined as* $\mathrm{nnz}(\boldsymbol{C}) + \sum_{i=1}^n \mathrm{nnz}(\boldsymbol{A}_i)$ *and* ω *is the fast matrix multiplication constant.*

Proof. Let $K \ge M$ be some parameter to be determined. Since the primal feasible set is lies inside the region $\mathrm{Tr}\boldsymbol{X} \le M \le K$, we have

$$\begin{aligned}
\min_{\sum_{i=1}^n y_i \boldsymbol{A}_i \succeq \boldsymbol{C}} \boldsymbol{b}^T\boldsymbol{y} &= \max_{\boldsymbol{X}\succeq\boldsymbol{0}, \mathrm{Tr}\boldsymbol{X}\le K, \boldsymbol{A}_i\bullet\boldsymbol{X}=b_i} \boldsymbol{C}\bullet\boldsymbol{X} \\
&= \max_{\boldsymbol{X}\succeq\boldsymbol{0}, \mathrm{Tr}\boldsymbol{X}\le K} \min_{\boldsymbol{y}} \boldsymbol{C}\bullet\boldsymbol{X} - \sum_i y_i\left(\boldsymbol{A}_i\bullet\boldsymbol{X} - b_i\right) \\
&= \min_{\boldsymbol{y}} \max_{\boldsymbol{X}\succeq\boldsymbol{0}, \mathrm{Tr}\boldsymbol{X}\le K} \left(\boldsymbol{b}^T\boldsymbol{y} + (\boldsymbol{C} - \sum_i y_i\boldsymbol{A}_i)\bullet\boldsymbol{X}\right) \\
&= \min_{\boldsymbol{y}} f_K(\boldsymbol{y}).
\end{aligned}$$

Lemma 9.3.4 shows that it takes $\mathrm{SO}_{\delta,\delta}(f_K) = O(S + m^{\omega+o(1)}\log(nKML/\delta))$ time to compute a (δ,δ)-separation oracle of f_K for any point $\boldsymbol{y}$ with $\|\boldsymbol{y}\|_\infty = O(L)$ where L is some parameter with $L \ge M$. Taking the radius $R = L$, Theorem 9.3.2 shows that it takes $O\left(n\mathrm{SO}_{\delta,\delta}(f_K)\log\left(\frac{n}{\alpha}\right) + n^3\log^{O(1)}\left(\frac{n}{\alpha}\right)\right)$ expected time with $\delta = \Theta\left(\alpha n^{-3/2}L\right)$ to find $\boldsymbol{y}$ such that

$$f_K(\boldsymbol{y}) - \min_{\|\boldsymbol{y}\|_\infty\le L} f_K(\boldsymbol{y}) \le \delta + \alpha\left(\max_{\|\boldsymbol{y}\|_\infty\le L} f_K(\boldsymbol{y}) - \min_{\|\boldsymbol{y}\|_\infty\le L} f_K(\boldsymbol{y})\right) \le \delta + 2\alpha\left(nML + 2nKML\right).$$

Picking $\alpha = \frac{\varepsilon}{7nMKL}$, we have $f_K(\boldsymbol{y}) \le \min_{\boldsymbol{y}} f_K(\boldsymbol{y}) + \varepsilon$. Therefore,

$$\boldsymbol{b}^T\boldsymbol{y} + K\max(\lambda_{\max}(\boldsymbol{C} - \sum_{i=1}^n y_i\boldsymbol{A}_i), 0) \le \mathrm{OPT} + \varepsilon.$$

Let $\beta = \max(\lambda_{\max}(\boldsymbol{C} - \sum_{i=1}^n y_i\boldsymbol{A}_i), 0)$. Then, we have that $\sum_{i=1}^n y_i\boldsymbol{A}_i \succeq \boldsymbol{C} - \beta\boldsymbol{I}$ and

$$\begin{aligned}
\boldsymbol{b}^T\boldsymbol{y} &\ge \min_{\sum_{i=1}^n y_i\boldsymbol{A}_i \succeq \boldsymbol{C}-\beta\boldsymbol{I}} \boldsymbol{b}^T\boldsymbol{y} \\
&= \max_{\boldsymbol{X}\succeq\boldsymbol{0}\boldsymbol{A}_i\bullet\boldsymbol{X}=b_i} (\boldsymbol{C} - \beta\boldsymbol{I})\bullet\boldsymbol{X} \\
&\ge \mathrm{OPT} - \beta M
\end{aligned}$$

because $\mathrm{Tr}\boldsymbol{X} \le M$. Hence, we have

$$\mathrm{OPT} - \beta M + \beta K \le \boldsymbol{b}^T\boldsymbol{y} + K\max(\lambda_{\max}(\boldsymbol{C} - \sum_{i=1}^n y_i\boldsymbol{A}_i), 0) \le \mathrm{OPT} + \varepsilon$$

Putting $K = M + 1$, we have $\beta \leq \varepsilon$. Thus,

$$\sum_{i=1}^{n} y_i \boldsymbol{A}_i \succeq \boldsymbol{C} - \varepsilon \boldsymbol{I}.$$

This gives the result for the dual with the running time $O\left(\left(nS + n^3 + nm^{\omega+o(1)}\right)\log^{O(1)}\left(\frac{nML}{\varepsilon}\right)\right)$.

Our Cutting Plane Method accesses the sub-problem

$$\max_{\boldsymbol{X}\succeq\boldsymbol{0},\mathrm{Tr}\boldsymbol{X}\leq K}(\boldsymbol{C} - \sum_i y_i \boldsymbol{A}_i) \bullet \boldsymbol{X}$$

only through the separation oracle. Let $\boldsymbol{z}$ be the output of our Cutting Plane Method and $\{\boldsymbol{v}_i\boldsymbol{v}_i^T\}_{i=1}^{O(n)}$ be the matrices used to construct the separation for the $O(n)$ hyperplanes the algorithm maintains at the end. Let $\boldsymbol{u}$ be the maximum eigenvector of $\boldsymbol{C} - \sum_{i=1}^n z_i \boldsymbol{A}_i$. Now, we consider a realization of f_K

$$\tilde{f}_K(\boldsymbol{y}) = \boldsymbol{b}^T\boldsymbol{y} + \max_{\boldsymbol{X}\in\mathrm{conv}(K\boldsymbol{u}\boldsymbol{u}^T,\boldsymbol{v}_i\boldsymbol{v}_i^T)} \left\langle \boldsymbol{X}, \boldsymbol{C} - \sum_{i=1}^n y_i \boldsymbol{A}_i \right\rangle.$$

Since applying our Cutting Plane Method to either f_K or $\tilde{f}_K$ gives the same result, the correctness of the our Cutting Plane Method shows that

$$\tilde{f}_K(\boldsymbol{z}) \leq \min_{\|\boldsymbol{y}\|_\infty\leq L} \tilde{f}_K(\boldsymbol{y}) + \varepsilon.$$

Note that the function $\tilde{f}_K$ is defined such that $\tilde{f}_K(\boldsymbol{z}) = f_K(\boldsymbol{z})$. Hence, we have

$$\min_{\|\boldsymbol{y}\|_\infty\leq L} f_K(\boldsymbol{y}) \leq f_K(\boldsymbol{z}) \leq \tilde{f}_K(\boldsymbol{z}) \leq \min_{\|\boldsymbol{y}\|_\infty\leq L} \tilde{f}_K(\boldsymbol{y}) + \varepsilon.$$

Also, note that $\tilde{f}_K(\boldsymbol{x}) \leq f_K(\boldsymbol{x})$ for all $\boldsymbol{x}$. Hence, we have

$$\min_{\|\boldsymbol{y}\|_\infty\leq L} f_K(\boldsymbol{y}) - \varepsilon \leq \min_{\|\boldsymbol{y}\|_\infty\leq L} \tilde{f}(\boldsymbol{y}) \leq \min_{\|\boldsymbol{y}\|_\infty\leq L} f_K(\boldsymbol{y}).$$

Now, we consider the primal version of $\tilde{f}$, namely

$$g(\boldsymbol{X}) \stackrel{\mathrm{def}}{=} \min_{\|\boldsymbol{y}\|_\infty\leq L} \boldsymbol{b}^T\boldsymbol{y} + \left\langle \boldsymbol{X}, \boldsymbol{C} - \sum_{i=1}^n y_i \boldsymbol{A}_i \right\rangle.$$

Sion's minimax theorem [239] shows that

$$\mathrm{OPT} \geq \max_{\boldsymbol{X}\in\mathrm{conv}(K\boldsymbol{u}\boldsymbol{u}^T,\boldsymbol{v}_i\boldsymbol{v}_i^T)} g(\boldsymbol{X}) = \min_{\|\boldsymbol{y}\|_\infty\leq L} \tilde{f}(\boldsymbol{y}) \geq \mathrm{OPT} - \varepsilon.$$

Therefore, to get the primal solution, we only need to find $\boldsymbol{u}$ by Lemma 9.3.3 and solve the maximization problem on g. Note that

$$\begin{aligned} g(\boldsymbol{X}) &= \min_{\|\boldsymbol{y}\|_\infty\leq L} \sum_{i=1}^n y_i \left(b_i - \langle \boldsymbol{X}, \boldsymbol{A}_i\rangle\right) + \langle \boldsymbol{X}, \boldsymbol{C}\rangle \\ &= -L\sum_i \left|b_i - \langle \boldsymbol{X}, \boldsymbol{A}_i\rangle\right| + \langle \boldsymbol{X}, \boldsymbol{C}\rangle. \end{aligned}$$

For notation simplicity, we write $K\boldsymbol{u}\boldsymbol{u}^T = \boldsymbol{v}_0\boldsymbol{v}_0^T$. Then, $\boldsymbol{X} = \sum_{j=0}^{O(n)} \alpha_j \boldsymbol{v}_j\boldsymbol{v}_j^T$ for some $\sum \alpha_j = 1$ and

$\alpha_j \geq 0$. Substituting this into the function g, we have

$$g(\boldsymbol{\alpha}) = -L\sum_j \left| b_i - \sum_j \alpha_j \boldsymbol{v}_j^T \boldsymbol{A}_i \boldsymbol{v}_j \right| + \sum_j \alpha_j \boldsymbol{v}_j^T \boldsymbol{C} \boldsymbol{v}_j.$$

Hence, this can be easily written as a linear program with $O(n)$ variables and $O(n)$ constraints in time $O(nS)$. Now, we can apply interior point method to find $\boldsymbol{\alpha}$ such that

$$g(\boldsymbol{\alpha}) \geq \max_{\boldsymbol{X} \in \mathrm{conv}(K\boldsymbol{u}\boldsymbol{u}^T, \boldsymbol{v}_i \boldsymbol{v}_i^T)} g(\boldsymbol{X}) - \varepsilon \geq \mathrm{OPT} - 2\varepsilon.$$

Let the corresponding approximate solution be $\widetilde{\boldsymbol{X}} = \sum \alpha_j \boldsymbol{v}_j \boldsymbol{v}_j^T$. Then, we have

$$\left\langle \widetilde{\boldsymbol{X}}, \boldsymbol{C} \right\rangle - L\sum_i |b_i - \langle \boldsymbol{X}, \boldsymbol{A}_i \rangle| \geq \mathrm{OPT} - 2\varepsilon.$$

Now, we let $\tilde{b}_i = \left\langle \widetilde{\boldsymbol{X}}, \boldsymbol{A}_i \right\rangle$. Then, we note that

$$\begin{aligned} \left\langle \widetilde{\boldsymbol{X}}, \boldsymbol{C} \right\rangle &\leq \max_{\boldsymbol{X} \succeq \boldsymbol{0} \boldsymbol{A}_i \bullet \boldsymbol{X} = \tilde{b}_i} \boldsymbol{C} \bullet \boldsymbol{X} \\ &= \min_{\sum_{i=1}^n y_i \boldsymbol{A}_i \succeq \boldsymbol{C}} \tilde{b}_i^T \boldsymbol{y} \\ &\leq \mathrm{OPT} + M \sum_i \left| b_i - \left\langle \widetilde{\boldsymbol{X}}, \boldsymbol{A}_i \right\rangle \right| \end{aligned}$$

because $\|\boldsymbol{y}\|_\infty \leq M$. Hence, we have

$$\mathrm{OPT} + (M-L)\sum_i \left| b_i - \left\langle \widetilde{\boldsymbol{X}}, \boldsymbol{A}_i \right\rangle \right| \geq \left\langle \widetilde{\boldsymbol{X}}, \boldsymbol{C} \right\rangle - L\sum_i \left| b_i - \left\langle \widetilde{\boldsymbol{X}}, \boldsymbol{A}_i \right\rangle \right| \geq \mathrm{OPT} - 2\varepsilon.$$

Now, we put $L = M + 2$, we have

$$\sum_i \left| b_i - \left\langle \widetilde{\boldsymbol{X}}, \boldsymbol{A}_i \right\rangle \right| \leq \varepsilon.$$

This gives the result for the primal. Note that it only takes $O(n^{5/2} \log^{O(1)}(nM/\varepsilon))$ to solve a linear program with $O(n)$ variables and $O(n)$ constraints because we have an explicit interior point deep inside the feasible set, i.e. $\alpha_i = \frac{1}{m}$ for some parameter m [164].[1] Hence, the running time is dominated by the cost of cutting plane method which is $O\left(\left(nS + n^3 + nm^{\omega+o(1)}\right) \log^{O(1)}\left(\frac{nM}{\varepsilon}\right)\right)$ by putting $L = M + 2$. □

We leave it as an open problem if it is possible to improve this result by reusing the computation in the separation oracle and achieve a running time of $O\left(\left(nS + n^3 + nm^2\right) \log^{O(1)}\left(\frac{nM}{\varepsilon}\right)\right)$.

9.4 Intersection of Convex Sets

In this section we introduce a general technique to optimize a linear function over the intersection of two convex sets, whenever the linear optimization problem on each of them can be done efficiently. At the very high level, this is accomplished by applying cutting plane to a suitably regularized version of the problem. In Section 9.4.1 we present the technique and in the remaining sections we provide

[1] Without this, the running time of interior point method depends on the bit complexity of the linear programs.

several applications including, matroid intersection (Section 9.4.2), submodular flow (Section 9.4.3), and minimizing over the intersection of an affine subspace and a convex set (Section 9.4.4).

9.4.1 The Technique

Throughout this section we consider variants of the following general optimization problem

$$\max_{\boldsymbol{x}\in K_1\cap K_2} \langle \boldsymbol{c}, \boldsymbol{x}\rangle \tag{9.9}$$

where $\boldsymbol{x}, \boldsymbol{c} \in \mathbb{R}^n$, K_1 and K_2 are convex subsets of $\mathbb{R}^n$. We assume that

$$\max_{\boldsymbol{x}\in K_1} \|\boldsymbol{x}\|_2 < M,\ \max_{\boldsymbol{x}\in K_2} \|\boldsymbol{x}\|_2 < M,\ \|\boldsymbol{c}\|_2 \le M \tag{9.10}$$

for some constant $M \ge 1$ and we assume that

$$K_1 \cap K_2 \neq \emptyset. \tag{9.11}$$

Instead of a separation oracle, we assume that K_1 and K_2 each have optimization oracles (see Section 9.2).

To solve this problem we first introduce a relaxation for the problem (9.9) that we can optimize efficiently. Because we have only the optimization oracles for K_1 and K_2, we simply have variables $\boldsymbol{x}$ and $\boldsymbol{y}$ for each of them in the objective. Since the output should (approximately) be in the intersection of K_1 and K_2, a regularization term $-\frac{\lambda}{2}\|\boldsymbol{x}-\boldsymbol{y}\|_2^2$ is added to force $\boldsymbol{x} \approx \boldsymbol{y}$ where λ is a large number to be determined later. Furthermore, we add terms to make the problem strong concave.

Lemma 9.4.1. *Assume (9.10) and (9.11). For $\lambda \ge 1$, let*

$$f_\lambda(\boldsymbol{x},\boldsymbol{y}) \stackrel{\text{def}}{=} \frac{1}{2}\langle \boldsymbol{c}, \boldsymbol{x}\rangle + \frac{1}{2}\langle \boldsymbol{c}, \boldsymbol{y}\rangle - \frac{\lambda}{2}\|\boldsymbol{x}-\boldsymbol{y}\|_2^2 - \frac{1}{2\lambda}\|\boldsymbol{x}\|_2^2 - \frac{1}{2\lambda}\|\boldsymbol{y}\|_2^2 \quad . \tag{9.12}$$

There is an unique maximizer $(\boldsymbol{x}_\lambda, \boldsymbol{y}_\lambda)$ for the problem $\max_{\boldsymbol{x}\in K_1, \boldsymbol{y}\in K_2} f_\lambda(\boldsymbol{x},\boldsymbol{y})$. The maximizer $(\boldsymbol{x}_\lambda, \boldsymbol{y}_\lambda)$ is a good approximation of the solution of (9.9), i.e. $\|\boldsymbol{x}_\lambda - \boldsymbol{y}_\lambda\|_2^2 \le \frac{6M^2}{\lambda}$ and

$$\max_{\boldsymbol{x}\in K_1\cap K_2} \langle \boldsymbol{c}, \boldsymbol{x}\rangle \le f_\lambda(\boldsymbol{x}_\lambda, \boldsymbol{y}_\lambda) + \frac{M^2}{\lambda}. \tag{9.13}$$

Proof. Let $\boldsymbol{x}^*$ be a maximizer of $\max_{\boldsymbol{x}\in K_1\cap K_2}\langle \boldsymbol{c}, \boldsymbol{x}\rangle$. By assumption (9.10), $\|\boldsymbol{x}^*\|_2 \le M$, and therefore

$$f_\lambda(\boldsymbol{x}^*, \boldsymbol{x}^*) = \langle \boldsymbol{c}, \boldsymbol{x}^*\rangle - \frac{\|\boldsymbol{x}^*\|_2^2}{\lambda} \ge \max_{\boldsymbol{x}\in K_1\cap K_2}\langle \boldsymbol{c}, \boldsymbol{x}\rangle - \frac{M^2}{\lambda} \quad . \tag{9.14}$$

This shows (9.13). Since f_λ is strongly concave in $\boldsymbol{x}$ and $\boldsymbol{y}$, there is a unique maximizer $(\boldsymbol{x}_\lambda, \boldsymbol{y}_\lambda)$. Let $\mathrm{OPT}_\lambda = f_\lambda(\boldsymbol{x}_\lambda, \boldsymbol{y}_\lambda)$. Then, we have

$$\begin{aligned}\mathrm{OPT}_\lambda &\le \frac{1}{2}\|\boldsymbol{c}\|_2\|\boldsymbol{x}_\lambda\|_2 + \frac{1}{2}\|\boldsymbol{c}\|_2\|\boldsymbol{y}_\lambda\|_2 - \frac{\lambda}{2}\|\boldsymbol{x}_\lambda - \boldsymbol{y}_\lambda\|_2^2 \\ &\le \frac{M^2}{2} + \frac{M^2}{2} - \frac{\lambda}{2}\|\boldsymbol{x}_\lambda - \boldsymbol{y}_\lambda\|_2^2.\end{aligned}$$

On the other hand, using $\lambda \ge 1$, (9.14) shows that

$$\mathrm{OPT}_\lambda \ge f_\lambda(\boldsymbol{x}^*, \boldsymbol{x}^*) \ge \max_{\boldsymbol{x}\in K_1\cap K_2}\langle \boldsymbol{c}, \boldsymbol{x}\rangle - \frac{M^2}{\lambda} \ge -2M^2.$$

Hence, we have

$$\|\boldsymbol{x}_\lambda - \boldsymbol{y}_\lambda\|_2^2 \quad \leq \quad \frac{2\left(M^2 - \mathrm{OPT}_\lambda\right)}{\lambda} \leq \frac{6M^2}{\lambda}. \tag{9.15}$$

□

Now we write $\max f_\lambda(\boldsymbol{x}, \boldsymbol{y})$ as a max-min problem. The reason for doing this is that the dual approximate solution is much easier to obtain and there is a way to read off a primal approximate solution from a dual approximate solution. This is analogous to the idea in [161] which showed how to convert a cut solution to a flow solution by adding regularization terms into the problem.

Lemma 9.4.2. *Assume (9.10) and (9.11). Let $\lambda \geq 2$. For any $\boldsymbol{x} \in K_1$ and $\boldsymbol{y} \in K_2$, the function f_λ can be represented as*

$$f_\lambda(\boldsymbol{x}, \boldsymbol{y}) = \min_{(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3) \in \Omega} g_\lambda(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3) \tag{9.16}$$

where $\Omega = \{(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3) : \|\boldsymbol{\theta}_1\|_2 \leq 2M,\ \|\boldsymbol{\theta}_2\|_2 \leq M,\ \|\boldsymbol{\theta}_3\|_2 \leq M\}$ and

$$g_\lambda(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3) = \left\langle \frac{\boldsymbol{c}}{2} + \lambda\boldsymbol{\theta}_1 + \frac{\boldsymbol{\theta}_2}{\lambda}, \boldsymbol{x} \right\rangle + \left\langle \frac{\boldsymbol{c}}{2} - \lambda\boldsymbol{\theta}_1 + \frac{\boldsymbol{\theta}_3}{\lambda}, \boldsymbol{y} \right\rangle + \frac{\lambda}{2}\|\boldsymbol{\theta}_1\|_2^2 + \frac{1}{2\lambda}\|\boldsymbol{\theta}_2\|_2^2 + \frac{1}{2\lambda}\|\boldsymbol{\theta}_3\|_2^2. \tag{9.17}$$

Let $h_\lambda(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3) = \max_{\boldsymbol{x} \in K_1, \boldsymbol{y} \in K_2} g_\lambda(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3)$. For any $(\boldsymbol{\theta}'_1, \boldsymbol{\theta}'_2, \boldsymbol{\theta}'_3)$ such that $h_\lambda(\boldsymbol{\theta}'_1, \boldsymbol{\theta}'_2, \boldsymbol{\theta}'_3) \leq \min_{(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3) \in \Omega} h_\lambda(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3) + \varepsilon$, we know $\boldsymbol{z} = \frac{1}{2}(\boldsymbol{\theta}'_2 + \boldsymbol{\theta}'_3)$ satisfies

$$\max_{\boldsymbol{x} \in K_1 \cap K_2} \langle \boldsymbol{c}, \boldsymbol{x} \rangle \leq \langle \boldsymbol{c}, \boldsymbol{z} \rangle + \frac{20M^2}{\lambda} + 20\lambda^3\varepsilon.$$

and $\|\boldsymbol{z} - \boldsymbol{x}_\lambda\|_2 + \|\boldsymbol{z} - \boldsymbol{y}_\lambda\|_2 \leq 4\sqrt{2\lambda\varepsilon} + \sqrt{\frac{6M^2}{\lambda}}$ where $(\boldsymbol{x}_\lambda, \boldsymbol{y}_\lambda)$ is the unique maximizer for the problem $\max_{\boldsymbol{x} \in K_1, \boldsymbol{y} \in K_2} f_\lambda(\boldsymbol{x}, \boldsymbol{y})$.

Proof. Note that for any $\|\boldsymbol{\zeta}\|_2 \leq \alpha$, we have

$$-\frac{1}{2}\|\boldsymbol{\zeta}\|_2^2 = \min_{\|\boldsymbol{\theta}\|_2 \leq \alpha} \langle \boldsymbol{\theta}, \boldsymbol{\zeta} \rangle + \frac{1}{2}\|\boldsymbol{\theta}\|_2^2$$

Using this and (9.10), we have (9.16) for all $\boldsymbol{x} \in K_1$ and $\boldsymbol{y} \in K_2$ as desired. Since Ω is closed and bounded set and the function g_λ is concave in $(\boldsymbol{x}, \boldsymbol{y})$ and convex in $(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3)$, Sion's minimax theorem [239] shows that

$$\max_{\boldsymbol{x} \in K_1, \boldsymbol{y} \in K_2} f_\lambda(\boldsymbol{x}, \boldsymbol{y}) = \min_{(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3) \in \Omega} h_\lambda(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3) \tag{9.18}$$

Since f_λ is strongly concave, there is an unique maximizer $(\boldsymbol{x}_\lambda, \boldsymbol{y}_\lambda)$ of f_λ. Since h_λ is strongly convex, there is a unique minimizer $(\boldsymbol{\theta}^*_1, \boldsymbol{\theta}^*_2, \boldsymbol{\theta}^*_3)$. By the definition of f_λ and h_λ, we have

$$h_\lambda(\boldsymbol{\theta}^*_1, \boldsymbol{\theta}^*_2, \boldsymbol{\theta}^*_3) \geq g_\lambda(\boldsymbol{x}_\lambda, \boldsymbol{y}_\lambda, \boldsymbol{\theta}^*_1, \boldsymbol{\theta}^*_2, \boldsymbol{\theta}^*_3) \geq f_\lambda(\boldsymbol{x}_\lambda, \boldsymbol{y}_\lambda) \quad .$$

Using (9.18), the equality above holds and hence $(\boldsymbol{\theta}^*_1, \boldsymbol{\theta}^*_2, \boldsymbol{\theta}^*_3)$ is the minimizer of $g_\lambda(\boldsymbol{x}_\lambda, \boldsymbol{y}_\lambda, \boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3)$ over $(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3)$. Since the domain Ω is large enough that $(\boldsymbol{\theta}^*_1, \boldsymbol{\theta}^*_2, \boldsymbol{\theta}^*_3)$ is an interior point in Ω, the optimality condition of g_λ shows that we have $\boldsymbol{\theta}^*_2 = \boldsymbol{x}_\lambda$ and $\boldsymbol{\theta}^*_3 = \boldsymbol{y}_\lambda$.

Since h_λ is $\frac{1}{\lambda}$ strongly convex, we have $\|\boldsymbol{\theta}'_1 - \boldsymbol{\theta}^*_1\|_2^2 + \|\boldsymbol{\theta}'_2 - \boldsymbol{\theta}^*_2\|_2^2 + \|\boldsymbol{\theta}'_3 - \boldsymbol{\theta}^*_3\|_2^2 \leq 2\lambda\varepsilon$ (Fact 9.2.3). Since $\boldsymbol{\theta}^*_2 = \boldsymbol{x}_\lambda$ and $\boldsymbol{\theta}^*_3 = \boldsymbol{y}_\lambda$, we have

$$\|\boldsymbol{\theta}'_2 - \boldsymbol{x}_\lambda\|_2^2 + \|\boldsymbol{\theta}'_3 - \boldsymbol{y}_\lambda\|_2^2 \leq 2\lambda\varepsilon. \tag{9.19}$$

Therefore, we have $\|\boldsymbol{x}_\lambda-\boldsymbol{y}_\lambda\|_2 \geq \|\boldsymbol{\theta}'_2-\boldsymbol{\theta}'_3\|_2-2\sqrt{2\lambda\varepsilon}$, $\|\boldsymbol{x}_\lambda\|_2 \geq \|\boldsymbol{\theta}'_2\|_2-\sqrt{2\lambda\varepsilon}$ and $\|\boldsymbol{y}_\lambda\|_2 \geq \|\boldsymbol{\theta}'_3\|_2-\sqrt{2\lambda\varepsilon}$. Using these, $\|\boldsymbol{x}_\lambda\|_2 \leq M$ and $\|\boldsymbol{y}_\lambda\|_2 \leq M$, we have

$$\begin{aligned} f_\lambda(\boldsymbol{\theta}'_2,\boldsymbol{\theta}'_3) &= \frac{1}{2}\langle \boldsymbol{c},\boldsymbol{\theta}'_2\rangle + \frac{1}{2}\langle \boldsymbol{c},\boldsymbol{\theta}'_3\rangle - \frac{\lambda}{2}\|\boldsymbol{\theta}'_2-\boldsymbol{\theta}'_3\|_2^2 - \frac{1}{2\lambda}\|\boldsymbol{\theta}'_2\|_2^2 - \frac{1}{2\lambda}\|\boldsymbol{\theta}'_3\|_2^2 \\ &\geq \frac{1}{2}\langle \boldsymbol{c},\boldsymbol{x}_\lambda\rangle + \frac{1}{2}\langle \boldsymbol{c},\boldsymbol{y}_\lambda\rangle - M\sqrt{2\lambda\varepsilon} \\ &\quad -\frac{\lambda}{2}\left(\|\boldsymbol{x}_\lambda-\boldsymbol{y}_\lambda\|_2+2\sqrt{2\lambda\varepsilon}\right)^2 \\ &\quad -\frac{1}{2\lambda}\left(\|\boldsymbol{x}_\lambda\|_2+\sqrt{2\lambda\varepsilon}\right)^2 - \frac{1}{2\lambda}\left(\|\boldsymbol{y}_\lambda\|_2+\sqrt{2\lambda\varepsilon}\right)^2 \\ &= \frac{1}{2}\langle \boldsymbol{c},\boldsymbol{x}_\lambda\rangle + \frac{1}{2}\langle \boldsymbol{c},\boldsymbol{y}_\lambda\rangle - \frac{\lambda}{2}\|\boldsymbol{x}_\lambda-\boldsymbol{y}_\lambda\|_2^2 - \frac{1}{2\lambda}\|\boldsymbol{x}_\lambda\|_2^2 - \frac{1}{2\lambda}\|\boldsymbol{y}_\lambda\|_2^2 \\ &\quad -M\sqrt{2\lambda\varepsilon} - 2\lambda\sqrt{2\lambda\varepsilon}\|\boldsymbol{x}_\lambda-\boldsymbol{y}_\lambda\|_2 - 4\lambda^2\varepsilon \\ &\quad -\frac{1}{\lambda}\|\boldsymbol{x}_\lambda\|_2\sqrt{2\lambda\varepsilon}-\varepsilon-\frac{1}{\lambda}\|\boldsymbol{y}_\lambda\|_2\sqrt{2\lambda\varepsilon}-\varepsilon. \end{aligned}$$

Using $\|\boldsymbol{x}_\lambda-\boldsymbol{y}_\lambda\|_2 \leq \sqrt{\frac{6M^2}{\lambda}}$ (Lemma 9.4.1), $\|\boldsymbol{x}_\lambda\|_2 < M$ and $\|\boldsymbol{y}_\lambda\|_2 < M$, we have

$$\begin{aligned} f_\lambda(\boldsymbol{\theta}'_2,\boldsymbol{\theta}'_3) &\geq f_\lambda(\boldsymbol{x}_\lambda,\boldsymbol{y}_\lambda) \\ &\quad -M\sqrt{2\lambda\varepsilon} - 2\lambda\sqrt{2\lambda\varepsilon}\|\boldsymbol{x}_\lambda-\boldsymbol{y}_\lambda\|_2 - 4\lambda^2\varepsilon \\ &\quad -\frac{1}{\lambda}\|\boldsymbol{x}_\lambda\|_2\sqrt{2\lambda\varepsilon}-\varepsilon-\frac{1}{\lambda}\|\boldsymbol{y}_\lambda\|_2\sqrt{2\lambda\varepsilon}-\varepsilon. \\ &\geq f_\lambda(\boldsymbol{x}_\lambda,\boldsymbol{y}_\lambda) \\ &\quad -M\sqrt{2\lambda\varepsilon} - 2\lambda\sqrt{12\varepsilon}M - 4\lambda^2\varepsilon \\ &\quad -2M\sqrt{2\frac{\varepsilon}{\lambda}}-2\varepsilon. \end{aligned}$$

Since $\lambda \geq 2$, we have

$$f_\lambda(\boldsymbol{\theta}'_2,\boldsymbol{\theta}'_3) \geq f_\lambda(\boldsymbol{x}_\lambda,\boldsymbol{y}_\lambda) - 20M\lambda\sqrt{\varepsilon} - 10\lambda^2\varepsilon.$$

Let $\boldsymbol{z} = \frac{\boldsymbol{\theta}'_2+\boldsymbol{\theta}'_3}{2}$. Lemma 9.4.1 shows that

$$\begin{aligned} \max_{\boldsymbol{x}\in K_1\cap K_2}\langle \boldsymbol{c},\boldsymbol{x}\rangle &\leq \max_{\boldsymbol{x}\in K_1,\boldsymbol{y}\in K_2} f_\lambda(\boldsymbol{x},\boldsymbol{y}) + \frac{M^2}{\lambda} \\ &\leq f_\lambda(\boldsymbol{\theta}'_2,\boldsymbol{\theta}'_3) + \frac{M^2}{\lambda} + 20M\lambda\sqrt{\varepsilon} + 10\lambda^2\varepsilon \\ &\leq \langle \boldsymbol{c},\boldsymbol{z}\rangle + \frac{20M^2}{\lambda} + 20\lambda^3\varepsilon \end{aligned}$$

because $20M\lambda\sqrt{\varepsilon} \leq 10\frac{M^2}{\lambda} + 10\lambda^3\varepsilon$. Furthermore, we have

$$\begin{aligned} \|\boldsymbol{z}-\boldsymbol{x}_\lambda\|_2 + \|\boldsymbol{z}-\boldsymbol{y}_\lambda\|_2 &\leq \|\boldsymbol{\theta}'_2-\boldsymbol{x}_\lambda\|_2 + \|\boldsymbol{\theta}'_3-\boldsymbol{y}_\lambda\|_2 + \|\boldsymbol{\theta}'_2-\boldsymbol{\theta}'_3\|_2 \\ &\leq 4\sqrt{2\lambda\varepsilon} + \sqrt{\frac{6M^2}{\lambda}}. \end{aligned}$$

□

We now apply our cutting plane method to solve the optimization problem (9.9). First we show how to transform the optimization oracles for K_1 and K_2 to get a separation oracle for h_λ, with the

appropriate parameters.

Lemma 9.4.3. *Suppose we have a ε-optimization oracle for K_1 and K_2 for some $0 < \varepsilon < 1$. Then on the set $\{\|\boldsymbol{\theta}\|_2 \le D\}$, we have a $(O(\sqrt{\varepsilon\lambda D}), O(\sqrt{\varepsilon\lambda D}))$-separation oracle for h_λ with time complexity $\mathrm{OO}_\varepsilon(K_1) + \mathrm{OO}_\varepsilon(K_2)$.*

Proof. Recall that the function h_λ is defined by

$$\begin{aligned} & h_\lambda(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3) \\ = & \max_{\boldsymbol{x}\in K_1, \boldsymbol{y}\in K_2} \left(\left\langle \frac{\boldsymbol{c}}{2} + \lambda\boldsymbol{\theta}_1 + \frac{\boldsymbol{\theta}_2}{\lambda}, \boldsymbol{x} \right\rangle + \left\langle \frac{\boldsymbol{c}}{2} - \lambda\boldsymbol{\theta}_1 + \frac{\boldsymbol{\theta}_3}{\lambda}, \boldsymbol{y} \right\rangle + \frac{\lambda}{2}\|\boldsymbol{\theta}_1\|_2^2 + \frac{1}{2\lambda}\|\boldsymbol{\theta}_2\|_2^2 + \frac{1}{2\lambda}\|\boldsymbol{\theta}_3\|_2^2 \right) \\ = & \max_{\boldsymbol{x}\in K_1} \left\langle \frac{\boldsymbol{c}}{2} + \lambda\boldsymbol{\theta}_1 + \frac{\boldsymbol{\theta}_2}{\lambda}, \boldsymbol{x} \right\rangle + \max_{\boldsymbol{y}\in K_2} \left\langle \frac{\boldsymbol{c}}{2} - \lambda\boldsymbol{\theta}_1 + \frac{\boldsymbol{\theta}_3}{\lambda}, \boldsymbol{y} \right\rangle + \frac{\lambda}{2}\|\boldsymbol{\theta}_1\|_2^2 + \frac{1}{2\lambda}\|\boldsymbol{\theta}_2\|_2^2 + \frac{1}{2\lambda}\|\boldsymbol{\theta}_3\|_2^2. \end{aligned}$$

Lemma 9.2.6 shows how to compute the subgradient of functions of the form $f(\boldsymbol{c}) = \max_{\boldsymbol{x}\in K} \langle \boldsymbol{c}, \boldsymbol{x}\rangle$ using the optimization oracle for K. The rest of the term are differentiable so its subgradient is just the gradient. Hence, by addition rule for subgradients (Fact 9.2.3), we have a $O(\varepsilon\lambda)$-subgradient oracle for f_λ using a $O(\varepsilon)$-optimization oracle for K_1 and K_2. The result then follows from Lemma 9.2.4. □

Theorem 9.4.4. *Assume (9.10) and (9.11). Suppose that we have ε-optimization oracle for every $\varepsilon > 0$. For $0 < \delta < 1$, we can find $\boldsymbol{z} \in \mathbb{R}^n$ such that*

$$\max_{\boldsymbol{x}\in K_1\cap K_2} \langle \boldsymbol{c}, \boldsymbol{x}\rangle \le \delta + \langle \boldsymbol{c}, \boldsymbol{z}\rangle$$

and $\|\boldsymbol{z} - \boldsymbol{x}\|_2 + \|\boldsymbol{z} - \boldsymbol{y}\|_2 \le \delta$ for some $\boldsymbol{x} \in K_1$ and $\boldsymbol{y} \in K_2$ in time

$$O\left(n\left(\mathrm{OO}_\eta(K_1) + \mathrm{OO}_\eta(K_2)\right) \log\left(\frac{nM}{\delta}\right) + n^3 \log^{O(1)}\left(\frac{nM}{\delta}\right)\right)$$

where $\eta = \Omega\left(\left(\frac{\delta}{nM}\right)^{O(1)}\right)$.

Proof. Setting $\lambda = \frac{40M^2}{\delta^2}$ and $\varepsilon = \frac{\delta^7}{10^7 M^6}$ in Lemma 9.4.2 we see that so long as we obtain any approximate solution $(\boldsymbol{\theta}_1', \boldsymbol{\theta}_2', \boldsymbol{\theta}_3')$ such that

$$h_\lambda(\boldsymbol{\theta}_1', \boldsymbol{\theta}_2', \boldsymbol{\theta}_3') \le \min_{(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3)\in\Omega} h_\lambda(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3) + \varepsilon,$$

then we obtain the point we want. To apply Theorem 9.3.2, we use

$$\tilde{h}(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3) = \begin{cases} h_\lambda(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3) & \text{if } (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3) \in \Omega \\ +\infty & \text{else} \end{cases}.$$

Lemma 9.4.3 shows that for any $\gamma > 0$ we can obtain a (γ, γ)-separation oracle of $h_\lambda(\boldsymbol{\theta})$ by using sufficiently accurate optimization oracles. Since Ω is just a product of ℓ^2 balls, we can produce a separating hyperplane easily when $(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3) \notin \Omega$. Hence, we can obtain a (γ, γ)-separation oracle of $\tilde{h}(\boldsymbol{\theta})$. For simplicity, we use $\boldsymbol{\theta}$ to represent $(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3)$. Note that $B_\infty(2M) \supseteq \Omega$ and therefore we can apply Theorem 9.3.2 with $R = 2M$ to compute $\boldsymbol{\theta}'$ such

$$\tilde{h}(\boldsymbol{\theta}') - \min_{\boldsymbol{\theta}\in\Omega} \tilde{h}(\boldsymbol{\theta}) \le \gamma + \alpha\left(\max_{\boldsymbol{\theta}\in\Omega} \tilde{h}(\boldsymbol{\theta}) - \min_{\boldsymbol{\theta}\in\Omega} \tilde{h}(\boldsymbol{\theta})\right)$$

in time $O\left(n\mathrm{SO}_{\gamma,\gamma} \log\left(\frac{n\kappa}{\alpha}\right) + n^3 \log^{O(1)}\left(\frac{n\kappa}{\alpha}\right)\right)$ where $\gamma = \Omega\left(\alpha \mathrm{MinWidth}(\Omega)/n^{O(1)}\right) = \Omega\left(\alpha M/n^{O(1)}\right)$

and $\kappa = \frac{2M}{\text{MinWidth}(\Omega)} = O(1)$. Using $\lambda \geq 1$ and $M \geq 1$, we have

$$\max_{\boldsymbol{\theta}\in\Omega} \tilde{h}(\boldsymbol{\theta}) - \min_{\boldsymbol{\theta}\in\Omega} \tilde{h}(\boldsymbol{\theta}) \leq O\left(\lambda M^2\right) \leq O\left(\frac{M^4}{\delta^2}\right).$$

Setting $\alpha = \Theta\left(\frac{\delta^9}{M^{10}}\right)$ with some small enough constant, we have that we can find $\boldsymbol{\theta}'$ such that

$$\begin{aligned} h_\lambda(\boldsymbol{\theta}') &\leq \min_{\boldsymbol{\theta}\in P} h_\lambda(\boldsymbol{\theta}) + \gamma + \alpha O\left(\frac{M^4}{\delta^2}\right) \\ &= \min_{\boldsymbol{\theta}\in P} h_\lambda(\boldsymbol{\theta}) + O\left(\frac{\delta^7}{M^6}\right) \\ &= \min_{\boldsymbol{\theta}\in P} h_\lambda(\boldsymbol{\theta}) + \varepsilon \end{aligned}$$

in time $O\left(n\text{SO}_{\gamma,\gamma}\log\left(\frac{nM}{\delta}\right) + n^3\log^{O(1)}\left(\frac{nM}{\delta}\right)\right)$ where $\gamma = \Omega\left(\left(\frac{\delta}{nM}\right)^{O(1)}\right)$. Lemma 9.4.3 shows that the cost of (γ,γ)-separation oracle is just $O(\text{OO}_\eta(K_1) + \text{OO}_\eta(K_2))$ where $\eta = \Omega\left(\left(\frac{\delta}{nM}\right)^{O(1)}\right)$. □

Remark 9.4.5. Note that the algorithm does not promise that we obtain a point close to $K_1 \cap K_2$. It only promises to give a point that is close to both some point in K_1 and some point in K_2. It appears to the authors that a further assumption is needed to get a point close to $K_1 \cap K_2$. For example, if K_1 and K_2 are two almost parallel lines, it would be difficult to get an algorithm that does not depend on the angle. However, as far as we know, most algorithms tackling this problem are pseudo-polynomial and have polynomial dependence on the angle. Our algorithm depends on the logarithmic of the angle which is useful for combinatorial problems.

This reduction is very useful for problems in many areas including linear programming, semi-definite programming and algorithmic game theory. In the remainder of this section we demonstrate its power by applying it to classical combinatorial problems.

There is however one issue with applying our cutting plane algorithm to these problems. As with other convex optimization methods, only an approximately optimal solution is found. On the other hand, typically an exact solution is insisted in combinatorial optimization. To overcome this gap, we introduce the following lemma which (1) transforms the objective function so that there is only one optimal solution and (2) shows that an approximate solution is close to the optimal solution whenever it is unique. As we shall see in the next two subsections, this allows us to round an approximate solution to an optimal one.

Lemma 9.4.6. *Given a linear program* $\min_{\boldsymbol{A}\boldsymbol{x}\geq\boldsymbol{b}} \boldsymbol{c}^T\boldsymbol{x}$ *where* $\boldsymbol{x},\boldsymbol{c} \in \mathbb{Z}^n$, $\boldsymbol{b} \in \mathbb{Z}^m$ *and* $\boldsymbol{A} \in \mathbb{Z}^{m\times n}$. *Suppose* $\{\boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b}\}$ *is an integral polytope (i.e. all extreme points are integral) contained in the set* $\{\|\boldsymbol{x}\|_\infty \leq M\}$. *Then we can find a random cost vector* $\boldsymbol{z} \in \mathbb{Z}^n$ *with* $\|\boldsymbol{z}\|_\infty \leq O(n^2M^2\|\boldsymbol{c}\|_\infty)$ *such that with constant probability,* $\min_{\boldsymbol{A}\boldsymbol{x}\geq\boldsymbol{b}} \boldsymbol{z}^T\boldsymbol{x}$ *has an unique minimizer* $\boldsymbol{x}^*$ *and this minimizer is one of the minimizer(s) of* $\min_{\boldsymbol{A}\boldsymbol{x}\geq\boldsymbol{b}} \boldsymbol{c}^T\boldsymbol{x}$. *Furthermore, if there is an interior point* $\boldsymbol{y}$ *such that* $\boldsymbol{z}^T\boldsymbol{y} < \min_{\boldsymbol{A}\boldsymbol{x}\geq\boldsymbol{b}} \boldsymbol{z}^T\boldsymbol{x} + \delta$, *then* $\|\boldsymbol{y} - \boldsymbol{x}^*\|_\infty \leq 2nM\delta$.

Proof. The first part of the lemma follows by randomly perturbing the cost vector $\boldsymbol{c}$. We consider a new cost vector $\boldsymbol{z} = 100n^2M^2\boldsymbol{c} + \boldsymbol{r}$ where each coordinate of $\boldsymbol{r}$ is sampled randomly from $\{0, 1, \cdots, 10nM\}$. [145, Lem 4] shows that the linear program $\min_{\boldsymbol{A}\boldsymbol{x}\geq\boldsymbol{b}} \boldsymbol{z}^T\boldsymbol{x}$ has a unique minimizer with constant probability. Furthermore, it is clear that the minimizer of $\min_{\boldsymbol{A}\boldsymbol{x}\geq\boldsymbol{b}} \boldsymbol{z}^T\boldsymbol{x}$ is a minimizer of $\min_{\boldsymbol{A}\boldsymbol{x}\geq\boldsymbol{b}} \boldsymbol{c}^T\boldsymbol{x}$ (as $\boldsymbol{r}_i \ll 100n^2M^2|\boldsymbol{c}_i|$).

Now we show the second part of the lemma. Given an interior point $\boldsymbol{y}$ of the polytope $\{\boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b}\}$, we can write $\boldsymbol{y}$ as a convex combination of the vertices of $\{\boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b}\}$, i.e. $\boldsymbol{y} = \sum t_i\boldsymbol{v}_i$. Note that

$\boldsymbol{z}^T\boldsymbol{y} = \sum t_i \boldsymbol{z}^T\boldsymbol{v}_i$. If all $\boldsymbol{v}_i$ are not the minimizer, then $\boldsymbol{z}^T\boldsymbol{v}_i \geq \mathrm{OPT}+1$ and hence $\boldsymbol{z}^T\boldsymbol{y} \geq \mathrm{OPT}+1$ which is impossible. Hence, we can assume that $\boldsymbol{v}_1$ is the minimizer. Hence, $\boldsymbol{z}^T\boldsymbol{v}_i = \mathrm{OPT}$ if $i=1$ and $\boldsymbol{z}^T\boldsymbol{v}_i \geq \mathrm{OPT}+1$ otherwise. We then have $\boldsymbol{z}^T\boldsymbol{y} \geq \mathrm{OPT}+(1-t_1)$ which gives $1-t_1 < \delta$. Finally, the claim follows from $\|\boldsymbol{y}-\boldsymbol{v}_1\|_\infty \leq \sum_{i\neq 1} t_i \|\boldsymbol{v}_i - \boldsymbol{v}_1\|_\infty \leq 2nM\delta$. □

9.4.2 Matroid Intersection

Let $M_1 = (E, \mathcal{I}_1)$ and $M_2 = (E, \mathcal{I}_2)$ be two matroids sharing the same ground set. In this section we consider the weighted matroid intersection problem

$$\min_{S \in \mathcal{I}_1 \cap \mathcal{I}_2} \boldsymbol{w}(S).$$

where $\boldsymbol{w} \in \mathbb{R}^E$ and $w(S) \stackrel{\text{def}}{=} \sum_{e\in S} w_e$.

For any matroid $M = (E, \mathcal{I})$, it is well known that the polytope of all independent sets has the following description [80]:

$$\mathrm{conv}(\mathcal{I}_1) = \{\boldsymbol{x} \in \mathbb{R}^E \text{ s.t. } 0 \leq x(S) \leq r(S) \text{ for all } S \subseteq E\} \tag{9.20}$$

where r is the rank function for M, i.e. $r(S)$ is the size of the largest independent set that is a subset of S. Furthermore, the polytope of the matroid intersection satisfies $\mathrm{conv}(\mathcal{I}_1 \cap \mathcal{I}_2) = \mathrm{conv}(\mathcal{I}_1) \cap \mathrm{conv}(\mathcal{I}_2)$.

It is well known that the optimization problem

$$\min_{S\in\mathcal{I}_1} w(S) \text{ and } \min_{S\in\mathcal{I}_2} w(S)$$

can be solved efficiently by the greedy method. Given a matroid (polytope), the greedy method finds a maximum weight independent subset by maintaining a candidate independent subset S and iteratively attempts to add new element to S in descending weight. A element i is added to S if $S \cup \{i\}$ is still independent. A proof of this algorithm is well-known and can be found in any standard textbook on combinatorial optimization.

Clearly, the greedy method can be implemented by $O(n)$ calls to the independence oracle (also called membership oracle). For rank oracle, it requires $O(r \log n)$ calls by finding the next element to add via binary search. Therefore, we can apply Theorem 9.4.4 to get the following result (note that this algorithm is the fastest if r is close to n for the independence oracle).

Theorem 9.4.7. *Suppose that the weights $\boldsymbol{w}$ are integer with $\|w\|_\infty \leq M$. Then, we can find*

$$S \in \arg\min_{S\in\mathcal{I}_1\cap\mathcal{I}_2} w(S)$$

in time $O\left(n\mathrm{GO}\log(nM) + n^3 \log^{O(1)}(nM)\right)$ where GO is the cost of greedy method for $\mathcal{I}_1$ and $\mathcal{I}_2$.

Proof. Applying Lemma 9.4.6, we can find a new cost $\boldsymbol{z}$ such that

$$\min_{\boldsymbol{x}\in\mathrm{conv}(\mathcal{I}_1)\cap\mathrm{conv}(\mathcal{I}_2)} \boldsymbol{z}^T\boldsymbol{x}$$

has an unique solution. Note that for any $\boldsymbol{x} \in \mathrm{conv}(\mathcal{I}_1)$, we have $\|\boldsymbol{x}\|_\infty \leq 1$. Hence, applying theorem 9.4.4, we can find $\boldsymbol{q}$ such that $\boldsymbol{q}^T\boldsymbol{z} \leq \mathrm{OPT}+\varepsilon$ and $\|\boldsymbol{q}-\boldsymbol{x}\|_2 + \|\boldsymbol{q}-\boldsymbol{y}\|_2 \leq \varepsilon$ for some $\boldsymbol{x} \in \mathrm{conv}(\mathcal{I}_1)$ and $\boldsymbol{y} \in \mathrm{conv}(\mathcal{I}_2)$. Using (9.20), we have the coordinate wise minimum of $\boldsymbol{x}, \boldsymbol{y}$, i.e. $\min\{\boldsymbol{x},\boldsymbol{y}\}$, is in $\mathrm{conv}(\mathcal{I}_1) \cap \mathrm{conv}(\mathcal{I}_2)$. Since $\|\boldsymbol{q} - \min\{\boldsymbol{x},\boldsymbol{y}\}\|_2 \leq \|\boldsymbol{q}-\boldsymbol{x}\|_2 + \|\boldsymbol{q}-\boldsymbol{y}\|_2 \leq \varepsilon$, we have

$$(\min\{\boldsymbol{x},\boldsymbol{y}\})^T \boldsymbol{z} \leq \mathrm{OPT} + nM\varepsilon.$$

Hence, we have a feasible point $\min\{\boldsymbol{x}, \boldsymbol{y}\}$ which has value close to optimal and Lemma 9.4.6 shows that $\|\min(\boldsymbol{x}, \boldsymbol{y}) - \boldsymbol{s}\|_\infty \leq 2n^2M^2\varepsilon$ where $\boldsymbol{s}$ is the optimal solution. Hence, we have $\|\boldsymbol{q} - \boldsymbol{s}\|_\infty \leq 2n^2M^2\varepsilon + \varepsilon$. Picking $\varepsilon = \frac{1}{6n^2M^2}$, we have $\|\boldsymbol{q} - \boldsymbol{s}\|_\infty < \frac{1}{2}$ and hence, we can get the optimal solution by rounding to the nearest integer.

Since optimization over $\mathcal{I}_1$ and $\mathcal{I}_2$ involves applying greedy method on certain vectors, it takes only $O(\mathrm{GO})$ time. Theorem 9.4.4 shows it only takes $O\left(n\mathrm{GO}\log(nM) + n^3\log^{O(1)}(nM)\right)$ in finding such $\boldsymbol{q}$. □

This gives the following corollary.

Corollary 9.4.8. *We have $O(n\min(n\mathcal{T}_{ind}, r\mathcal{T}_{rank})\log(nM) + n^3\log^{O(1)} nM)$ time algorithm for weighted matroid intersection. Here $\mathcal{T}_{ind}$ is the time needed to check if a subset is independent, and $\mathcal{T}_{rank}$ is the time needed to compute the rank of a given subset.*

Proof. By Theorem 9.4.7, it suffices to show that the optimization oracle for the matroid polytope can be implemented in $O(n\mathcal{T}_{\mathrm{ind}})$ and $O(r\mathcal{T}_{\mathrm{rank}}\log n)$ time. This is simply attained by the well-known greedy algorithm which iterates through all the positively-weighted elements in decreasing order, and adds an element to our candidate independent set whenever possible.

For the independence oracle, this involves one oracle call for each element. On the other hand, for the rank oracle, we can find the next element to add by binary search which takes time $O(\mathcal{T}_{\mathrm{rank}}\log n)$. Since there are at most r elements to add, we have the desired running time. □

9.4.3 Submodular Flow

Let $G = (V, E)$ be a directed graphwith $|E| = m$, let f be a submodular function on $\mathbb{R}^V$ with $|V| = n$, $f(\emptyset) = 0$ and $f(V) = 0$, and let A be the incidence matrix of G. In this section we consider the submodular flow problem

$$\begin{aligned} \text{Minimize} \quad & \langle c, \varphi\rangle \\ \text{subject to} \quad & l(e) \leq \varphi(e) \leq u(e) \quad \forall e \in E \\ & x(v) = (A\varphi)(v) \quad \forall v \in V \\ & \sum_{v\in S} x(v) \leq f(S) \quad \forall S \subseteq V \end{aligned} \tag{9.21}$$

where $c \in \mathbb{Z}^E$, $l \in \mathbb{Z}^E$, $u \in \mathbb{Z}^E$ where $C = \|\boldsymbol{c}\|_\infty$ and $U = \max(\|u\|_\infty, \|l\|_\infty, \max_{S\subset V}|f(S)|)$. Here c is the cost on edges, φ is the flow on edges, l and u are lower and upper bounds on the amount of flow on the edges, and $x(v)$ is the net flow out of vertex v. The submodular function f upper bounds the total net flow out of any subset S of vertices by $f(S)$.

Theorem 9.4.9. *Suppose that the cost vector $\boldsymbol{c}$ is integer weight with $\|\boldsymbol{c}\|_\infty \leq C$ and the capacity vector and the submodular function satisfy $U = \max(\|u\|_\infty, \|l\|_\infty, \max_{S\subset V}|f(S)|)$. Then, we can solve the submodular flow problem (9.21) in time $O\left(n^2\mathrm{EO}\log(mCU) + n^3\log^{O(1)}(mCU)\right)$ where EO is the cost of function evaluation.*

Proof. First, we can assume $l(e) \leq u(e)$ for every edge e, otherwise, the problem is infeasible. Now, we apply a similar transformation in [110] to modify the graph. We create a new vertex v_0. For every vertex v in V, we create a edge from v_0 to v with capacity lower bounded by 0, upper bounded by $4nU$, and with cost $2mCU$. Edmonds and Giles showed that the submodular flow polytope is integral [83]. Hence, there is an integral optimal flow on this new graph. If the optimal flow passes through the

newly created edge, then it has cost at least $2mCU - mCU$ because the cost of all other edges in total has at least $-mCU$. That means the optimal flow has the cost larger than mCU which is impossible. So the optimal flow does not use the newly created edges and vertex and hence the optimal flow in the new problem gives the optimal solution of the original problem. Next, we note that for any φ on the original graph such that $l(e) \le \varphi(e) \le u(e)$, we can send suitable amount of flow from v_0 to v to make φ feasible. Hence, this modification makes the feasibility problem trivial.

Lemma 9.4.6 shows that we can assume the new problem has an unique solution and it only blows up C by a $(mU)^{O(1)}$ factors.

Note that the optimal value is an integer and its absolute value at most mCU. By binary search, we can assume we know the optimal value OPT. Now, we reduce the problem to finding a feasible φ with $\{\langle d, \varphi\rangle \le \mathrm{OPT} + \varepsilon\}$ with ε determined later. Let P_ε be the set of such φ. Note that $P_\varepsilon = K_{1,\varepsilon} \cap K_{2,\varepsilon}$ where

$$\begin{aligned} K_{1,\varepsilon} &= \left\{ x \in \mathbb{R}^V \text{ such that } \begin{array}{c} l(e) \le \varphi(e) \le u(e) \quad \forall e \in E \\ x(v) = (A\varphi)(v) \quad \forall v \in V \\ \langle d, \varphi\rangle \le \mathrm{OPT} + \varepsilon \end{array} \text{ for some } \varphi \right\}, \\ K_{2,\varepsilon} &= \left\{ y \in \mathbb{R}^V \text{ such that } \sum_{v\in S} y(v) \le f(S) \quad \forall S \subseteq V, \sum_{v \in V} y(v) = f(V) \right\}. \end{aligned}$$

Note that the extra condition $\sum_v y(v) = f(V)$ is valid because $\sum_v y(v) = \sum_v (A\varphi)(v) = 0$ and $f(V) = 0$, and $K_{1,\varepsilon}$ has radius bounded by $O((mCU)^{O(1)})$ and $K_{2,\varepsilon}$ has radius bounded by $O(nU)$. Furthermore, for any vector $\boldsymbol{c} \in \mathbb{R}^V$, we note that

$$\begin{aligned} \max_{x\in K_{1,\varepsilon}} \langle c, x\rangle &= \max_{l\le\varphi\le u, \langle d,\varphi\rangle \le \mathrm{OPT}+\varepsilon, x = A\varphi} \langle c, x\rangle \\ &= \max_{l\le\varphi\le u, \langle d,\varphi\rangle \le \mathrm{OPT}+\varepsilon} \langle c, A\varphi\rangle \\ &= \max_{l\le\varphi\le u, \langle d,\varphi\rangle \le \mathrm{OPT}+\varepsilon} \left\langle A^T c, \varphi\right\rangle. \end{aligned}$$

To solve this problem, again we can do a binary search on $\langle d, \varphi\rangle$ and reduce the problem to

$$\max_{l\le\varphi\le u, \langle d,\varphi\rangle = K} \left\langle A^T c, \varphi\right\rangle$$

for some value of K. Since $A^T c$ is fixed, this is a linear program with only the box constraints and an extra equality constraint. Hence, it can be solved in nearly linear time [164, Thm 17, ArXiv v1]. As the optimization oracle for $K_{1,\varepsilon}$ involves only computing $A^T c$ and solving this simple linear program, it takes only $O(n^2 \log^{O(1)}(mCU/\varepsilon))$ time. On the other hand, since $K_{2,\varepsilon}$ is just a base polyhedron, the optimization oracle for $K_{2,\varepsilon}$ can be done by greedy method and only takes $O(n\mathrm{EO})$ time.

Applying Theorem 9.4.4, we can find q such that $\|q - x\|_2 + \|q - y\|_2 \le \delta$ for some $x \in K_{1,\varepsilon}$, $y \in K_{2,\varepsilon}$ and δ to be chosen later. According to the definition of $K_{1,\varepsilon}$, there is φ such that $l(e) \le \varphi(e) \le u(e)$ and $x(v) = (A\varphi)(v)$ for all v and $\langle d, \varphi\rangle \le \mathrm{OPT} + \varepsilon$. Since $\|y - x\|_2 \le 2\delta$, that means $|y(v) - (A\varphi)(v)| \le 2\delta$ for all v.

- Case 1) If $y(v) \ge (A\varphi)(v)$, then we can replace $y(v)$ by $(A\varphi)(v)$, note that y is still in $K_{2,\varepsilon}$ because of the submodular constraints.
- Case 2) If $y(v) \le (A\varphi)(v)$, then we can send a suitable amount of flow from v_0 to v to make φ feasible $y(v) \le (A\varphi)(v)$.

Note that under this modification, we increased the objective value by $(\delta n)(2mCU)$ because the new edge cost $2mCU$ per unit of flow. Hence, we find a flow φ which is feasible in new graph with objective

value $\varepsilon + (\delta n)(2mCU)$ far from optimum value. By picking $\delta = \frac{1}{2mnCU}$, we have the value 2ε far from OPT. Now, we use Lemma 9.4.6 to shows that when ε is small enough, i.e, $\frac{1}{(mCU)^c}$ for some constant c, then we can guarantee that $\|y - x^*\|_\infty \le \frac{1}{4}$ where x^* is the optimal demand. Now, we note that $\|q - y\|_2 \le \delta$ and we note that we only modify y by a small amount, we in fact have $\|q - x^*\|_\infty < \frac{1}{2}$. Hence, we can read off the solution x^* by rounding q to the nearest integer. Note that we only need to solve the problem $K_{1,\varepsilon} \cap K_{2,\varepsilon}$ to $\frac{1}{(mCU)^{\Theta(1)}}$ accuracy and the optimization oracle for $K_{1,\varepsilon}$ and $K_{2,\varepsilon}$ takes time $O(n^2 \log^{O(1)}(mCU))$ and $O(n\mathrm{EO})$ respectively. Hence, Theorem 9.4.4 shows that it takes $O\left(n^2\mathrm{EO}\log(mCU) + n^3\log^{O(1)}(mCU)\right)$ time to find x^* exactly.

After getting x^*, one can find φ^* by solving a min cost flow problem using interior point method (Chapter 6), which takes $O(m\sqrt{n}\log^{O(1)}(mCU))$ time. □

9.4.4 Affine Subspace of Convex Set

In this section, we give another example about using optimization oracle directly via regularization. We consider the following optimization problem

$$\max_{\boldsymbol{x}\in K \text{ and } \boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}} \langle \boldsymbol{c}, \boldsymbol{x}\rangle \tag{9.22}$$

where $\boldsymbol{x}, \boldsymbol{c} \in \mathbb{R}^n$, K is a convex subset of $\mathbb{R}^n$, $\boldsymbol{A} \in \mathbb{R}^{r\times n}$ and $\boldsymbol{b} \in \mathbb{R}^m$. We suppose that $r \ll n$ and thus, the goal of this subsection is to show how to obtain an algorithm takes only $\tilde{O}(r)$ many iterations. To do this, we assume a slightly stronger optimization oracle for K:

Definition 9.4.10. Given a convex set K and $\delta > 0$. A δ-2nd-order-optimization oracle for K is a function on $\mathbb{R}^n$ such that for any input $\boldsymbol{c} \in \mathbb{R}^n$ and $\lambda > 0$, it outputs $\boldsymbol{y}$ such that

$$\max_{\boldsymbol{x}\in K}\left(\langle \boldsymbol{c}, \boldsymbol{x}\rangle - \lambda\|\boldsymbol{x}\|^2\right) \le \delta + \langle \boldsymbol{c}, \boldsymbol{y}\rangle - \lambda\|\boldsymbol{y}\|^2.$$

We denote by $\mathrm{OO}^{(2)}_{\delta,\lambda}(K)$ the time complexity of this oracle.

The strategy for solving this problem is very similar to the intersection problem and hence some details are omitted.

Theorem 9.4.11. *Assume that* $\max_{\boldsymbol{x}\in K}\|\boldsymbol{x}\|_2 < M$, $\|\boldsymbol{b}\|_2 < M$, $\|\boldsymbol{c}\|_2 < M$, $\|\boldsymbol{A}\|_2 < M$ *and* $\lambda_{\min}(\boldsymbol{A}) > 1/M$. *Assume that* $K \cap \{\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}\} \neq \emptyset$ *and we have* ε*-2nd-order-optimization oracle for every* $\varepsilon > 0$. *For* $0 < \delta < 1$, *we can find* $\boldsymbol{z} \in K$ *such that*

$$\max_{\boldsymbol{x}\in K \text{ and } \boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}} \langle \boldsymbol{c}, \boldsymbol{x}\rangle \le \delta + \langle \boldsymbol{c}, \boldsymbol{z}\rangle$$

and $\|\boldsymbol{A}\boldsymbol{z} - \boldsymbol{b}\|_2 \le \delta$. *This algorithm takes time*

$$O\left(r\mathrm{OO}^{(2)}_{\eta,\lambda}(K)\log\left(\frac{nM}{\delta}\right) + r^3\log^{O(1)}\left(\frac{nM}{\delta}\right)\right)$$

where r *is the number of rows in* $\boldsymbol{A}$, $\eta = \left(\frac{\delta}{nM}\right)^{\Theta(1)}$ *and* $\lambda = \left(\frac{\delta}{nM}\right)^{\Theta(1)}$.

Proof. The proof is based on the minimax problem

$$\mathrm{OPT}_\lambda \stackrel{\text{def}}{=} \min_{\|\boldsymbol{\eta}\|_2 \le 1/\lambda} \max_{\boldsymbol{x}\in K} \langle \boldsymbol{c}, \boldsymbol{x}\rangle + \langle \boldsymbol{\eta}, \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\rangle - \frac{1}{\lambda}\|\boldsymbol{x}\|_2^2$$

where $\lambda = \left(\frac{\delta}{nM}\right)^c$ for some large constant c. We note that

$$\begin{aligned} \mathrm{OPT}_\lambda &= \max_{\boldsymbol{x}\in K} \min_{\|\boldsymbol{\eta}\|_2 \le 1/\lambda} \langle \boldsymbol{c}, \boldsymbol{x}\rangle + \langle \boldsymbol{\eta}, \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\rangle - \frac{1}{\lambda}\|\boldsymbol{x}\|_2^2 \\ &= \max_{\boldsymbol{x}\in K} \langle \boldsymbol{c}, \boldsymbol{x}\rangle - \frac{1}{\lambda}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2 - \frac{1}{\lambda}\|\boldsymbol{x}\|_2^2. \end{aligned}$$

Since $\lambda_{\min}(\boldsymbol{A}) > 1/M$ and the set K is bounded by M, one can show that the saddle point $(\boldsymbol{x}^*, \boldsymbol{\eta}^*)$ of the minimax problem gives a good enough solution $\boldsymbol{x}$ for the original problem for large enough constant c.

For any $\boldsymbol{\eta}$, we define

$$\boldsymbol{x}_{\boldsymbol{\eta}} = \arg\max_{\boldsymbol{x}\in K} \langle \boldsymbol{c}, \boldsymbol{x}\rangle + \langle \boldsymbol{\eta}, \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\rangle - \frac{1}{\lambda}\|\boldsymbol{x}\|_2^2.$$

Since the problem is strongly concave in $\boldsymbol{x}$, one can prove that

$$\|\boldsymbol{x}_{\boldsymbol{\eta}} - \boldsymbol{x}^*\|_2 \le \left(\frac{nM}{\delta}\right)^{O(1)} \|\boldsymbol{\eta} - \boldsymbol{\eta}^*\|_2.$$

Hence, we can first find an approximate minimizer of the function $f(\boldsymbol{\eta}) = \max_{\boldsymbol{x}\in K} \langle \boldsymbol{c}, \boldsymbol{x}\rangle + \langle \boldsymbol{\eta}, \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\rangle - \frac{1}{\lambda}\|\boldsymbol{x}\|_2^2$ and use the oracle to find $\boldsymbol{x}_{\boldsymbol{\eta}}$.

To find an approximate minimizer of f, we note that the subgradient of f can be found using the optimization oracle similar to Theorem 9.4.4. Hence, the result follows from our cutting plane method and the fact that $\boldsymbol{\eta} \in \mathbb{R}^r$. □

Remark 9.4.12. In Chapter 6, we considered the special case $K = \{\boldsymbol{x} : 0 \le x_i \le 1\}$ and showed that it can be solved in $\tilde{O}(\sqrt{r})$ iterations using interior point methods. This gives the current fastest algorithm for the maximum flow problem on directed weighted graphs. This result generalizes that result to any convex set K but with $\tilde{O}(r)$ iterations. This suggests the following open problem: under what condition on K can one optimize linear functions over affine subspaces of K with r constraints in $\tilde{O}(\sqrt{r})$ iterations?

Chapter 10

Submodular Minimization In Nearly-Cubic Time

10.1 Introduction

Submodular functions and submodular function minimization (SFM) are fundamental to the field of combinatorial optimization. Examples of submodular functions include graph cut functions, set coverage function, and utility functions from economics. Since the seminal work by Edmonds in 1970 [82], submodular functions and the problem of minimizing such functions (i.e. submodular function minimization) have served as a popular modeling and optimization tool in various fields such as theoretical computer science, operations research, game theory, and most recently, machine learning. Given its prevalence, fast algorithms for SFM are of immense interest both in theory and in practice.

Throughout this chapter, we consider the standard formulation of SFM: we are given a submodular function f defined over the subsets of a n-element ground set. The values of f are integers, have absolute value at most M, and are evaluated by querying an oracle that takes time EO. Our goal is to produce an algorithm that solves this SFM problem, i.e. finds a minimizer of f, while minimizing both the number of oracle calls made and the total running time.

We provide new $O(n^2 \log nM \cdot \text{EO} + n^3 \log^{O(1)} nM)$ and $O(n^3 \log^2 n \cdot \text{EO} + n^4 \log^{O(1)} n)$ time algorithms for SFM. These algorithms improve upon the previous fastest weakly and strongly polynomial time algorithms for SFM which had a a running time of $O((n^4 \cdot \text{EO} + n^5) \log M)$ [118] and $O(n^5 \cdot \text{EO} + n^6)$ [216] respectively. Consequently, we improve the running times in both regimes by roughly a factor of $O(n^2)$.

Both of our algorithms bear resemblance to the classic approach of Grötschel, Lovász and Schrijver [110, 109] using the Lovász extension. In fact our weakly polynomial time algorithm directly uses the Lovász extension as well as the results of Chapter 9 to achieve these results. Our strongly polynomial time algorithm also uses the Lovász extension, along with more modern tools from the past 15 years.

At a high level, our strongly polynomial algorithms apply our cutting plane method in conjunction with techniques originally developed by Iwata, Fleischer, and Fujishige (IFF) [121]. Our cutting plane method is performed for enough iterations to sandwich the feasible region in a narrow strip from which useful structural information about the minimizers can be deduced. Our ability to derive the new information hinges on a significant extension of IFF techniques.

Over the past few decades, SFM has drawn considerable attention from various research communities, most recently in machine learning [24, 149]. Given this abundant interest in SFM, we hope that our ideas will be of value in various practical applications. Indeed, one of the critiques against existing theoretical algorithms is that their running time is too slow to be practical. Our contribution, on the contrary, shows that this school of algorithms can actually be made fast theoretically and we hope it may potentially be competitive against heuristics which are more commonly used.

10.1.1 Previous Work

Here we provide a brief survey of the history of algorithms for SFM. For a more comprehensive account of the rich history of SFM, we refer the readers to recent surveys [189, 120].

The first weakly and strongly polynomial time algorithms for SFM were based on the ellipsoid method [142] and were established in the foundational work of Grötschel, Lovász and Schrijver in 1980's [110, 109]. Their work was complemented by a landmark paper by Cunningham in 1985 which provided a pseudopolynomial algorithm that followed a flow-style algorithmic framework [61]. His tools foreshadowed much of the development in SFM that would take place 15 years later. Indeed, modern algorithms synthesize his framework with inspirations from various max flow algorithms.

The first such "flow style" strongly polynomial algorithms for SFM were discovered independently in the breakthrough papers by Schrijver [231] and Iwata, Fleischer, and Fujishige (IFF) [121]. Schrijver's algorithm has a running of $O(n^8 \cdot \mathrm{EO} + n^9)$ and borrows ideas from the push-relabel algorithms [105, 68] for the maximum flow problem. On the other hand, IFF's algorithm runs in time $O(n^7 \log n \cdot \mathrm{EO})$ and $O(n^5 \cdot \mathrm{EO} \log M)$, and applies a flow-scaling scheme with the aid of certain proximity-type lemmas as in the work of Tardos [247]. Their method has roots in flow algorithms such as [117, 106].

Subsequent work on SFM provided algorithms with considerably faster running time by extending the ideas in these two "genesis" papers [231, 121] in various novel directions [264, 87, 118, 216, 125]. Currently, the fastest weakly and strongly polynomial time algorithms for SFM have a running time of $O((n^4 \cdot \mathrm{EO} + n^5) \log M)$ [118] and $O(n^5 \cdot \mathrm{EO} + n^6)$ [216] respectively. Despite this impressive track record, the running time has not been improved in the last eight years.

We remark that all of the previous algorithms for SFM proceed by maintaining a convex combination of $O(n)$ BFS's of the base polyhedron, and incrementally improving it in a relatively local manner. As we shall discuss in Section 10.1.2, our algorithms do not explicitly maintain a convex combination. This may be one of the fundamental reasons why our algorithms achieve a faster running time.

Finally, beyond the distinction between weakly and strongly polynomial time algorithms for SFM, there has been interest in another type of SFM algorithm, known as fully combinatorial algorithms in which only additions and subtractions are permitted. Previous such algorithms include [125, 118, 119]. We do not consider such algorithms in the remainder of this chapter and leave it as an open question if it is possible to turn our algorithms into fully combinatorial ones.

10.1.2 Our Results and Techniques

In this chapter, we show how to improve upon the previous best known running times for SFM by a factor of $O(n^2)$ in both the strongly and weakly polynomial regimes. In Table 10.1 summarizes the running time of the previous algorithms as well as the running times of the fastest algorithms presented in this chapter.

Both our weakly and strongly polynomial algorithms for SFM utilize a convex relaxation of the submodular function, called the *Lovász extension*. Our algorithms apply our cutting plane method from Chapter 8 using a separation oracle given by the subgradient of the Lovász extension. To the best of the author's knowledge, Grötschel, Lovász and Schrijver were the first to formulate this convex optimization framework for SFM [110, 109].

For weakly polynomial algorithms, our contribution is two-fold. First, we show that cutting plane methods such as Vaidya's [253] can be applied to SFM to yield faster algorithms. Second, as our cutting plane method, Theorem 9.3.2, improves upon previous cutting plane algorithms and consequently the running time for SFM as well. This gives a running time of $O(n^2 \log nM \cdot \mathrm{EO} + n^3 \log^{O(1)} nM)$, an improvement over the previous best algorithm by Iwata [118] by a factor of almost $O(n^2)$.

Our strongly polynomial algorithms, on the other hand, require substantially more innovation. We

Years	Authors	Running times	Remarks
1981,1988	Grötschel, Lovász,	$\widetilde{O}(n^5 \cdot \mathrm{EO} + n^7)$[189]	first weakly and strongly
1985	Cunningham [61]	$O(Mn^6 \log nM \cdot \mathrm{EO})$	first pseudopoly
2000	Schrijver [231]	$O(n^8 \cdot \mathrm{EO} + n^9)$	first combin. strongly
2000	Iwata, Fleischer,	$O(n^5 \cdot \mathrm{EO} \log M)$ $O(n^7 \log n \cdot \mathrm{EO})$	first combin. strongly
2000	Iwata, Fleischer [87]	$O(n^7 \cdot \mathrm{EO} + n^8)$	
2003	Iwata [118]	$O((n^4 \cdot \mathrm{EO} + n^5) \log M)$ $O((n^6 \cdot \mathrm{EO} + n^7) \log n)$	current best weakly
2003	Vygen [264]	$O(n^7 \cdot \mathrm{EO} + n^8)$	
2007	Orlin [216]	$O(n^5 \cdot \mathrm{EO} + n^6)$	current best strongly
2009	Iwata, Orlin [125]	$O((n^4 \cdot \mathrm{EO} + n^5) \log nM)$ $O((n^5 \cdot \mathrm{EO} + n^6) \log n)$	
-	This Chapter	$O(n^2 \log nM \cdot \mathrm{EO} + n^3 \log^{O(1)} nM)$ $O(n^3 \log^2 n \cdot \mathrm{EO} + n^4 \log^{O(1)} n)$	

Table 10.1: Algorithms for submodular function minimization. Note that some of these algorithms were published in both conferences and journals, in which case the year we provided is the earlier one.

first begin with a very simple geometric argument that SFM can be solved in $O(n^3 \log n \cdot \mathrm{EO})$ oracle calls (but in exponential time). This proof only uses Grunbaum's Theorem from convex geometry and is completely independent from the rest of this chapter. It was the starting point of our method and suggests that a running time of $\widetilde{O}(n^3 \cdot \mathrm{EO} + n^{O(1)})$ for submodular minimization is in principle achievable.

To make this existence result algorithmic, we first run cutting plane, Theorem 8.3.26, for enough iterations such that we compute either a minimizer or a set P containing the minimizers that fits within in a narrow strip. This narrow strip consists of the intersection of two approximately parallel hyperplanes. If our narrow strip separates P from one of the faces $x_i = 0$, $x_i = 1$, we can effectively eliminate the element i from our consideration and reduce the dimension of our problem by 1. Otherwise a pair of elements p, q can be identified for which q is guaranteed to be in any minimizer containing p (but p may not be contained in a minimizer). Our first algorithm deduces only one such pair at a time. This technique immediately suffices to achieve a $\widetilde{O}(n^4 \cdot \mathrm{EO} + n^5)$ time algorithm for SFM (See Section 10.4.3). We then improve the running time to $\widetilde{O}(n^3 \cdot \mathrm{EO} + n^4)$ by showing how to deduce many such pairs simultaneously. Similar to past algorithms, this structural information is deduced from a point in the so-called base polyhedron (See Section 10.2).

Readers well-versed in SFM literature may recognize that our strongly polynomial algorithms are reminiscent of the scaling-based approach first used by IFF [121] and later in [118, 125]. While both approaches share the same skeleton, there are differences as to how structural information about minimizers is deduced. A comparison of our algorithms and previous ones are presented in Section 10.5.

Finally, there is one more crucial difference between these algorithms which we believe is responsible for much of our speedup. One common feature shared by all the previous algorithms is that they maintain a convex combination of $O(n)$ BFS's of the base polyhedron, and incrementally improve on it by introducing new BFS's by making local changes to existing ones. Our algorithms, on the other hand, choose new BFS's by the cutting plane method. Because of this, our algorithm considers the geometry of the existing BFS's where each of them has influences over the choice of the next BFS. In some sense, our next BFS is chosen in a more "global" manner.

10.1.3 Organization

The rest of this chapter is organized as follows. We first begin with a gentle introduction to submodular functions in Section 10.2. In Section 10.3, we apply our cutting plane method to SFM to obtain a faster weakly polynomial algorithms. In Section 10.4 we then present our results for achieving better strongly polynomial algorithms, where a warm-up $\widetilde{O}(n^4 \cdot \mathrm{EO} + n^5)$ algorithm is given before the full-fledged $\widetilde{O}(n^3 \cdot \mathrm{EO} + n^4)$ algorithm. Finally, we end this chaper with a discussion and comparison between our algorithms and previous ones in Section 10.5.

We note that there are a few results in this chapter that can be read fairly independently of the rest of this chapter. In Theorem *10.3.2* we show how Vaidya's algorithm can be applied to SFM to obtain a faster weakly polynomial running time. Also in Theorem 10.4.1 we present a simple geometric argument that SFM can be solved with $O(n^3 \log n \cdot \mathrm{EO})$ oracle calls but with exponential time. These results can be read with only a working knowledge of the Lovász extension of submodular functions.

10.2 Preliminaries

Here we introduce background on submodular function minimization (SFM) and notation that we use throughout this chapter. Our exposition is kept to a minimal amount sufficient for our purposes. We refer interested readers to the extensive survey by McCormick [189] for further intuition.

10.2.1 Submodular Function Minimization

Throughout the rest of this chapter, let $V = \{1, ..., n\} = [n]$ denote a ground set and let $f : 2^V \longrightarrow \mathbb{Z}$ denote a *submodular function* defined on subsets of this ground set. We use V and $[n]$ interchangeably and let $[0] \stackrel{\text{def}}{=} \emptyset$. We abuse notation by letting $S + i \stackrel{\text{def}}{=} S \cup \{i\}$ and $S - i \stackrel{\text{def}}{=} S \backslash \{i\}$ for an element $i \in V$ and a set $S \subseteq 2^V$. Formally, we call a function submodular if it obeys the following property of *diminishing marginal differences*:

Definition 10.2.1 (Submodularity)**.** A function $f : 2^V \longrightarrow \mathbb{Z}$ is *submodular* if $f(T + i) - f(T) \leq f(S + i) - f(S)$ for any $S \subseteq T$ and $i \in V \backslash T$.

For convenience we assume without loss of generality that $f(\emptyset) = 0$ by replacing $f(S)$ by $f(S) - f(\emptyset)$ for all S. We also let $M \stackrel{\text{def}}{=} \max_{S \in 2^V} |f(S)|$.

The central goal of this chapter is to design algorithms for SFM, i.e. computing the minimizer of f. We call such an algorithm *strongly polynomial* if its running time depends only polynomially on n and EO, the time needed to compute $f(S)$ for a set S, and we call such an algorithm *weakly polynomial* if it also depends polylogarithmically on M.

10.2.2 Lovász Extension

Our new algorithms for SFM all consider a convex relaxation of a submodular function, known as the Lovász extension, and then carefully apply our cutting plane methods to it. Here we formally introduce the Lovász extension and present basic facts that we use throughout this chapter.

The Lovász extension of $\hat{f} : [0, 1]^n \longrightarrow \mathbb{R}$ of our submodular function f is defined for all $\boldsymbol{x}$ by

$$\hat{f}(\boldsymbol{x}) \stackrel{\text{def}}{=} \mathbb{E}_{t \sim [0,1]}[f(\{i : x_i \geq t\})],$$

where $t \sim [0, 1]$ is drawn uniformly at random from $[0, 1]$. The Lovász extension allows us to reduce SFM to minimizing a convex function defined over the interior of the hypercube. Below we state that the Lovász extension is a convex relaxation of f and that it can be evaluated efficiently.

Theorem 10.2.2. *The Lovász extension $\hat{f}$ satisfies the following properties:*

1. *$\hat{f}$ is convex and $\min_{\boldsymbol{x}\in[0,1]^n} \hat{f}(\boldsymbol{x}) = \min_{S\subset[n]} f(S)$;*
2. *$f(S) = \hat{f}(I_S)$, where I_S is the characteristic vector for S, i.e. $I_S(i) = \begin{cases} 1 & \text{if } i \in S \\ 0 & \text{if } i \notin S \end{cases}$;*
3. *If S is a minimizer of f, then I_S is a minimizer of $\hat{f}$;*
4. *Suppose $x_1 \geq \cdots \geq x_n \geq x_{n+1} \stackrel{\text{def}}{=} 0$, then*

$$\hat{f}(\boldsymbol{x}) = \sum_{i=1}^{n} f([i])(x_i - x_{i+1}) = \sum_{i=1}^{n} (f([i]) - f([i-1]))x_i \,.$$

Proof. See [109] or any standard textbook on combinatorial optimization, e.g. [232]. □

Next we show that we can efficiently compute a subgradient of the Lovász or alternatively, a separating hyperplane for the set of minimizers of our submdoular function f. First we remind the reader of the definition of a separation oracle, and then we prove the necessary properties of the hyperplane, Theorem 10.2.4.

Definition 10.2.3 (separation oracle, Defintion 2.3.5 restated for Lovász extension)**.** Given a point $\bar{x}$ and a convex function $\hat{f}$ over a convex set P, $\boldsymbol{a}^T\boldsymbol{x} \leq b$ is a separating hyperplane if $\boldsymbol{a}^T\bar{x} \geq b$ and any minimizer x^* of $\hat{f}$ over P satisfies $\boldsymbol{a}^T x^* \leq b$.

Theorem 10.2.4. *Given a point $\bar{x} \in [0,1]^n$ assume without loss of generality (by re-indexing the coordinates) that $\bar{x}_1 \geq \cdots \geq \bar{x}_n$. Then the following inequality is a valid separating hyperplane for $\boldsymbol{x}$ and f:*

$$\sum_{i=1}^{n} (f([i]) - f([i-1]))x_i \leq \hat{f}(\bar{x})$$

i.e., it satisfies the following:

1. *(separating) $\bar{x}$ lies on $\sum_{i=1}^{n}(f([i]) - f([i-1]))x_i \leq \hat{f}(\bar{x})$.*
2. *(valid) For any $\boldsymbol{x}$, we have $\sum_{i=1}^{n}(f([i]) - f([i-1]))x_i \leq \hat{f}(\boldsymbol{x})$. In particular, $\sum_{i=1}^{n}(f([i]) - f([i-1]))x_i^* \leq \hat{f}(\bar{x})$ for any minimizer $\boldsymbol{x}^*$, i.e. the separating hyperplane does not cut out any minimizer.*

Moreover, such a hyperplane can be computed with n oracle calls to f and in time $O(n \cdot EO + n^2)$.

Proof. Note that by Theorem 10.2.2 we have that $\sum_{i\in[n]}(f([i]) - f([i-1]))x_i = \hat{f}(\bar{x})$ and thus the hyperplane satisfies the separating condition. Moreover, clearly computing it only takes time $O(n \cdot \mathrm{EO} + n^2)$ as we simply need to sort the coordinates and evaluate f at n points, i.e. each of the $[i]$. All that remains is to show that the hyperplane satisfies the valid condition.

Let $L^{(t)} \stackrel{\text{def}}{=} \{i : x_i \geq t\}$. Recall that $\hat{f}(\boldsymbol{x}) = \mathbb{E}_{t\sim[0,1]}[f(L_t)]$. Thus $\hat{f}(\boldsymbol{x})$ can be written as a convex combination $\hat{f}(\boldsymbol{x}) = \sum_t \alpha_t f(L^{(t)})$, where $\alpha_t \geq 0$ and $\sum_t \alpha_t = 1$. However, by diminishing marginal

differences we see that for all t

$$\begin{aligned}\sum_{i\in[n]}(f([i])-f([i-1]))\left(I_{L^{(t)}}\right)_i &= \sum_{i\in L^{(t)}}(f([i])-f([i-1]))\\ &\le \sum_{i\in L^{(t)}}\left(f([i]\cap L^{(t)})-f([i-1]\cap L^{(t)})\right)\\ &= f(L^{(t)})-f(\emptyset)=f(L^{(t)})\end{aligned}$$

and therefore since $\sum_t \alpha_t I_{L^{(t)}} = \boldsymbol{x}$ we have

$$\sum_{i\in[n]}(f[i]-f([i-1])x_i = \sum_t \alpha_t \sum_{i=1}^{n}(f([i])-f([i-1]))\left(I_{L^{(t)}}\right)_i \le \sum_t \alpha_t f(L^{(t)}) = \hat{f}(\boldsymbol{x}).$$

□

10.2.3 Polyhedral Aspects of SFM

Here we provide a natural primal dual view of SFM that we use throughout the analysis. We provide a dual convex optimization program to minimizing the Lovász extension and provide several properties of these programs. We believe the material in this section helps crystallize some of the intuition behind our algorithm and we make heavy use of the notation presented in this section. However, we will not need to appeal to the strong duality of these programs in our proofs.

Consider the following primal and dual programs, where we use the shorthands $y(S) = \sum_{i\in S} y_i$ and $y_i^- = \min\{0, y_i\}$. Here the primal constraints are often called the *base polyhedron* $\mathcal{B}(f) \stackrel{\text{def}}{=} \{\boldsymbol{y} \in \mathbb{R}^n : y(S) \le f(S) \forall S \subsetneq V, y(V) = f(V)\}$ *and the dual program directly corresponds to minimizing the Lovász extension and thus* f.

Primal	Dual
$\max y^-(V)$ $y(S) \le f(S) \forall S \subsetneq V$ $y(V) = f(V)$	$\min \hat{f}(\boldsymbol{x})$ $0 \le \boldsymbol{x} \le 1$

Theorem 10.2.5. *$\boldsymbol{h}$ is a basic feasible solution (BFS) of the base polyhedron $\mathcal{B}(f)$ if and only if*

$$h_i = f(\{v_1, ..., v_i\}) - f(\{v_1, ..., v_{i-1}\})$$

for some permutation $v_1, ..., v_n$ of the ground set V. We call $v_1, ..., v_n$ the defining permutation of $\boldsymbol{h}$. We call v_i precedes v_j for $i < j$.

This theorem gives a nice characterization of the BFS's of $\mathcal{B}(f)$. It also gives the key observation underpinning our approach: **the coefficients of each separating hyperplane in Theorem 10.2.4 precisely corresponds to a primal BFS (Theorem 10.2.5)**. Our analysis relies heavily on this connection. We re-state Theorem 10.2.4 in the language of BFS.

Lemma 10.2.6. *We have $\boldsymbol{h}^T\boldsymbol{x} \le \hat{f}(\boldsymbol{x})$ for any $\boldsymbol{x} \in [0,1]^n$ and BFS $\boldsymbol{h}$.*

Proof. Any BFS is given by some permutation. Thus this is just Theorem 10.2.4 in disguise. □

We also note that since the objective function of the primal program is non-linear, we cannot say that the optimal solution to the primal program is a BFS. Instead we only know that it is a convex combination of the BFS's that satisfy the following property. A proof can be found in any standard textbook on combinatorial optimization.

Theorem 10.2.7. *The above primal and dual programs have no duality gap. Moreover, there always exists a primal optimal solution* $\boldsymbol{y} = \sum_k \lambda^{(k)} \boldsymbol{h}^{(k)}$ *with* $\sum_k \lambda^{(k)} = 1$ *(a convex combination of BFS* $\boldsymbol{h}^{(k)}$*) s.t. any* i *with* $y_i < 0$ *precedes any* j *with* $y_j > 0$ *in the defining permutation for each BFS* $\boldsymbol{h}^{(k)}$*.*

Our algorithms will maintain collections of BFS and use properties of $\boldsymbol{h} \in \mathcal{B}(f)$, i.e. convex combination of BFS. To simplify our analysis at several points we will want to assume that such a vector $\boldsymbol{h} \in \mathcal{B}(f)$ is *non-degenerate*, meaning it has both positive and negative entries. Below, we prove that such degenerate points in the base polytope immediately allow us to trivially solve the SFM problem.

Lemma 10.2.8 (Degenerate Hyperplanes)**.** *If* $\boldsymbol{h} \in \mathcal{B}(f)$ *is non-negative then* $\emptyset$ *is a minimizer of* f *and if* $\boldsymbol{h}$ *is non-positive then* V *is a minimizer of* f*.*

Proof. While this follows immediately from Theorem 10.2.7, for completeness we prove this directly. Let $S \in 2^V$ be arbitrary. If $\boldsymbol{h} \in \mathcal{B}(f)$ is non-negative then by the we have

$$f(S) \geq \boldsymbol{h}(S) = \sum_{i \in S} h_i \geq 0 = f(\emptyset)\,.$$

On the other hand if $\boldsymbol{h}$ is non-positive then by definition we have

$$f(S) \geq \boldsymbol{h}(S) = \sum_{i \in S} h_i \geq \sum_{i \in V} h_i = h(V) = f(V)\,.$$

□

10.3 Improved Weakly Polynomial Algorithms for SFM

In this section we show how our cutting plane method can be used to obtain a $O(n^2 \log nM \cdot \mathrm{EO} + n^3 \log^{O(1)} nM)$ time algorithm for SFM. Our main result in this section is the following theorem, which shows how directly applying our results from earlier chapters to minimize the Lovász extension yields the desired running time.

Theorem 10.3.1. *We have an* $O(n^2 \log nM \cdot EO + n^3 \log^{O(1)} nM)$ *time algorithm for submodular function minimization.*

Proof. We apply Theorem 9.3.2 to the Lovász extension $\hat{f} : [0,1]^n \longrightarrow \mathbb{R}$ with the separation oracle given by Theorem 10.2.4. $\hat{f}$ fulfills the requirement on the domain as its domain $\Omega = [0,1]^n$ is symmetric about the point $(1/2, \ldots, 1/2)$ and has exactly $2n$ constraints.

In the language of Theorem 9.3.2, our separation oracle is a $(0,0)$-separation oracle with $\eta = 0$ and $\delta = 0$.

We first show that $\delta = 0$. Firstly, our separating hyperplane can be written as

$$\sum_{i=1}^{n} (f([i]) - f([i-1]))x_i \leq \hat{f}(\bar{x}) = \sum_{i=1}^{n} (f([i]) - f([i-1]))\bar{x}_i,$$

where the equality follows from Theorem 10.2.2. Secondly, for any $\boldsymbol{x}$ with $\hat{f}(\boldsymbol{x}) \leq \hat{f}(\bar{x})$ we have by Theorem 10.2.4 that

$$\sum_{i=1}^{n} (f([i]) - f([i-1]))x_i \leq \hat{f}(\boldsymbol{x}) \leq \hat{f}(\bar{x})$$

which implies that $\boldsymbol{x}$ is not cut away by the hyperplane.

Next we show that $\eta = 0$. Our separating hyperplane induces a valid halfspace whenever it is not nonzero, i.e. $f([i]) \neq f([i-1])$ for some i. In the case that it is zero $f([i]) = f([i-1])\forall i$, by the same argument above, we have $\hat{f}(\bar{x}) = \sum_{i=1}^{n}(f([i]) - f([i-1]))\bar{x}_i = 0$ and

$$\hat{f}(\boldsymbol{x}) \geq \sum_{i=1}^{n}(f([i]) - f([i-1]))x_i = 0 = \hat{f}(\bar{x}).$$

In other words, $\bar{x}$ is an exact minimizer, i.e. $\eta = 0$.

Note that $\left|\hat{f}(\boldsymbol{x})\right| = \left|\mathbb{E}_{t\sim[0,1]}[f(\{i : x_i \geq t\})]\right| \leq M$ as $M = \max_S |f(S)|$. Now plugging in $\alpha = \frac{1}{4M}$ in the guarantee of Theorem 8.3.26, we can find a point x^* such that

$$\begin{aligned}\hat{f}(x^*) - \min_{\boldsymbol{x}\in[0,1]^n}\hat{f}(\boldsymbol{x}) &\leq \frac{1}{4M}\left(\max_{\boldsymbol{x}\in[0,1]^n}\hat{f}(\boldsymbol{x}) - \min_{\boldsymbol{x}\in[0,1]^n}\hat{f}(\boldsymbol{x})\right)\\ &\leq \frac{1}{4M}(2M)\\ &< 1\end{aligned}$$

We claim that $\min_{t\in[0,1]} f(\{i : x_i^* \geq t\})$ is minimum. To see this, recall from 10.2.2 that $\hat{f}$ has an integer minimizer and hence $\min_{\boldsymbol{x}\in[0,1]^n}\hat{f}(\boldsymbol{x}) = \min_S f(S)$. Moreover, $\hat{f}(x^*)$ is a convex combination of $f(\{i : x_i^* \geq t\})$ which gives

$$1 > \hat{f}(x^*) - \min_{\boldsymbol{x}\in[0,1]^n}\hat{f}(\boldsymbol{x}) = \hat{f}(x^*) - \min_S f(S) \geq \min_{t\in[0,1]} f(\{i : x_i^* \geq t\}) - \min_S f(S).$$

Since f is integer-valued, we must then have $\min_{t\in[0,1]} f(\{i : x_i^* \geq t\}) = \min_S f(S)$ as desired. Since our separation oracle can be computed by n oracle calls and runs in time $O(n \cdot \mathrm{EO} + n^2)$, by Theorem 9.3.2 the overall running time is then $O(n^2 \log nM \cdot \mathrm{EO} + n^3 \log^{O(1)} nM)$ as claimed. □

Needless to say the proof above completely depends on Theorem 9.3.2. We remark that one can use the Vaidya's cutting plane instead of ours to get a time complexity $O(n^2 \log nM \cdot \mathrm{EO} + n^{\omega+1} \log^{O(1)} n \cdot \log M)$. There is actually an alternate argument that gives a time complexity of $O(n^2 \log M \cdot \mathrm{EO} + n^{O(1)} \cdot \log M)$. Thus it requires slightly fewer oracle calls at the expense of slower running time. A proof is offered in this section, which can be skipped without any risk of discontinuation. This proof relies the following cutting plane method.

Theorem 10.3.2 ([36]). *Given any convex set $K \subset [0,1]^n$ with a separation oracle of cost SO, in time $O(kSO + kn^{O(1)})$ one can find either find a point $\boldsymbol{x} \in K$ or find a polytope P such that $K \subset P$ and the volume of K is at most $\left(\frac{2}{3}\right)^k$.*

The Theorem allows us to decrease the volume of the feasible region by a factor of $\left(\frac{2}{3}\right)^k$ after k iterations. Similar to above, we apply cutting plane to minimize $\hat{f}$ over the hypercube $[0,1]^n$ for $O(n \log M)$ iterations, and outputs *any* integral point in the remaining feasible region P.

Lemma 10.3.3. *Let x^* achieve the minimum function value $\hat{f}(x^*)$ among the points used to query the separation oracle. Then*

1. *$x^* \in P^{(k)}$, the current feasible region.*
2. *Any $\boldsymbol{x}$ with $\hat{f}(\boldsymbol{x}) \leq \hat{f}(x^*)$ belongs to $P^{(k)}$.*
3. *suppose $x_{i_1}^* \geq \cdots \geq x_{i_n}^*$ and let $S_j = \{i_1, \ldots, i_j\}$. Then $S_l \in \arg\min_{S_j} f(S_j)$ also belongs to $P^{(k)}$.*

Proof. For any separating hyperplane $\boldsymbol{h}^T x \leq \hat{f}(\bar{x})$ given by $\bar{x}$, we have by Lemma 10.2.6 that $\boldsymbol{h}^T x^* \leq \hat{f}(x^*)$. Since $\hat{f}(x^*)$ is the minimum among all $\hat{f}(\bar{x})$, $\boldsymbol{h}^T x^* \leq \hat{f}(\bar{x})$ and hence x^* is not removed by any new separating hyperplane. In other words, $x^* \in P^{(k)}$. The argument for (2) is analogous.

For (3), recall that by the definition of Lovász extension $\hat{f}(x^*)$ is a convex combination of $f(S_j)$ and thus the indicator variable I_{S_l} for S_l satisfies $f(I_{S_l}) \leq \hat{f}(x^*)$. By Lemma 10.2.6 again, this implies $\boldsymbol{h}^T I_{S_l} \leq f(I_{S_l}) \leq \hat{f}(x^*) \leq \hat{f}(\bar{x})$ for any separating hyperplane $\boldsymbol{h}^T x \leq \hat{f}(\bar{x})$. □

Theorem 10.3.4. *Suppose that we run Cutting Plane in Theorem 10.3.2 for $O(n \log M)$ iterations. Then S_l from the last lemma also minimizes f.*

Proof. We use the notations from the last lemma. After $k = Kn \log_{2/3} M$ iterations, the volume of the feasible region $P^{(k)}$ is at most $1/M^{Kn}$. By the last lemma, $I_{S_l} \in P^{(k)}$.

Suppose for the sake of contradiction that S minimizes f but $f(S) < f(S_l)$. Since f is integer-valued, $f(S) + 1 \leq f(S_l)$. Let $r \stackrel{\text{def}}{=} 1/6M$. Consider the set $B \stackrel{\text{def}}{=} \{\boldsymbol{x} : 0 \leq x_i \leq r \; \forall \text{i} \notin S, \, 1 - r \leq x_i \leq 1 \; \forall i \in S\}$. We claim that for $\boldsymbol{x} \in B$,

$$\hat{f}(\boldsymbol{x}) \leq f(S) + 1.$$

To show this, note that $f(\{i : x_i \geq t\}) = f(S)$ for $r < t \leq 1 - r$ as $x_i \leq r$ for $i \notin S$ and $x_i \geq 1 - r$ for $i \in S$. Now using conditional probability and $|f(T)| \leq M$ for any T,

$$\begin{aligned}
\hat{f}(\boldsymbol{x}) &= \mathbb{E}_{t\sim[0,1]}[f(\{i : x_i \geq t\})] \\
&= (1 - 2r)\, \mathbb{E}[f(\{i : x_i \geq t\}) | r < t \leq 1 - r] + \\
&\quad r\left(\mathbb{E}[f(\{i : x_i \geq t\}) | 0 \leq t \leq r] + \mathbb{E}[f(\{i : x_i \geq t\}) | 1 - r \leq t \leq 1]\right) \\
&= (1 - r)\, f(S) + r\left(\mathbb{E}[f(\{i : x_i \geq t\}) | 0 \leq t \leq r + \mathbb{E}[f(\{i : x_i \geq t\}) | 1 - r \leq t \leq 1]\right) \\
&\leq (1 - 2r)\, f(S) + 2rM \\
&\leq f(S) + 4rM \\
&\leq f(S) + 1
\end{aligned}$$

But now $B \subseteq P^{(k)}$ as $\hat{f}(\boldsymbol{x}) \leq f(S) + 1 \leq f(S_l)$ and by (2) of the last lemma. This would lead to a contradiction since

$$\text{vol}(B) = \frac{1}{(6M)^n} > \frac{1}{M^{Kn}} \geq \text{vol}(P^{(k)})$$

for sufficiently large K. □

Corollary 10.3.5. *There is an $O(n^2 \log M \cdot EO + n^{O(1)} \log M)$ time algorithm for submodular function minimization.*

Proof. This simply follows from the last lemma, Theorem 10.3.2, and the fact that our separation oracle runs in time $O(n \cdot \text{EO} + n^2)$. □

Curiously, we obtained $O(\log M)$ rather than $O(\log nM)$ as in our algorithm. We leave it as an open problem whether one can slightly improve our running time to $O(n^2 \log M \cdot \text{EO} + n^3 \log^{O(1)} n \cdot \log M)$. The rest of this chapter is devoted to obtaining better strongly polynomial running time.

■ 10.4 Improved Strongly Polynomial Algorithms for SFM

In this section we show how our cutting plane method can be used to obtain a $\widetilde{O}(n^3 \cdot \text{EO} + n^4)$ time algorithm for SFM, which improves over the currently fastest $O(n^5 \cdot \text{EO} + n^6)$ time algorithm by Orlin.

10.4.1 Improved Oracle Complexity

We first present a simple geometric argument that f can be minimized with just $O(n^3 \log n \cdot \mathrm{EO})$ oracle calls. While this is our desired query complexity (and it improves upon the previous best known bounds by a factor of $O(n^2)$ unfortunately the algorithm runs in exponential time. Nevertheless, it does provide some insight into how our more efficient algorithms should proceed and it alone, does suggests that information theoretically, $O(n^3 \log n \cdot \mathrm{EO})$ calls suffice to solve SFM. In the rest of this chapter, we combine this insight with some of the existing SFM tools developed over the last decade to get improved polynomial time algorithms.

Theorem 10.4.1. *Submodular functions can be minimized with $O(n^3 \log n \cdot EO)$ oracle calls.*

Proof. We use the cutting plane method in Theorem 10.3.2 with the separation oracle given by Theorem 10.2.4. This method reduce the volume of the feasible region by a factor of $(\frac{2}{3})^k$ after k iterations if the optimal has not found yet.

Now, we argue that after $O(n \log n)$ iterations of this procedure we have either found a minimizer of f or we have enough information to reduce the dimension of the problem by 1. To see this, first note that if the separation oracle ever returns a degenerate hyperplane, then by Lemma 10.2.8 then either $\emptyset$ or V is the minimizer, which we can determine in time $O(\mathrm{EO}+n)$. Otherwise, after $100n \log n$ iterations, our feasible region P must have a volume of at most $1/n^{10n}$. In this case, we claim that the remaining integer points in P all lie on a hyperplane. This holds, as if this was not the case, then there is a simplex $\triangle$, with integral vertices $v_0, v_1, \ldots, v_n$, contained in P. But then

$$\mathrm{vol}(P) \geq \mathrm{vol}(\triangle) = \frac{1}{n!} \left|\det\left(v_1 - v_0 \; v_2 - v_0 \; \ldots \; v_n - v_0\right)\right| \geq \frac{1}{n!}$$

where the last inequality holds since the determinant of an integral matrix is integral, yielding a contradiction.

In other words after $O(n \log n)$ iterations, we have reduced the dimension of all viable solutions by at least 1. Thus, we can recurse by applying the cutting plane method to the lower dimensional feasible region, i.e. P is (replaced by) the convex combination of all the remaining integer points. There is a minor technical issue we need to address as our space is now lower dimensional and the starting region is not necessarily the hypercube anymore and the starting volume is not necessarily equal to 1.

We argue that the starting volume is bounded by $n^{O(n)}$. If this is indeed the case, then our previous argument still works as the volume goes down by a factor of $1/n^{O(n)}$ in $O(n \log n)$ iterations.

Let $v \in P$ be an integer point. Now the $\dim(()P)$-dimensional ball of radius $\sqrt{n}$ centered at v must contain all the other integer points in P as any two points of $\{0,1\}^n$ are at most $\sqrt{n}$ apart. Thus the volume of P is bounded by the volume of the ball which is $n^{O(n)}$. Now to get the volume down to $1/n^{10n}$, the number of iterations is still $O(n \log n)$.

In summary, we have reduced our dimension by 1 using $O(n \log n)$ iterations which requires $O(n^2 \log n \cdot \mathrm{EO})$ oracle calls (as each separating hyperplane is computed with $n \cdot \mathrm{EO}$ oracle calls). This can happen at most n times. The overall query complexity is then $O(n^3 \log n \cdot \mathrm{EO})$.

Note that the minimizer $\boldsymbol{x}$ obtained may not be integral. This is not a problem as the definition of Lovász extension implies that if $\hat{f}(\boldsymbol{x})$ is minimal, then $f(\{i : x_i \geq t\})$ is minimal for any $t \in [0,1]$.

We remark that this algorithm does not have a polynomial runtime. Even though all the integral vertices of P lie on a hyperplane, the best way we know of that identifies it takes exponential time by checking for all the integer points $\{0,1\}^n$. □

Remark 10.4.2. Note that this algorithm works for minimizing any convex function over the hypercube that obtains its optimal value at a vertex of the hypercube. Formally, our proof of Theorem 10.4.1

holds whenever a function $f : 2^V \longrightarrow \mathbb{R}^n$ admits a convex relaxation $\hat{f}$ with the following properties:

1. For every $S \subseteq V$, $\hat{f}(I_S) = f(S)$.
2. Every $\hat{f}(\boldsymbol{x})$ can be written as a convex combination $\sum_{S\in\mathcal{S}} \alpha_S f(S)$, where $\sum \alpha_S = 1$, $|\mathcal{S}| = O(n)$, and $\mathcal{S}$ can be computed without any oracle call.
3. A subgradient $\partial\hat{f}(\boldsymbol{x})$ of $\hat{f}$ at any point $\boldsymbol{x} \in [0,1]^n$ can be computed with $O(n \cdot \mathrm{EO})$ oracle calls.

In this case, the proof of Theorem 10.4.1, implies that $\hat{f}$ and f can be minimized with $O(n^3 \log n \cdot \mathrm{EO})$ oracle calls by using the separating hyperplane $\partial\hat{f}(\bar{x})^T(\boldsymbol{x} - \bar{x}) \leq 0$.

10.4.2 Technical Tools

To improve upon the running time of the algorithm in the previous section, we use more structure of our submodular function f. Rather than merely showing that we can decrease the dimension of our SFM problem by 1 we show how we can reduce the degrees of freedom of our problem in a more principled way. In Section 10.4.2.1 we formally define the abstraction we use for this and discuss how to change our separation oracle to accommodate this abstraction, and in Section 10.4.2.2 we show how we can deduce these constraints. These tools serve as the foundation for the faster strongly polynomial time SFM algorithms we present in Section 10.4.3 and Section 10.4.4.

10.4.2.1 SFM over Ring Family

For the remainder of this chapter we consider a more general problem than SFM in which we wish to compute a minimizer of our submodular function f over a ring family of the ground set $V = [n]$. A ring family $\mathcal{F}$ is a collection of subsets of V such that for any $S_1, S_2 \in \mathcal{F}$, we have $S_1 \cup S_2, S_1 \cap S_2 \in \mathcal{F}$. Thus SFM corresponds to the special case where $\mathcal{F}$ consists of every subset of V. This generalization has been considered before in the literature and was essential to the IFF algorithm.

It is well known that any ring family $\mathcal{F}$ over V can be represented by a directed graph $D = (V, A)$ where $S \in \mathcal{F}$ iff S contains all of the descendants of any $i \in S$. An equivalent definition is that for any arc $(i, j) \in A$, $i \in S$ implies $j \in S$. It is customary to assume that A is acyclic as any (directed) cycle of A can be contracted (see section 10.4.3.1).

We denote by $R(i)$ the set of descendants of i (including i itself) and $Q(i)$ the set of ancestors of i (including i itself). Polyhedrally, an arc $(i, j) \in A$ can be encoded as the constraint $x_i \leq x_j$ as shown by the next lemma.

Lemma 10.4.3. *Let $\mathcal{F}$ be a ring family over V and $D = (V, A)$ be its directed acyclic graph representation. Suppose $f : V \longrightarrow \mathbb{R}$ is submodular with Lovász extension $\hat{f}$. Then the characteristic vector I_S of any minimizer $S = \arg\min_{S\in\mathcal{F}} f(S)$ over $\mathcal{F}$ is also the solution to*

$$\begin{aligned} \min & \hat{f}(\boldsymbol{x}) \\ & x_i \leq x_j \; \forall (i,j) \in A \\ & 0 \leq \boldsymbol{x} \leq 1 \end{aligned} \tag{10.1}$$

Proof. Let x^* be a minimizer, and $L^{(t)} = \{i : x^*_i \geq t\}$. It is easy to check that the indicator variable $I_{L^{(t)}}$ satisfies (10.1) since x^* does. Moreover, recall that $\hat{f}(x^*) = \mathbb{E}_{t\sim[0,1]}[f(L_t)]$. Thus $\hat{f}(x^*)$ can be written as a convex combination $\hat{f}(x^*) = \sum_t \alpha_t f(L^{(t)}) = \sum_t \alpha_t \hat{f}(I_{L^{(t)}})$, where $\alpha_t > 0$ and $\sum_t \alpha_t = 1$. Thus all such $\hat{f}(I_{L^{(t)}})$ are minimal, i.e. (10.1) has no "integrality gap". □

We also modify our separation oracle to accommodate for this generalization as follows. Before doing so we need a definition which relates our BFS to the ring family formalism.

Definition 10.4.4. A permutation $(v_1, \ldots, v_n)$ of V is said to be *consistent* with an arc (i, j) if j precedes i in $(v_1, \ldots, v_n)$. Similarly, a BFS of the base polyhedron is consistent with (i, j) if j precedes i in its defining permutation. $(v_1, \ldots, v_n)$ (or a BFS) is consistent with A if it is consistent with every $(i, j) \in A$.

Readers may find it helpful to keep in mind the following picture which depicts the relative positions between $R(i), i, Q(i)$ in the defining permutation of $\boldsymbol{h}$ that is consistent with A:

$$\cdots\cdots\ R(i)\backslash\{i\}\ \cdots\cdots\ i\ \cdots\cdots\ Q(i)\backslash\{i\}\ \cdots\cdots$$

In Theorem 10.2.4, given $\bar{x} \in [0, 1]^n$ our separating hyperplane is constructed by sorting the entries of $\bar{x}$. This hyperplane is associated with some BFS $\boldsymbol{h}$ of the base polyhedron. As we shall see towards the end of the section, we would like $\boldsymbol{h}$ to be consistent with every arc $(i, j) \in A$.

This task is easy initially as $\bar{x}$ satisfies $x_i \leq x_j$ for $(i, j) \in A$ for the starting polytope of (10.1). If $x_i < x_j$, nothing special has to be done as j must precede i in the ordering. On the other hand, whenever $x_i = x_j$, we can always break ties by ranking j ahead of i.

However, a technical issue arises due to the fact that our cutting plane algorithm may drop constraints from the current feasible region P. In other words, $\bar{x}$ may violate $x_i \geq 0$, $x_j \leq 1$ or $x_i \leq x_j$ if it is ever dropped. Fortunately this can be fixed by reintroducing the constraint. We summarize the modification needed in the pseudocode below and formally show that it fulfills our requirement.

Algorithm 25: Modified Separation Oracle

Input: $\bar{x} \in \mathbb{R}^n$ and the set of arcs A
if $\bar{x}_i < 0$ *for some* i **then**
  Output: $x_i \geq 0$
else if $\bar{x}_j > 1$ *for some* j **then**
  Output: $x_j \leq 1$
else if $\bar{x}_i > \bar{x}_j$ *for some* $(i, j) \in A$ **then**
  Output: $x_i \leq x_j$
else
  Let $i_1, \ldots, i_n$ be a permutation of V such that $\bar{x}_{i_1} \geq \ldots \geq \bar{x}_{i_n}$ and for all $(i, j) \in A$, j precedes i in $i_1, \ldots, i_n$.
  Output: $\boldsymbol{h}^T \boldsymbol{x} \leq \hat{f}(\bar{x})$, where $\boldsymbol{h}$ is the BFS defined by the permutation $i_1, \ldots, i_n$.

Lemma 10.4.5. *Our modified separation oracle returns either some BFS $\boldsymbol{h} = 0$ or a valid separating hyperplane, i.e.*

1. *$\bar{x}$ either lies on the separating hyperplane or is cut away by it.*
2. *Any minimizer of (10.1) is not cut away by the separating hyperplane.*

Such a hyperplane can be computed with n oracle calls to f and in time $O(n \cdot EO + n^2)$.

Proof. If we get $x_i \geq 0$, $x_j \leq 1$ or $x_i \leq x_j$ (if loop or the first two else loops), then clearly $\bar{x}$ is cut away by it and any minimizer must of course satisfy $x_i \geq 0$, $x_j \leq 1$ and $x_i \leq x_j$ as they are the constraints in (10.1). This proves (1) and (2) for the case of getting $x_i \geq 0$, $x_j \leq 1$ or $x_i \leq x_j$.

Thus it remains to consider the case $\boldsymbol{h}^T\boldsymbol{x} \leq \hat{f}(\bar{x})$ (last else loop). First of all, $\bar{x}$ lies on it as $\hat{f}(\bar{x}) = \boldsymbol{h}^T\bar{x}$. This proves (1). For (2), we have from Lemma 10.2.6 that $\boldsymbol{h}^T\boldsymbol{x} \leq \hat{f}(\boldsymbol{x})$. If x^* is a minimizer of (10.1), we must then have $\boldsymbol{h}^T x^* \leq \hat{f}(x^*) \leq \hat{f}(\bar{x})$ as $\bar{x}$ is also feasible for (10.1).

Finally we note that the running time is self-evident. □

We stress again that the main purpose of modifying our separation oracle is to ensure that any BFS $\boldsymbol{h}$ used to define a new separating hyperplane must be consistent with every $(i,j) \in A$.

10.4.2.2 Identifying New Valid Arcs

The reason for considering the ring family generalization of SFM is that our algorithms (and some previous algorithms too) work by adding new arcs to our digraph D. This operation yields a strongly polynomial algorithm since there are only $2 \cdot \binom{n}{2}$ possible arcs to add. Of course, a new arc (i,j) is valid only if $i \in S_{\min} \implies j \in S_{\min}$ for some minimizer $S_{\min}$. Here we show how to identify such valid arcs by extracting information from certain nice elements of the base polyhedron.

This is guaranteed by the next four lemmas, which are stated in a way different from previous works e.g. our version is extended to the ring family setting. This is necessary as our algorithms require a more general formulation. We also give a new polyhedral proof, which is mildly simpler than the previous combinatorial proof. On the other hand, Lemma 10.4.10 is new and unique to our work. It is an important ingredient of our $\widetilde{O}(n^3 \cdot \mathrm{EO} + n^4)$ time algorithm.

Recall that each BFS of the base polyhedron is defined by some permutation of the ground set elements.

First, we prove the following two lemmas which show that should we ever encounter a non-degenerate point in the base polytope with a coordinate of very large value, then we can immediately conclude that that coordinate must be or must not be in solution to SFM over the ring family.

Lemma 10.4.6. *If $\boldsymbol{y} \in \mathcal{B}(f)$ is non-degenerate and satisfies $y_i > -(n-1)\min_j y_j$, then i is not in any minimizer of f (over the ring family A).*

Proof. We proceed by contradiction and suppose that S is a minimizer of f that contains i. Now since $\boldsymbol{y}$ is non-degenerate we know that $\min_j y_j \leq 0$ and by the definition of $\boldsymbol{y}$ we have the following contradiction

$$0 < y_i + (n-1)\min_j y_j \leq \sum_{j \in S} y_j = \boldsymbol{y}(S) \leq f(S) \leq f(\emptyset) = 0\,.$$

□

Lemma 10.4.7. *If $\boldsymbol{y} \in \mathcal{B}(f)$ is non-degenerate and satisfies $y_i < -(n-1)\max_j y_j$, then i is in every minimizer of f (over the ring family A).*

Proof. We proceed by contradiction and suppose that S is a minimizer of f that does not contain i. Now since $\boldsymbol{y}$ is non-degenerate we know that $\max_j y_j \geq 0$ and therefore

$$\sum_{j \in [n]} y_j = y_i + \sum_{j \in S} y_j + \sum_{j \in V-(S+i)} y_j < -(n-1)\max_j y_j + \sum_{j \in S} y_j + (|V| - |S| - 1)\max_j y_j \leq \sum_{j \in S} y_j\,.$$

However by the definition of $\boldsymbol{y}$ we have

$$\sum_{j \in S} y_j = \boldsymbol{y}(S) \leq f(S) \leq f(V) = \sum_{j \in [n]} y_j\,.$$

Thus we have a contradiction and the result follows. □

Now we are ready to present conditions under which a new valid arc can be added. We begin with a simple observation. Let $\texttt{upper}(i) \stackrel{\text{def}}{=} f(R(i)) - f(R(i)-i)$ and $\texttt{lower}(i) \stackrel{\text{def}}{=} f(V\backslash Q(i)+i) - f(V\backslash Q(i))$. As the names suggest, they bound the value of h_i for any BFS used.

Lemma 10.4.8. *For any BFS $\boldsymbol{h}$ used to construct a separating hyperplane given by our modified separation oracle, we have $\texttt{lower}(i) \le h_i \le \texttt{upper}(i)$.*

Proof. Note that by Lemma 10.4.5, $\boldsymbol{h}$ is consistent with every $(j_1, j_2) \in A$ and hence i must precede $Q(i)$ and be preceded by $R(i)$. Let S be the set of elements preceding i in the defining permutation of $\boldsymbol{h}$. Then $h_i = f(S+i) - f(S) \le f(R(i)) - f(R(i)-i)$ because of diminishing return and $R(i) - i \subseteq S$. The lower bound follows from the same argument as $Q(i) - i$ comes after i, and so $Q(i) \subseteq V\backslash S$. □

In the following two lemmas, we show that if $\texttt{upper}(i)$ is ever sufficiently positive or $\texttt{lower}(i)$ is sufficiently negative, then we find a new arc.

While these lemmas may appear somewhat technical but actually has an intuitive interpretation. Suppose an element p is in a minimizer $S_{\min}$ of f over the ring family D. Then $R(p)$ must also be part of $S_{\min}$. Now if $f(R(p))$ is very large relative to $f(R(p) - p)$, there should be some element $q \in S_{\min}\backslash R(p)$ compensating for the discrepancy. The lemma says that such an element q can in fact be found efficiently.

Lemma 10.4.9 (new arc)**.** *Let $\boldsymbol{y} = \sum_k \lambda^{(k)} \boldsymbol{y}^{(k)}$ be a non-degenerate convex combination of $O(n)$ base polyhedron BFS's $\boldsymbol{y}^{(k)}$ which are consistent with every arc $(i,j) \in A$. If some element p satisfies $\texttt{upper}(p) > n^4 \max y_j$, then we can find, using $O(n \cdot EO)$ oracle calls and $O(n^2)$ time, some $q \notin R(p)$ such that the arc (p,q) is valid, i.e. if p is in a minimizer, then so is q.*

Proof. If $\max y_j < 0$ then we are immediately done by Lemma 10.2.8. We assume $\max y_j \ge 0$ in the proof. For all k let $\boldsymbol{y}'^{(k)}$ be the BFS obtained by taking the defining permutation of $\boldsymbol{y}^{(k)}$ and moving $R(p)$ to the front while preserving the relative ordering of $R(p)$ within each permutation). Furthermore, let $\boldsymbol{y}' \stackrel{\text{def}}{=} \sum_k \lambda^{(k)} \boldsymbol{y}'^{(k)}$. Then since $y'^{(k)}_p = f(R(p)) - f(R(p) - p) = \texttt{upper}(p)$ we have $\texttt{upper}(p) = y'_p = f(R(p)) - f(R(p) - p)$. Moreover,

$$y'_j \ge y_j \quad \forall j \in R(p) \text{ and } y'_j \le y_j \quad \forall j \notin R(p) \tag{10.2}$$

by diminishing marginal return.

Now, suppose p is in a minimizer $S_{\min}$. Then $R(p) \subseteq S_{\min}$ by definition. We then define $f'(S) = f(S \cup R(p))$ for $S \subseteq V\backslash R(p)$. It can be checked readily that f' is submodular and $S_{\min}\backslash R(p)$ is a minimizer of f' (over the corresponding ring family). Note that now $\boldsymbol{y}'_{V\backslash R(p)}$ (the restriction of $\boldsymbol{y}'$ to $V\backslash R(p)$) is a convex combination of the BFS's of the base polyhedron $\mathcal{B}(f')$ of f'. We shall show that $\boldsymbol{y}'_{V\backslash R(p)}$ has the desired property in Lemma 10.4.7.

Note that $y'(V\backslash R(p) + p) \le y(V\backslash R(p) + p)$ since

$$y'(V\backslash R(p)+p) = y'(V) - y'(R(p)-p) = y(V) - y'(R(p)-p) \le y(V) - y(R(p)-p) = y(V\backslash R(p)+p).$$

But now since $\boldsymbol{y}$ is non-degenerate $\max_j y_j \ge 0$ and therefore

$$\begin{aligned} y'(V\backslash R(p)) &\le y(V\backslash R(p)+p) - y'_p \\ &= y(V\backslash R(p)+p) - (f(R(p)) - f(R(p)-p)) \\ &\le n \max y_j - (f(R(p)) - f(R(p)-p)) \\ &< (n - n^4) \max y_j \end{aligned} \tag{10.3}$$

Therefore by the Pigeonhole Principle some $q \notin R(p)$ must satisfy

$$\begin{aligned} y'_q &< \left((n-n^4)\max y_j\right)/(n-1) \\ &= -(n^3+n^2+n)\max y_j \\ &\le -(n^3+n^2+n)\max_{j\notin R(p)} y_j \\ &\le -(n^3+n^2+n)\max_{j\notin R(p)} y'_j \quad \text{by (10.2)} \end{aligned}$$

By Lemma 10.4.7, this q must be in any minimizer of f'. In other words, whenever p is in a minimizer of f, then so is q.

Note however that computing all $\boldsymbol{y}'$ would take $O(n^2)$ oracle calls in the worst case as there are $O(n)$ $\boldsymbol{y}'^{(k)}$'s. We use the following trick to identify some q with $y'_q < -(n-1)\max y_j$ using just $O(n)$ calls. The idea is that we actually only want to have sufficient decreases in $y'(V\backslash R(p))$ which can be accomplished by having a large corresponding decrease in some $\boldsymbol{y}'^{(k)}$.

For each k, by the same argument above (see (10.3))

$$y'^{(k)}(V\backslash R(p)) - y^{(k)}(V\backslash R(p)) \le y_p^{(k)} - (f(R(p)) - f(R(p)-p)) \tag{10.4}$$

The "weighted decrease" $\lambda^{(k)}\left(y_p^{(k)} - (f(R(p)) - f(R(p)-p))\right)$ for $\boldsymbol{y}'^{(k)}$ sum up to

$$\sum \lambda^{(k)}\left(y_p^{(k)} - (f(R(p)) - f(R(p)-p))\right) = y_p - (f(R(p)) - f(R(p)-p)) < (1-n^4)\max y_j$$

Thus by the Pigeonhole Principle, some l will have

$$\lambda^{(l)}\left(y_p^{(l)} - (f(R(p)) - f(R(p)-p))\right) < \left((1-n^4)\max y_j\right)/O(n) < -n^2 \max y_j.$$

For this $\boldsymbol{y}^{(l)}$ we compute $\boldsymbol{y}'^{(l)}$. We show that $\boldsymbol{y}'' = \lambda^{(l)}\boldsymbol{y}'^{(l)} + \sum_{k\neq l}\lambda^{(k)}\boldsymbol{y}^{(k)}$ has the same property as $\boldsymbol{y}'$ above.

$$\begin{aligned} y''(V\backslash R(p)) &= \lambda^{(l)} y'^{(l)}(V\backslash R(p)) + \sum_{k\neq l}\lambda^{(k)} y^{(k)}(V\backslash R(p)) \\ &= y(V\backslash R(p)) + \lambda^{(l)}\left(y'^{(l)}(V\backslash R(p)) - y^{(l)}(V\backslash R(p))\right) \\ &\le y(V\backslash R(p)) + \lambda^{(l)}\left(y_p^{(l)} - (f(R(p)) - f(R(p)-p))\right) \quad \text{by (10.4)} \\ &< (n-1)\max y_j - n^2 \max y_j \\ &< (n-n^2)\max y_j \end{aligned}$$

Then some $q \in V\backslash R(p)$ must satisfy

$$y''_q < \frac{n-n^2}{n-1}\max y_j = -n\max y_j$$

That is, the arc (p,q) is valid. This takes $O(n)$ oracle calls as given $\boldsymbol{y} = \sum_k \lambda^{(k)}\boldsymbol{y}^{(k)}$, computing $\boldsymbol{y}''$ requires knowing only $f(R(p))$, $f(R(p)-p)$, and $\boldsymbol{y}'^{(l)}$ which can be computed from $\boldsymbol{y}^{(l)}$ with n oracle calls. The runtime is $O(n^2)$ which is needed for computing $\boldsymbol{y}''$. □

Lemma 10.4.10. *Let $\boldsymbol{y} = \sum_k \lambda^{(k)}\boldsymbol{y}^{(k)}$ be a non-degenerate convex combination of base polyhedron BFS $\boldsymbol{y}^{(k)}$ which is consistent with every arc $(i,j) \in A$. If $\texttt{lower}(p) < n^4 \min y_j$, then we can find, using $O(n \cdot EO)$ oracle calls and $O(n^2)$ time, some $q \notin Q(p)$ such that the arc (q,p) is valid, i.e. if p is not in a minimizer, then q is not either.*

Proof. It is possible to follow the same recipe in the proof of Lemma 10.4.9 but using Lemma 10.4.6 instead of Lemma 10.4.7. Here we offer a proof which directly invokes Lemma 10.4.7 on a different submodular function.

Let g be defined by $g(S) \stackrel{\text{def}}{=} f(V \backslash S)$ for any S, and A_g be the set of arcs obtained by reversing the directions of the arcs of A. Consider the problem of minimizing g over the ring family A_g. Using subscripts to avoid confusion with f and g, e.g. $R_g(i)$ is the set of descendants of i w.r.t. A_g, it is not hard to verify the following:

- g is submodular
- $R_g(i) = Q_f(i)$
- $g(R_g(p)) - g(R_g(p) - p) = -\left(f(V \backslash Q_f(p) + p) - f(V \backslash Q_f(p))\right)$
- $-\boldsymbol{y}^{(k)}$ is a BFS of $\mathcal{B}(g)$ if and only if $\boldsymbol{y}^{(k)}$ is a BFS of $\mathcal{B}(f)$
- $\max(-y_j) = -\min y_j$

By using the above correspondence and applying Lemma 10.4.9 to g and A_g, we can find, using $O(n)$ oracle calls and $O(n^2)$ time, some $q \notin R_g(p) = Q(p)$ such that the arc (p, q) is valid for g and A_g. In other words, the reverse (q, p) will be valid for f and A. □

These lemmas lay the foundation of our algorithm. They suggests that if the positive entries of a point in the base polyhedron are small relative to some $\texttt{upper}(p) = f(R(p)) - f(R(p) - p)$, a new arc (p, q) can be added to A. This can be seen as a robust version of Lemma 10.2.8.

Finally, we end the section with a technical lemma that will be used crucially for both of our algorithms. The importance of it would become obvious when it is invoked in our analyses.

Lemma 10.4.11. *Let $\boldsymbol{h}''$ denote a convex combination of two vectors $\boldsymbol{h}$ and $\boldsymbol{h}'$ in the base polyhedron, i.e. $\boldsymbol{h}'' = \lambda \boldsymbol{h} + (1 - \lambda)\boldsymbol{h}'$ for some $\lambda \in [0, 1]$. Further suppose that*

$$\|\boldsymbol{h}''\|_2 \leq \alpha \min\left\{\lambda \|\boldsymbol{h}\|_2, (1-\lambda)\|\boldsymbol{h}'\|_2\right\}$$

for some $\alpha \leq \frac{1}{2\sqrt{n}}$. Then for $p = \arg\max_j(\max\{\lambda|h_j|, (1-\lambda)|h'_j|\})$ we have

$$\texttt{lower}(p) \leq -\frac{1}{2\alpha\sqrt{n}} \cdot \|\boldsymbol{h}''\|_\infty \quad \text{and} \quad \texttt{upper}(p) \geq \frac{1}{2\alpha\sqrt{n}} \cdot \|\boldsymbol{h}''\|_\infty \quad .$$

Proof. Suppose without loss of generality that $\lambda|h_p| \geq (1-\lambda)|h'_p|$. Then by assumptions we have

$$\|\boldsymbol{h}''\|_\infty \leq \|\boldsymbol{h}''\|_2 \leq \alpha \cdot \min\left\{\lambda\|\boldsymbol{h}\|_2, (1-\lambda)\|\boldsymbol{h}'\|_2\right\} \leq \alpha\sqrt{n}\,|\lambda h_p| \quad .$$

However, since $\alpha \leq \frac{1}{2\sqrt{n}}$ we see that

$$\left|\lambda h_p + (1-\lambda)h'_p\right| \leq \|h''\|_\infty \leq \alpha\sqrt{n}\,|\lambda h_p| \leq \frac{1}{2}\,|\lambda h_p| \quad .$$

Consequently, λh_p and $(1-\lambda)h'_p$ have opposite signs and $\left|(1-\lambda)h'_p\right| \geq \frac{1}{2}\left|\lambda h'_p\right|$. We then have,

$$\texttt{lower}(p) \leq \min\left\{h_p, h'_p\right\} \leq \min\left\{\lambda h_p, (1-\lambda)h'_p\right\} \leq -\frac{1}{2}\,|\lambda h_p| \leq -\frac{1}{2\alpha\sqrt{n}}\|h''\|_\infty$$

and

$$\texttt{upper}(p) \geq \max\left\{h_p, h'_p\right\} \geq \max\left\{\lambda h_p, (1-\lambda)h'_p\right\} \geq \frac{1}{2}\,|\lambda h_p| \geq \frac{1}{2\alpha\sqrt{n}}\|h''\|_\infty\,.$$

□

10.4.3 $\widetilde{O}(n^4 \cdot \mathbf{EO} + n^5)$ Time Algorithm

Here we present a $\widetilde{O}(n^4 \cdot \mathrm{EO} + n^5)$ time, i.e. strongly polynomial time algorithm, for SFM. We build upon the algorithm achieved in the section to achieve a faster running time in Section 10.4.4.

Our new algorithm combines the existing tools for SFM developed over the last decade with our cutting plane method. While there are certain similarities with previous algorithms (especially [118, 125, 121]), our approach significantly departs from all the old approaches in one important aspect.

All of the previous algorithms actively maintain a point in the base polyhedron and represent it as a *convex combination* of BFS's. At each step, a new BFS may enter the convex combination and an old BFS may exit. Our algorithm, on the other hand, maintains only a *collection* of BFS's (corresponding to our separating hyperplanes), rather than an explicit convex combination. A "good" convex combination is computed from the *collection* of BFS's only after running Cutting Plane for enough iterations. We believe that this crucial difference is the fundamental reason which offers the speedup. This is achieved by the Cutting Plane method which considers the *geometry* of the collection of BFS's. On the other hand, considering only a convex combination of BFS's effectively narrows our sight to only one *point* in the base polyhedron.

Overview

Now we are ready to describe our strongly polynomial time algorithm. Similar to the weakly polynomial algorithm, we first run our cutting plane for enough iterations on the initial feasible region $\{\boldsymbol{x} \in [0,1]^n : x_i \leq x_j \, \forall (i,j) \in A\}$, after which a pair of approximately parallel supporting hyperplanes F_1, F_2 of width $1/n^{\Theta(1)}$ can be found. Our strategy is to write F_1 and F_2 as a nonnegative combination of the facets of remaining feasible region P. This combination is made up of newly added separating hyperplanes as well as the inequalities $x_i \geq 0$, $x_j \leq 1$ and $x_i \leq x_j$. We then argue that one of the following updates can be done:

- Collapsing: $x_i = 0$, $x_j = 1$ or $x_i = x_j$
- Adding a new arc (i,j): $x_i \leq x_j$ for some $(i,j) \notin A$

The former case is easy to handle by elimination or contraction. If $x_i = 0$, we simply eliminate i from the ground set V; and if $x_i = 1$, we redefine f so that $f(S) = f(S+i)$ for any $S \subseteq V - i$. $x_i = x_j$ can be handled in a similar fashion. In the latter case, we simply add the arc (i,j) to A. We then repeat the same procedure on the new problem.

Roughly speaking, our strongly polynomial time guarantee follows as eliminations and contractions can happen at most n times and at most $2 \cdot \binom{n}{2}$ new arcs can be added. While the whole picture is simple, numerous technical details come into play in the execution. We advise readers to keep this overview in mind when reading the subsequent sections.

Algorithm

Our algorithm is summarized below. Again, we remark that our algorithm simply uses Theorem 10.4.12 regarding our cutting plane and is agnostic as to how the cutting plane works, thus it could be replaced with other methods, albeit at the expense of slower runtime.

1. Run cutting plane on (10.1) (Theorem 10.4.12 with $\tau = \Theta(1)$) using our modified separation oracle (Section 10.4.2.1).
2. Identify a pair of "narrow" approximately parallel supporting hyperplanes or get some BFS $\boldsymbol{h} = 0$ (in which case both $\emptyset$ and V are minimizers).

3. Deduce from the hyperplanes some new constraint of the forms $x_i = 0, x_j = 1, x_i = x_j$ or $x_i \leq x_j$ (Section 10.4.3.2).

4. Consolidate A and f (Section 10.4.3.1).

5. Repeat by running our cutting plane method on (10.1) with updated A and f. (Note that Any previously found separating hyperplanes are discarded.)

We call step (1) a *phase* of cutting plane. The minimizer can be constructed by unraveling the recursion.

10.4.3.1 Consolidating A and f

Here we detail how the set of valid arcs A and submodular function f should be updated once we deduce new information $x_i = 0, x_i = 1, x_i = x_j$ or $x_i \leq x_j$. Recall that $R(i)$ and $Q(i)$ are the sets of descendants and ancestors of i respectively (including i itself). The changes below are somewhat self-evident, and are actually used in some of the previous algorithms so we only sketch how they are done without a detailed justification.

Changes to the digraph representation D of our ring family include:

- $x_i = 0$: remove $Q(i)$ from the ground set and all the arcs incident to $Q(i)$
- $x_i = 1$: remove $R(i)$ from the ground set and all the arcs incident to $R(i)$
- $x_i = x_j$: contract i and j in D and remove any duplicate arcs
- $x_i \leq x_j$: insert the arc (i, j) to A
- For the last two cases, we also contract the vertices on a directed cycle of A until there is no more. Remove any duplicate arcs.

Here we can contract any cycle $(i_1, \ldots, i_k)$ because the inequalities $x_{i_1} \leq x_{i_2}, \ldots, x_{i_{k-1}} \leq x_{i_k}, x_{i_k} \leq x_{i_1}$ imply $x_{i_1} = \ldots = x_{i_k}$.

Changes to f:

- $x_i = 0$: replace f by $f' : 2^{V \setminus Q(i)} \longrightarrow \mathbb{R}$, $f'(S) = f(S)$ for $S \subseteq V \setminus Q(i)$
- $x_i = 1$: replace f by $f' : 2^{V \setminus R(i)} \longrightarrow \mathbb{R}$, $f'(S) = f(S \cup R(i))$ for $S \subseteq V \setminus R(i)$
- $x_i = x_j$: see below
- $x_i \leq x_j$: no changes to f needed if it does not create a cycle in A; otherwise see below
- Contraction of $C = \{i_1, \ldots, i_k\}$: replace f by $f' : 2^{V \setminus C + l} \longrightarrow \mathbb{R}$, $f'(S) = f(S)$ for $S \subseteq V \setminus C$ and $f'(S) = f((S - l) \cup C)$ for $S \ni l$

Strictly speaking, these changes are in fact *not* needed as they will automatically be taken care of by our cutting plane method. Nevertheless, performing them lends a more natural formulation of the algorithm and simplifies its description.

10.4.3.2 Deducing New Constraints $x_i = 0$, $x_j = 1$, $x_i = x_j$ or $x_i \leq x_j$

Here we show how to deduce new constraints through the result of our cutting plane method. This is the most important ingredient of our algorithm. As mentioned before, similar arguments were used first by IFF [121] and later in [118, 125]. There are however two important differences for our method:

- We maintain a collection of BFS's rather a convex combination; a convex combination is computed and needed only after each phase of cutting plane.
- As a result, our results are proved mostly *geometrically* whereas the previous ones were proved mostly *combinatorially*.

Our ability to deduce such information hinges on the power of the cutting plane method in Chapter 8. We re-state our main result Theorem 8.3.26 in the language of SFM. Note that Theorem 10.4.12 is formulated in a fairly general manner in order to accommodate for the next section. Readers may wish to think $\tau = \Theta(1)$ for now.

Theorem 10.4.12 (Theorem 8.3.26 restated for SFM)**.** *For any $\tau \geq 100$, applying our cutting plane method, Theorem 10.4.12, to (10.1) with our modified separation oracle (or its variant in Section 10.4.4) with high probability in n either*

1. *Finds a degenerate BFS $\boldsymbol{h} \geq \vec{0}$ or $\boldsymbol{h} \leq \mathbf{0}$.*
2. *Finds a polytope P consisting of $O(n)$ constraints which are our separating hyperplanes or the constraints in (10.1). Moreover, P satisfies the following inequalities*

$$\boldsymbol{c}^T\boldsymbol{x} \leq M \quad \text{and} \quad \boldsymbol{c}'^T\boldsymbol{x} \leq M',$$

both of which are nonnegative combinations of the constraints of P, where

$$||\boldsymbol{c}+\boldsymbol{c}'||_2 \leq \min\{||\boldsymbol{c}||_2, ||\boldsymbol{c}'||_2\}/n^{\Theta(\tau)} \quad \text{and} \quad |M+M'| \leq \min\{||\boldsymbol{c}||_2, ||\boldsymbol{c}'||_2\}/n^{\Theta(\tau)}.$$

Furthermore, the algorithm runs in expected time $O(n^2\tau \log n \cdot EO + n^3\tau^{O(1)} \log^{O(1)} n)$.

Proof. In applying Theorem 10.4.12 we let K be the set of minimizers of f over the ring family and the box is the hypercube with $R = 1$. We run cutting plane with our modified separation oracle (Lemma 10.4.5). The initial polytope $P^{(0)}$ can be chosen to be, say, the hypercube. If some separating hyperplane is degenerate, then we have the desired result (and know that either $\emptyset$ or V is optimal). Otherwise let P be the current feasible region. Note that $P \neq \emptyset$, because our minimizers of $\hat{f}$ are all in $P^{(0)}$ and $P^{(k)}$ as they are never cut away by the separating hyperplanes.

Let $\mathcal{S}$ be the collection of inequalities (10.1) as well as the separating hyperplanes $\boldsymbol{h}^T\boldsymbol{x} \leq \hat{f}(\bar{x}_h) = \boldsymbol{h}^T\bar{x}_h$ used. By Theorem 8.3.26, all of our minimizers will be contained in P, consisting of $O(n)$ constraints $\boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b}$. Each such constraint $\boldsymbol{a}_i^T\boldsymbol{x} \geq b_i$ is a scaling and shifting of some inequality $\boldsymbol{p}_i^T\boldsymbol{x} \geq q_i$ in $\mathcal{S}$, i.e. $\boldsymbol{a}_i = \boldsymbol{p}_i/||\boldsymbol{p}_i||_2$ and $b_i \leq q_i/||\boldsymbol{p}_i||_2$.

By taking $\varepsilon = 1/n^{\Theta(\tau)}$ with sufficiently large constant in Θ, our theorem certifies that P has a narrow width by $\boldsymbol{a}_1$, some nonnegative combination $\sum_{i=2}^{O(n)} t_i\boldsymbol{a}_i$ and point $\boldsymbol{x}_o \in P$ with $||\boldsymbol{x}_o||_\infty \leq 3\sqrt{n}R = 3\sqrt{n}$ satisying the following:

$$\left\|\boldsymbol{a}_1 + \sum_{i=2}^{O(n)} t_i\boldsymbol{a}_i\right\|_2 \leq 1/n^{\Theta(\tau)}$$

$$0 \leq \boldsymbol{a}_1^T\boldsymbol{x}_o - \boldsymbol{b}_1 \leq 1/n^{\Theta(\tau)}$$

$$0 \le \left(\sum_{i=2}^{O(n)} t_i a_i\right)^T \boldsymbol{x}_o - \sum_{i=2}^{O(n)} t_i b_i \le 1/n^{\Theta(\tau)}$$

We convert these inequalities to $\boldsymbol{p}$ and q. Let $t'_i \stackrel{\text{def}}{=} t_i \cdot ||\boldsymbol{p}_1||_2/||\boldsymbol{p}_i||_2 \ge 0$.

$$\left\|\boldsymbol{p}_1 + \sum_{i=2}^{O(n)} t'_i \boldsymbol{p}_i\right\|_2 \le ||\boldsymbol{p}_1||_2/n^{\Theta(\tau)}$$

$$0 \le \boldsymbol{p}_1^T \boldsymbol{x}_o - q_1 \le ||\boldsymbol{p}_1||_2/n^{\Theta(\tau)}$$

$$0 \le \left(\sum_{i=2}^{O(n)} t'_i \boldsymbol{p}_i\right)^T \boldsymbol{x}_o - \sum_{i=2}^{O(n)} t'_i q_i \le ||\boldsymbol{p}_1||_2/n^{\Theta(\tau)}$$

We claim that[1] $\boldsymbol{c} = -\boldsymbol{p}_1$, $M = -q_1$, $\boldsymbol{c}' = -\sum_{i=2}^{O(n)} t'_i \boldsymbol{p}_i$, $M' = -\sum_{i=2}^{O(n)} t'_i q_i$ satisfy our requirement.

We first show that $||\boldsymbol{c}+\boldsymbol{c}'||_2 \le \min\{||\boldsymbol{c}||_2, ||\boldsymbol{c}'||_2\}/n^{\Theta(\tau)}$. We have $||\boldsymbol{c}+\boldsymbol{c}'||_2 \le ||\boldsymbol{c}||_2/n^{\Theta(\tau)}$ from the first inequality. If $||\boldsymbol{c}||_2 \le ||\boldsymbol{c}'||_2$ we are done. Otherwise, by triangle inequality

$$||\boldsymbol{c}'||_2 - ||\boldsymbol{c}||_2 \le ||\boldsymbol{c}+\boldsymbol{c}'||_2 \le ||\boldsymbol{c}||_2/n^{\Theta(\tau)} \implies 2||\boldsymbol{c}||_2 \ge ||\boldsymbol{c}'||_2$$

and hence $||\boldsymbol{c}+\boldsymbol{c}'||_2 \le ||\boldsymbol{c}||_2/n^{\Theta(\tau)} \le ||\boldsymbol{c}'||_2/2n^{\Theta(\tau)} = ||\boldsymbol{c}'||_2/n^{\Theta(\tau)}$.

We also need to prove $|M + M'| \le \min\{||\boldsymbol{c}||_2, ||\boldsymbol{c}'||_2\}/n^{\Theta(\tau)}$. Summing the second and third inequalities,

$$-||\boldsymbol{c}||_2/n^{\Theta(\tau)} \le (\boldsymbol{c}+\boldsymbol{c}')^T \boldsymbol{x}_o - (M + M') \le 0$$

Recall that we have $||\boldsymbol{x}_o||_\infty \le 3\sqrt{n}$. Then

$$\begin{aligned} |M + M'| &\le |(\boldsymbol{c}+\boldsymbol{c}')^T \boldsymbol{x}_o - (M + M')| + |(\boldsymbol{c}+\boldsymbol{c}')^T \boldsymbol{x}_o| \\ &\le ||\boldsymbol{c}||_2/n^{\Theta(\tau)} + 3\sqrt{n}||\boldsymbol{c}+\boldsymbol{c}'||_2 \\ &\le ||\boldsymbol{c}||_2/n^{\Theta(\tau)} + 3\sqrt{n}||\boldsymbol{c}||_2/n^{\Theta(\tau)} \\ &= ||\boldsymbol{c}||_2/n^{\Theta(\tau)} \end{aligned}$$

as desired. Our result then follows as we proved $2||\boldsymbol{c}'||_2 \ge ||\boldsymbol{c}||_2$.

Finally, we have the desired runtime as our modified separation oracle runs in time $O(n \cdot \text{EO} + n^2 \log^{O(1)} n)$. □

Informally, the theorem above simply states that after $O(n\tau \log n)$ iterations of cutting plane, the remaining feasible region P can be sandwiched between two approximately parallel supporting hyperplanes of width $1/n^{O(\tau)}$. A good intuition to keep in mind is that every $O(n)$ iterations of cutting plane reduces the minimum width by a constant factor.

Remark 10.4.13. As shown in the proof of Theorem 10.4.12, one of the two approximately parallel hyperplanes can actually be chosen to be a constraint of our feasible region P. However we do not exploit this property as it does not seem to help us and would break the notational symmetry in $\boldsymbol{c}$ and $\boldsymbol{c}'$.

Setup

[1] Minus signs is needed because we express our inequalities as e.g. $\boldsymbol{h}^T\boldsymbol{x} \le \boldsymbol{h}^T\bar{x}_h$ whereas in Theorem 8.3.26, $\boldsymbol{a}_i^T\boldsymbol{x} \ge b_i$ is used. We apologize for the inconvenience.

In each phase, we run cutting plane using Theorem 10.4.12 with $\tau = \Theta(1)$. If some separating hyperplane used is degenerate, we have found the minimizer by Lemma 10.2.8.

Now assume none of the separating hyperplanes is degenerate. By Theorem 10.4.12, P is sandwiched by a pair of approximately parallel supporting hyperplanes F, F' which are of width $1/10n^{10}$ apart. The width here can actually be $1/n^c$ for any constant c by taking a sufficiently large constant in Theta.

Here, we show how to deduce from F and F' some $x_i = 0,, x_j = 1, x_i = x_j$, or $x_i \leq x_j$ constraint on the minimizers of f over the ring family. Let

$$\boldsymbol{c}^T\boldsymbol{x} = \sum c_i x_i \leq M \quad \text{and} \quad \boldsymbol{c}'^T\boldsymbol{x} = \sum c'_i x_i \leq M'$$

be the inequality for F and F' such that

$$|M + M'|,\ ||\boldsymbol{c} + \boldsymbol{c}'||_2 \leq \mathtt{gap}, \quad \text{where } \mathtt{gap} \stackrel{\text{def}}{=} \frac{1}{10n^{10}} \min\{||\boldsymbol{c}||_2, ||\boldsymbol{c}'||_2\}.$$

By the same theorem we can write $\boldsymbol{c}^T\boldsymbol{x} \leq M$ as a nonnegative combination of the constraints for P. Recall that the constraints for P take on four different forms: (1) $-x_i \leq 0$; (2) $x_j \leq 1$; (3) $-(x_j - x_i) \leq 0$; (4) $\boldsymbol{h}^T\boldsymbol{x} = \sum h_i x_i \leq \hat{f}(\bar{x}_h)$. Here the first three types are present initially whereas the last type is the separating hyperplane added. As alleged previously, the coefficient vector $\boldsymbol{h}$ corresponds to a BFS of the base polyhedron for f. Our analysis crucially exploits this property.

Thus suppose $\boldsymbol{c}^T\boldsymbol{x} = \sum_i c_i x_i \leq M$ is a nonnegative combination of our constraints with weights $\alpha_i, \beta_j, \gamma_{ij}, \lambda_h \geq 0$. The number of (positive) $\alpha_i, \beta_j, \gamma_{ij}, \lambda_h$ is at most $O(n)$. Here we denote separating hyperplanes by $\boldsymbol{h}^T\boldsymbol{x} \leq \hat{f}(\bar{x}_h)$. Let H be the set of BFS's used to construct separating hyperplanes.

$$\boldsymbol{c}^T\boldsymbol{x} = -\sum_i \alpha_i x_i + \sum_j \beta_j x_j + \sum_{(i,j)\in A} \gamma_{ij}(x_i - x_j) + \sum_{h\in H} \lambda_h \boldsymbol{h}^T\boldsymbol{x} \quad \text{and} \quad M = \sum_j \beta_j + \sum_{h\in H} \lambda_h \hat{f}(\bar{x}_h). \tag{10.5}$$

Similarly, we write the inequality for F' as a nonnegative combination of the constraints for P and the number of (positive) $\alpha'_i, \beta'_j, \gamma'_{ij}, \lambda'_h$ is $O(n)$:

$$\boldsymbol{c}'^T\boldsymbol{x} = -\sum \alpha'_i x_i + \sum \beta'_j x_j + \sum_{(i,j)\in A} \gamma'_{ij}(x_i - x_j) + \sum_{h\in H} \lambda'_h \boldsymbol{h}^T\boldsymbol{x} \quad \text{and} \quad M' = \sum \beta'_j + \sum_{h\in H} \lambda'_h \hat{f}(\bar{x}_h). \tag{10.6}$$

We also scale $\boldsymbol{c}, \boldsymbol{c}', \alpha, \alpha', \beta, \beta', \gamma, \gamma', \lambda, \lambda'$ so that

$$\sum_{h\in H} (\lambda_h + \lambda'_h) = 1$$

as this does not change any of our preceding inequalities regarding F and F'.

Now that F, F' have been written as combinations of our constraints, we have gathered the necessary ingredients to derive our new arc. We first give a geometric intuition why we would expect to be able to derive a new constraint. Consider the nonnegative combination making up F. We think of the coefficient β_j as the contribution of $x_j \leq 1$ to F. Now if β_j is very large, F is "very parallel" to $x_j \leq 1$ and consequently F' would miss $x_j = 0$ as the gap between F and F' is small. P would then miss $x_j = 0$ too as it is sandwiched between F and F'. Similarly, a large α_i and a large γ_{ij} would respectively imply that $x_i = 1$ and $(x_i = 0, x_j = 1)$ would be missed. The same argument works for F' as well.

But on the other hand, if the contributions from $x_i \geq 0, x_j \leq 1, x_i \leq x_j$ to both F and F' are small, then the supporting hyperplanes $\boldsymbol{c}^T\boldsymbol{x} \leq \ldots$ and $\boldsymbol{c}'^T\boldsymbol{x} \leq \ldots$ would be mostly made up of

separating hyperplanes $\boldsymbol{h}^T\boldsymbol{x} \le \hat{f}(\bar{x}_h)$. By summing up these separating hyperplanes (whose coefficients form BFS's), we would then get a point in the base polyhedron which is very close to the origin 0. Moreover, by Lemma 10.4.11 and Lemma 10.4.9 we should then be able to deduce some interesting information about the minimizer of f over D.

The rest of this section is devoted to realizing the vision sketched above. We stress that while the algebraic manipulations may be long, they are simply the execution of this elegant geometric picture.

Now, consider the following weighted sum of $\boldsymbol{h}^T\boldsymbol{x} \le \hat{f}(\bar{x}_h)$:

$$\left(\sum_{h\in H}\lambda_h\boldsymbol{h}^T + \sum_{h\in H}\lambda'_h\boldsymbol{h}^T\right)\boldsymbol{x} = \sum_{h\in H}\lambda_h\boldsymbol{h}^T\boldsymbol{x} + \sum_{h\in H}\lambda'_h\boldsymbol{h}^T\boldsymbol{x} \le \sum_{h\in H}\lambda_h\hat{f}(\bar{x}_h) + \sum_{h\in H}\lambda'_h\hat{f}(\bar{x}_h).$$

Observe that $\sum_{h\in H}\lambda_h\boldsymbol{h}^T + \sum_{h\in H}\lambda'_h\boldsymbol{h}^T$ is in the base polyhedron since it is a convex combination of BFS $\boldsymbol{h}$. Furthermore, using (10.5) and (10.6) this can also be written as

$$\begin{aligned}\left(\sum_{h\in H}\lambda_h\boldsymbol{h}^T + \sum_{h\in H}\lambda'_h\boldsymbol{h}^T\right)\boldsymbol{x} = &\left(\boldsymbol{c}^T\boldsymbol{x} + \sum\alpha_i x_i - \sum\beta_j x_j + \sum_{(i,j)\in A}\gamma_{ij}(x_j - x_i)\right)\\ &+ \left(\boldsymbol{c}'^T\boldsymbol{x} + \sum\alpha'_i x_i - \sum\beta'_j x_j + \sum_{(i,j)\in A}\gamma'_{ij}(x_j - x_i)\right)\end{aligned} \tag{10.7}$$

and

$$\begin{aligned}\sum_{h\in H}\lambda_h\hat{f}(\bar{x}_h) + \sum_{h\in H}\lambda'_h\hat{f}(\bar{x}_h) &= \left(M - \sum\beta_j\right) + \left(M' - \sum\beta'_j\right)\\ &= (M + M') - \sum\beta_j - \sum\beta'_j\end{aligned}$$

Furthermore, we can bound $\boldsymbol{c}^T\boldsymbol{x} + \boldsymbol{c}'^T\boldsymbol{x}$ by $\boldsymbol{c}^T\boldsymbol{x} + \boldsymbol{c}'^T\boldsymbol{x} \ge -||\boldsymbol{c}+\boldsymbol{c}'||_1 \ge -\sqrt{n}||\boldsymbol{c}+\boldsymbol{c}'||_2 \ge -\sqrt{n}\mathtt{gap}$ as $\boldsymbol{x} \le 1$. Since $M + M' \le \mathtt{gap}$, we obtain

$$\begin{aligned}LHS \stackrel{\text{def}}{=} &\sum\alpha_i x_i + \sum\alpha'_i x_i - \sum\beta_j x_j - \sum\beta'_j x_j + \sum_{(i,j)\in A}\gamma_{ij}(x_j - x_i) + \sum_{(i,j)\in A}\gamma'_{ij}(x_j - x_i)\\ &\le 2\sqrt{n}\mathtt{gap} - \sum\beta_j - \sum\beta'_j\end{aligned}$$

Geometrically, the next lemma states that if the contribution from, say $x_i \ge 0$, to F is too large, then F' would be forced to miss $x_i = 1$ because they are close to one another.

Lemma 10.4.14. *Suppose $\boldsymbol{x}$ satisfies (10.1), $LHS \le 2\sqrt{n}\mathtt{gap} - \sum\beta_j - \sum\beta'_j$ and $\alpha_i, \beta_j, \gamma_{ij}, \alpha'_i, \beta'_j, \gamma'_{ij} \ge 0$.*

1. *If $\alpha_i > 2\sqrt{n}\mathtt{gap}$ or $\alpha'_i > 2\sqrt{n}\mathtt{gap}$, then $x_i < 1$.*
2. *If $\beta_j > 2\sqrt{n}\mathtt{gap}$ or $\beta'_j > 2\sqrt{n}\mathtt{gap}$, then $x_j > 0$.*
3. *If $\gamma_{ij} > 2\sqrt{n}\mathtt{gap}$ or $\gamma'_{ij} > 2\sqrt{n}\mathtt{gap}$, then $0 \le x_j - x_i < 1$.*

Proof. We only prove it for $\alpha_i, \beta_j, \gamma_{ij}$ as the other case follows by symmetry.

Using $0 \le x \le 1$ and $x_i \le x_j$ for $(i,j) \in A$, we have $LHS \ge \alpha_i x_i - \sum\beta_j - \sum\beta'_j$. Hence $\alpha_i x_i \le 2\sqrt{n}\mathtt{gap}$ and we get $x_i < 1$ if $\alpha_i > 2\sqrt{n}\mathtt{gap}$.

Similarly, $LHS \ge -\beta_k x_k - \sum_{j\ne k}\beta_j - \sum\beta'_j$ which gives $-\beta_k x_k \le 2\sqrt{n}\mathtt{gap} - \beta_k$. Then $x_k > 0$ if $\beta_k > 2\sqrt{n}\mathtt{gap}$.

Finally, $LHS \ge \gamma_{ij}(x_j - x_i) - \sum\beta_j - \sum\beta'_j$ which gives $\gamma_{ij}(x_j - x_i) \le 2\sqrt{n}\mathtt{gap}$. Then $x_j - x_i < 1$ if $\gamma_{ij} > 2\sqrt{n}\mathtt{gap}$. We have $x_i \le x_j$ since $(i,j) \in A$. □

So if either condition of Lemma 10.4.14 holds, we can set $x_i = 0$ or $x_j = 1$ or $x_i = x_j$ since our problem (10.1) has an integral minimizer and any minimizer of $\hat{f}$ is never cut away by Lemma 10.4.5. Consequently, in this case we can reduce the dimension by at least 1. From now on we may assume that

$$\max\{\alpha_i, \alpha'_i, \beta_j, \beta'_j, \gamma_{ij}, \gamma'_{ij}\} \leq 2\sqrt{n}\mathtt{gap}. \tag{10.8}$$

Geometrically, (10.8) says that if the supporting hyperplanes are both mostly made up of the separating hyperplanes, then their aggregate contributions to F and F' should be small in absolute value.

The next lemma identifies some $p \in V$ for which $f(R(p)) - f(R(p) - p)$ is "big". This prepares for the final step of our approach which invokes Lemma 10.4.9.

Lemma 10.4.15. *Let* $\boldsymbol{y} \stackrel{\text{def}}{=} \sum_{h\in H} \lambda_h \boldsymbol{h}$ *and* $\boldsymbol{y}' \stackrel{\text{def}}{=} \sum_{h\in H} \lambda'_h \boldsymbol{h}$ *and let* $p \in \arg\max_l\{\max\{|y_l|, |y'_l|\}\}$ *then*

$$\mathtt{upper}(p) \geq n^7 \|\boldsymbol{y} + \boldsymbol{y}'\|_\infty$$

assuming (10.8).

Proof. Recall that $\|\boldsymbol{c} + \boldsymbol{c}'\|_2 \leq \mathtt{gap}$ where $\mathtt{gap} = \frac{1}{10n^{10}} \min\{||\boldsymbol{c}||_2, ||\boldsymbol{c}'||_2\}$,

$$\boldsymbol{c} = \boldsymbol{y} - \sum_i \alpha_i \mathbf{1}_i + \sum_j \beta_j \mathbf{1}_j + \sum_{(i,j)} \gamma_{ij}(\mathbf{1}_i - \mathbf{1}_j) \quad \text{and } \boldsymbol{c}' = \boldsymbol{y}' - \sum_i \alpha'_i \mathbf{1}_i + \sum_j \beta'_j \mathbf{1}_j + \sum_{(i,j)} \gamma'_{ij}(\mathbf{1}_i - \mathbf{1}_j)\,.$$

By (10.8) we know that $\|\boldsymbol{c} - \boldsymbol{y}\|_2 \leq 4n^2\mathtt{gap} \leq \frac{4}{10n^8}\|\boldsymbol{c}\|_2$ and $\|\boldsymbol{c}' - \boldsymbol{y}'\|_2 \leq 4n^2\mathtt{gap} \leq \frac{4}{10n^8}\|\boldsymbol{c}'\|_2$. Consequently, by the triangle inequality we have that

$$\|\boldsymbol{y} + \boldsymbol{y}'\|_2 \leq \|\boldsymbol{c} + \boldsymbol{c}'\|_2 + \|\boldsymbol{c} - \boldsymbol{y}\|_2 + \|\boldsymbol{c}' - \boldsymbol{y}'\|_2 \leq 9n^2\mathtt{gap}$$

and

$$\|\boldsymbol{c}\|_2 \leq \|\boldsymbol{c} - \boldsymbol{y}\|_2 + \|\boldsymbol{y}\|_2 \leq \frac{4}{10n^8}\|\boldsymbol{c}\|_2 + \|\boldsymbol{y}\|_2 \quad \Rightarrow \quad \|\boldsymbol{c}\|_2 \leq 2\|\boldsymbol{y}\|_2$$

Similarly, we have that $\|\boldsymbol{c}'\|_2 \leq 2\|\boldsymbol{y}'\|_2$. Consequently since $\mathtt{gap} \leq \frac{1}{10n^{10}} \min\{||\boldsymbol{c}||_2, ||\boldsymbol{c}'||_2\}$, we have that

$$\|\boldsymbol{y} + \boldsymbol{y}'\|_2 \leq \frac{2}{n^8} \min\left\{\|\boldsymbol{y}\|_2, \|\boldsymbol{y}'\|_2\right\}$$

and thus, invoking Lemma 10.4.11 yields the result. □

We summarize the results in the lemma below.

Corollary 10.4.16. *Let* P *be the feasible region after running cutting plane on* (10.1)*. Then one of the following holds:*

1. *We found a degenerate BFS and hence either* $\emptyset$ *or* V *is a minimizer.*
2. *The integral points of* P *all lie on some hyperplane* $x_i = 0$, $x_j = 1$ *or* $x_i = x_j$ *which we can find.*
3. *Let* H *be the collection of BFS's* $\boldsymbol{h}$ *used to construct our separating hyperplanes for* P*. Then there is a convex combination* $\boldsymbol{y}$ *of* H *such that* $n^4|y_i| < \max_p \mathtt{upper}(p)$ *for all* i*.*

Proof. As mentioned before, (1) happens if some separating hyperplane is degenerate. We have (2) if one of the conditions in Lemma 10.4.14 holds. Otherwise, $y = \sum_{h\in H} \lambda_h \boldsymbol{h} + \sum_{h\in H} \lambda'_h \boldsymbol{h}$ is a candidate for Case 3 by Lemma 10.4.15. □

Let us revisit the conditions of Lemma 10.4.9 and explain that they are satisfied by Case 3 of the last lemma.

- $\boldsymbol{y}$ is a convex combination of at most $O(n)$ BFS's. This holds in Case 3 since our current feasible region consists of only $O(n)$ constraints thanks to the Cutting Plane method.
- Those BFS's must be consistent with every arc of A. This holds because Case 3 uses the BFS's for constructing our separating hyperplane. Our modified separation oracle guarantees that they are consistent with A.

Thus in Case 3 of the last corollary, Lemma 10.4.9 allows us to deduce a new constraint $x_p \leq x_q$ for some $q \notin R(p)$.

10.4.3.3 Running Time

Here we bound the total running time of our algorithm and prove the following.

Theorem 10.4.17. *Our algorithm runs in time* $O(n^4 \log n \cdot EO + n^5 \log^{O(1)} n)$.

Proof. To avoid being repetitive, we appeal to Corollary 10.4.16. Each phase of cutting plane takes time $O(n^2 \log n \cdot \mathrm{EO} + n^3 \log^{O(1)} n)$ (Theorem 10.4.12 with τ being a big constant. Given F and F' represented as a nonnegative combination of facets, we can check for the conditions in Lemma 10.4.14 in $O(n)$ time as there are only this many facets of P. This settles Case 2 of Corollary 10.4.16. Finally, Lemma 10.4.9 tells us that we can find a new arc in $O(n \cdot \mathrm{EO} + n^2)$ time for Case 3 of Corollary 10.4.16. Our conclusion follows from the fact that we can get $x_i = 0$, $x_i = 1$, $x_i = x_j$ at most n times and $x_i \leq x_j$ at most $O(n^2)$ times. □

10.4.4 $\widetilde{O}(n^3 \cdot \mathrm{EO} + n^4)$ Time Algorithm

Here we show how to improve our running time for strongly polynomial SFM to $\widetilde{O}(n^3 \cdot \mathrm{EO} + n^4)$. Our algorithm can be viewed as an extension of the algorithm we presented in the previous Section 10.4.3. The main bottleneck of our previous algorithm was the time needed to identify a new arc, which cost us $\widetilde{O}(n^2 \cdot \mathrm{EO} + n^3)$. Here we show how to reduce our amortized cost for identifying a valid arc down to $\widetilde{O}(n \cdot \mathrm{EO} + n^2)$ and thereby achieve our result.

The key observation we make to improve this running time is that our choice of p for adding an arc in the previous lemma can be relaxed. p actually need not be $\arg\max_i \texttt{upper}(i)$; instead it is enough to have $\texttt{upper}(p) > n^4 \max\{\alpha_i, \alpha'_i, \beta_j, \beta'_j, \gamma_{ij}, \gamma'_{ij}\}$. For each such p a new constraint $x_p \leq x_q$ can be identified via Lemma 10.4.9. So if there are many p's satisfying this we will be able to obtain many new constraints and hence new valid arcs (p, q).

On the other hand, the bound in Lemma 10.4.15 says that our point in the base polyhedron is small in *absolute* value. This is actually stronger than what we need in Lemma 10.4.9 which requires only its positive entries to be "small". However as we saw in Lemma 10.4.10 we can generate a constraint of the form $x_q \leq x_p$ whenever $\texttt{lower}(p)$ is sufficiently negative.

Using this idea, we divide V into different buckets according to $\texttt{upper}(p)$ and $\texttt{lower}(p)$. This will allow us to get a speedup for two reasons.

First, bucketing allows us to disregard unimportant elements of V during certain executions of our cutting plane method. If both $\texttt{upper}(i)$ and $\texttt{lower}(i)$ are small in absolute value, then i is essentially negligible because for a separating hyperplane $\boldsymbol{h}^T \boldsymbol{x} \leq \hat{f}(\bar{x})$, any $h_i \in [\texttt{lower}(i), \texttt{upper}(i)]$ small in absolute value would not really make a difference. We can then run our cutting plane algorithm only on those non-negligible i's, thereby reducing our time complexity. Of course, whether h_i is small is something relative. This suggests that partitioning the ground set by the relative size of $\texttt{upper}(i)$ and $\texttt{lower}(i)$ is a good idea.

Second, bucketing allows us to ensure that we can always add an arc for many edges simultaneously. Recall that we remarked that all we want is $n^{O(1)}|y_i| \leq \texttt{upper}(p)$ for some $\boldsymbol{y}$ in the base polyhedron. This would be sufficient to identify a new valid arc (p, q). Now if the marginal differences $\texttt{upper}(p)$ and $\texttt{upper}(p')$ are close in value, knowing $n^{O(1)}|y_i| \leq \texttt{upper}(p)$ would effectively give us the same for p' for free. This suggests that elements with similar marginal differences should be grouped together.

The remainder of this section simply formalizes these ideas. In Section 10.4.4.1 we discuss how we partition the ground set V. In Section 10.4.4.2, we present our cutting plane method on a subset of the coordinates. Then in Section 10.4.4.3 we show how we find new arcs. Finally, in Section 10.4.4.4 we put all of this together to achieve our desired running time.

10.4.4.1 Partitioning Ground Set into Buckets

We partition the ground set V into different buckets according to the values of $\texttt{upper}(i)$ and $\texttt{lower}(i)$. This is reminiscent to Iwata-Orlin's algorithm [125] which considers elements with big $\texttt{upper}(i)$. However they did not need to do bucketing by size or to consider $\texttt{lower}(i)$, whereas these seem necessary for our algorithm.

Let $N = \max_i\{\max\{\texttt{upper}(i), -\texttt{lower}(i)\}\}$ be the largest marginal difference in absolute value. By Lemma (10.4.8), $N \geq 0$. We partition our ground set V as follows:

$$B_1 = \{i : \texttt{upper}(i) \geq N/n^{10} \text{ or } \texttt{lower}(i) \leq -N/n^{10}\}$$

$$\begin{aligned} B_k \quad = \{i \notin B_1 \cup \ldots \cup B_{k-1} : \quad & N/n^{10k} \leq \texttt{upper}(i) < N/n^{10(k-1)} \\ & \text{or } -N/n^{10(k-1)} < \texttt{lower}(i) \leq -N/n^{10k}\}, \quad k \geq 2 \end{aligned}$$

We call B_k *buckets*. Our buckets group elements by the values of $\texttt{upper}(i)$ and $\texttt{lower}(i)$ at $1/n^{10}$ "precision". There are two cases.

- Case 1: the number of buckets is at most $\log n$[2], in which case $\texttt{upper}(i) > N/n^{O(\log n)}$ or $\texttt{lower}(i) < -N/n^{O(\log n)}$ for all i.
- Case 2: there is some k for which $|B_1 \cup \ldots \cup B_k| \geq |B_{k+1}|$.

This is because if there is no such k in Case 2, then by induction each bucket B_{k+1} has at least $2^k|B_1| \geq 2^k$ elements and hence $k \leq \log n$.

Case 1 is easier to handle, and is in fact a special case of Case 2. We first informally sketch the treatment for Case 1 which should shed some light into how we deal with Case 2.

We run Cutting Plane for $O(n \log^2 n)$ iterations (i.e. $\tau = \Theta(\log n)$). By Theorem 10.4.12, our feasible region P would be sandwiched by a pair of approximately parallel supporting hyperplanes of width at most $1/n^{\Theta(\log n)}$. Now proceeding as in the last section, we would be able to find some $\boldsymbol{y}$ in the base polyhedron and some element p such that $n^{\Theta(\log n)}|y_i| \leq \texttt{upper}(p)$. This gives

$$n^{\Theta(\log n)}|y_i| \leq \frac{\texttt{upper}(p)}{n^{\Theta(\log n)}} \leq \frac{N}{n^{\Theta(\log n)}}.$$

Since $\texttt{upper}(i) > N/n^{\Theta(\log n)}$ or $\texttt{lower}(i) < -N/n^{\Theta(\log n)}$ for all i in Case 1, we can then conclude that some valid arc (i, q) or (q, i) can be added for every i. Thus we add $n/2$ arcs simultaneously in one phase of the algorithm at the expense of blowing up the runtime by $O(\log n)$. This saves a factor of $n/\log n$ from our runtime in the last section, and the amortized cost for an arc would then be $\widetilde{O}(n \cdot \text{EO} + n^2)$.

[2] More precisely, $B_k = \emptyset$ for $k > \lceil \log n \rceil$.

On the other hand, in Case 2 we have a "trough" at B_{k+1}. Roughly speaking, this trough is useful for acting as a soft boundary between $B_1 \cup \ldots \cup B_k$ and $\bigcup_{l \geq k+2} B_l$. Recall that we are able to "ignore" $\bigcup_{l \geq k+2} B_l$ because their h_i is relatively small in absolute value. In particular, we know that for any $p \in B_1 \cup \ldots \cup B_k$ and $i \in B_l$, where $l \geq k+2$,

$$\max\{\texttt{upper}(p), -\texttt{lower}(p)\} \geq n^{10} \max\{\texttt{upper}(i), -\texttt{lower}(i)\}.$$

This is possible because B_{k+1}, which is sandwiched in between, acts like a shield preventing B_l to "mess with" $B_1 \cup \ldots \cup B_k$. This property comes at the expense of sacrificing B_{k+1} which must confront B_l.

Furthermore, we require that $|B_1 \cup \ldots \cup B_k| \geq |B_{k+1}|$, and run Cutting Plane on $B = (B_1 \cup \ldots \cup B_k) \cup B_{k+1}$. If $|B_{k+1}| \gg |B_1 \cup \ldots \cup B_k|$, our effort would mostly be wasted on B_{k+1} which is sacrificed, and the amortized time complexity for $B_1 \cup \ldots \cup B_k$ would then be large.

Before discussing the algorithm for Case 2, we need some preparatory work.

10.4.4.2 Separating Hyperplane: Project and Lift

Our speedup is achieved by running our cutting plane method on the projection of our feasible region onto $B := (B_1 \cup \cdots \cup B_k) \cup B_{k+1}$. More precisely, we start by running our cutting plane on $P^B = \{\boldsymbol{x} \in \mathbb{R}^B : \exists \boldsymbol{x'} \in \mathbb{R}^{\bar{B}} \text{ s.t. } (\boldsymbol{x}, \boldsymbol{x'}) \text{ satisfies (10.1)}\}$, which has a lower dimension. However, to do this, we need to specify a separation oracle for P^B. Here we make one of the most natural choices.

We begin by making an apparently immaterial change to our set of arcs A. Let us take the *transitive closure* of A by adding the arc (i, j) whenever there is a path from i to j. Clearly this would not change our ring family as a path from i to j implies $j \in R(i)$. Roughly speaking, we do this to handle pathological cases such as $(i, k), (k, j) \in A, (i, j) \notin A$ and $i, j \in B, k \notin B$. Without introducing the arc (i, j), we risk confusing a solution containing i but not j as feasible since we are restricting our attention to B and ignoring $k \notin B$.

Definition 10.4.18. Given a digraph $D = (V, A)$, the transitive closure of A is the set of arcs (i, j) for which there is a directed path from i to j. We say that A is *complete* if it is equal to its transitive closure.

Given $\bar{x} \in [0, 1]^B$, we define the completion of $\bar{x}$ with respect to A as follows.

Definition 10.4.19. Given $\bar{x} \in [0, 1]^B$ and a set of arcs A, $x^{\mathcal{C}} \in [0, 1]^n$ is a completion of $\bar{x}$ if $x^{\mathcal{C}}_B = \bar{x}$ and $x^{\mathcal{C}}_i \leq x^{\mathcal{C}}_j$ for every $(i, j) \in A$. Here $x^{\mathcal{C}}_B$ denotes the restriction of $x^{\mathcal{C}}$ to B.

Lemma 10.4.20. *Given $\bar{x} \in [0, 1]^B$ and a complete set of arcs A, there is a completion of $\bar{x}$ if $\bar{x}_i \leq \bar{x}_j$ for every $(i, j) \in A \cap (B \times B)$. Moreover, it can be computed in $O(n^2)$ time.*

Proof. We set $x^{\mathcal{C}}_B = \bar{x}$. For $i \notin B$, we set

$$x^{\mathcal{C}}_i = \begin{cases} 1 & \text{if } \nexists j \in B \text{ s.t. } (i, j) \in A \\ \min_{(i,j) \in A, j \in B} x^{\mathcal{C}}_j & \text{otherwise} \end{cases}$$

One may verify that $x^{\mathcal{C}}$ satisfies our requirement as A is complete. Computing each $x^{\mathcal{C}}_i$ takes $O(n)$ time. Since $|V \backslash B| = |\bar{B}| \leq n$, computing the whole $x^{\mathcal{C}}$ takes $O(n^2)$ time. □

This notion of completion is needed since our original separation oracle requires a full dimensional input $\bar{x}$. Now that $\bar{x} \in \mathbb{R}^B$, we need a way of extending it to $\mathbb{R}^n$ while retaining the crucial property that $\boldsymbol{h}$ is consistent with every arc in A.

Algorithm 26: Projected Separation Oracle

Input: $\bar{x} \in \mathbb{R}^B$ and a complete set of arcs A
if $\bar{x}_i < 0$ *for some* $i \in B$ **then**
  Output: $x_i \geq 0$
else if $\bar{x}_j > 1$ *for some* $j \in B$ **then**
  Output: $x_j \leq 1$
else if $\bar{x}_i > \bar{x}_j$ *for some* $(i,j) \in A \cap B^2$ **then**
  Output: $x_i \leq x_j$
else
  Let $x^{\mathcal{C}} \in \mathbb{R}^n$ be a completion of $\bar{x}$
  Let $i_1, \ldots, i_n$ be a permutation of V such that $x^{\mathcal{C}}_{i_1} \geq \ldots \geq x^{\mathcal{C}}_{i_n}$ and for all $(i,j) \in A$, j precedes i in $i_1, \ldots, i_n$.
  Output: $\boldsymbol{h}_B^T \boldsymbol{x}_B = \sum_{i \in B} h_i x_i \leq \sum_{i \in B} h_i \bar{x}_i$, where $\boldsymbol{h}$ is the BFS defined by the permutation $i_1, \ldots, i_n$.

Note that the runtime is still $O(n \cdot \mathrm{EO} + n^2 \log^{O(1)} n)$ as $x^{\mathcal{C}}$ can be computed in $O(n^2)$ time by the last lemma.

We reckon that the hyperplane $\boldsymbol{h}_B^T \boldsymbol{x}_B \leq \sum_{i \in B} h_i \bar{x}_i$ returned by the oracle is *not* a valid separating hyperplane (i.e. it may cut out the minimizers). Nevertheless, we will show that it is a decent "proxy" to the true separating hyperplane $\boldsymbol{h}^T \boldsymbol{x} \leq \hat{f}(x^{\mathcal{C}}) = \sum_{i \in V} h_i x^{\mathcal{C}}_i$ and is good enough to serve our purpose of sandwiching the remaining feasible region in a small strip. To get a glimpse, note that the terms missing $\boldsymbol{h}_B^T \boldsymbol{x}_B \leq \sum_{i \in B} h_i \bar{x}_i$ all involve h_i for $i \notin B$, which is "negligible" compared to $B_1 \cup \cdots \cup B_k$.

One may try to make $\boldsymbol{h}_B^T \boldsymbol{x}_B \leq \sum_{i \in B} h_i \bar{x}_i$ valid, say, by $\boldsymbol{h}_B^T \boldsymbol{x}_B \leq \sum_{i \in B} h_i \bar{x}_i + \sum_{i \notin B} |h_i|$. The problem is that such hyperplanes would not be separating for $\bar{x}$ anymore as $\boldsymbol{h}_B^T \bar{x} = \sum_{i \in B} h_i \bar{x}_i < \sum_{i \in B} h_i \bar{x}_i + \sum_{i \notin B} |h_i|$. Consequently, we lose the width (or volume) guarantee of our cutting plane algorithm. Although this seems problematic, it is actually still possible to show a guarantee sufficient for our purpose as $\sum_{i \notin B} |h_i|$ is relatively small. We leave it as a nontrivial exercise to interested readers.

In conclusion, it seems that one cannot have the best of both worlds: the hyperplane returned by the oracle cannot be simultaneously valid and separating.

Algorithm

We take k to be the first for which $|B_1 \cup \ldots \cup B_k| \geq |B_{k+1}|$, i.e. $|B_1 \cup \ldots \cup B_l| < |B_{l+1}|$ for $l \leq k-1$. Thus $k \leq \log n$. Let $b = |B|$, and so $|B_1 \cup \cdots \cup B_k| \geq b/2$. Case 1 is a special case by taking $B = V$.

Our algorithm is summarized below. Here A is always complete as A is replaced its transitive closure whenever a new valid arc is added.

1. Run Cutting Plane on $P^B = \{x \in \mathbb{R}^B : \exists x' \in \mathbb{R}^{\bar{B}} \text{ s.t. } (x, x') \text{ satisfies (10.1)}\}$ with the new projected separation oracle.
2. Identify a pair of "narrow" approximately parallel supporting hyperplanes.
3. Deduce from the hyperplanes certain new constraints of the forms $x_i = 0, x_j = 1, x_i = x_j$ or $x_i \leq x_j$ by lifting separating hyperplanes back to $\mathbb{R}^n$
4. Consolidate A and f. If some $x_i \leq x_j$ added, replace A by its transitive closure.
5. Repeat Step 1 with updated A and f. (Any previously found separating hyperplanes are discarded.)

The minimizer can be constructed by unraveling the recursion.

First of all, to be able to run Cutting Plane on P^B we must come up with a polyhedral description of P^B which consists of just the constraints involving B. This is shown in the next lemma.

Lemma 10.4.21. *Let* $P^B = \{\boldsymbol{x} \in \mathbb{R}^B : \exists \boldsymbol{x}' \in \mathbb{R}^{\bar{B}} \text{ s.t. } (\boldsymbol{x}, \boldsymbol{x}') \text{ satisfies (10.1)}\}$. *Then*

$$P^B = \{\boldsymbol{x} \in \mathbb{R}^B : 0 \le \boldsymbol{x} \le 1, x_i \le x_j \forall (i,j) \in A \cap (B \times B)\}$$

Proof. It is clear that $P^B \subseteq \{\boldsymbol{x} \in \mathbb{R}^B : 0 \le \boldsymbol{x} \le 1, x_i \le x_j \forall (i,j) \in A \cap (B \times B)\}$ as the constraints $0 \le x \le 1, x_i \le x_j \forall (i,j) \in A \cap (B \times B)$ all appear in (10.1).

Conversely, for any $\boldsymbol{x} \in \mathbb{R}^B$ satisfying $0 \le \boldsymbol{x} \le 1, x_i \le x_j \forall (i,j) \in A \cap (B \times B)$, we know there is some completion $x^{\mathcal{C}}$ of $\boldsymbol{x}$ by Lemma 10.4.20 as A is complete. Now $x^{\mathcal{C}}$ satisfies (10.1) by definition, and hence $\boldsymbol{x} \in P^B$. □

The only place where we have really changed the algorithm is Step (3).

10.4.4.3 Deducing New Constraints $x_i = 0$, $x_j = 1$, $x_i = x_j$ or $x_i \le x_j$

Our method will deduce one of the following:

- $x_i = 0$, $x_j = 1$ or $x_i = x_j$
- for each $p \in B_1 \cup \cdots \cup B_k$, $x_p \le x_q$ for some $q \notin R(p)$ or $x_p \ge x_q$ for some $q \notin Q(p)$

Our argument is very similar to the last section's. Roughly speaking, it is the same argument but with "noise" introduced by $i \notin B$. We use extensively the notations from the last section.

Our main tool is again Theorem 10.4.12. Note that n should be replaced by b in the Theorem statement. We invoke it with $\tau = k \log_b n = O(\log^2 n)$ (using $k \le \log n$) to get a width of $1/b^{\Theta(\tau)} = 1/n^{\Theta(k)}$. This takes time at most $O(bn \log^2 n \cdot \mathrm{EO} + bn^2 \log^{O(1)} n)$. Again, this is intuitively clear as we run it for $O(kb \log n)$ iterations, each of which takes time $O(n \cdot \mathrm{EO} + n^2 \log^{O(1)} n)$.

After each phase of (roughly $O(kb \log n)$ iterations) of Cutting Plane, P^B is sandwiched between a pair of approximately parallel supporting hyperplanes F and F' which have width $1/n^{20k}$. Let F and F' be

$$\boldsymbol{c}^T \boldsymbol{x}_B = \sum_{i \in B} c_i x_i \le M, \quad \boldsymbol{c}'^T \boldsymbol{x}_B = \sum_{i \in B} c'_i x_i \le M',$$

such that

$$|M + M'|, \ ||\boldsymbol{c} + \boldsymbol{c}'||_2 \le \mathtt{gap}, \quad \text{where } \mathtt{gap} = \frac{1}{n^{20k}} \min\{||\boldsymbol{c}||_2, ||\boldsymbol{c}'||_2\}.$$

The rest of this section presents an execution of the ideas discussed above. All of our work is basically geared towards bringing the amortized cost for identifying a valid arc down to $\widetilde{O}(n \cdot \mathrm{EO} + n^2)$. Again, we can write these two constraints as a nonnegative combination. Here $\bar{x}_h^{\mathcal{C}}$ is the completion of the point $\bar{x}_h$ used to construct $\boldsymbol{h}_B^T \boldsymbol{x}_B \le \boldsymbol{h}_B^T \left(\bar{x}_h^{\mathcal{C}}\right)_B$. (Recall that $\left(\bar{x}_h^{\mathcal{C}}\right)_B$ is the restriction of $\bar{x}_h^{\mathcal{C}}$ to B.)

$$\boldsymbol{c}^T \boldsymbol{x}_B = -\sum_{i \in B} \alpha_i x_i + \sum_{j \in B} \beta_j x_j + \sum_{(i,j) \in A \cap B^2} \gamma_{ij}(x_i - x_j) + \sum_{h \in H} \lambda_h \boldsymbol{h}_B^T \boldsymbol{x}_B \quad \text{and} \quad M = \sum_{j \in B} \beta_j + \sum_{h \in H} \lambda_h \boldsymbol{h}_B^T \left(\bar{x}_h^{\mathcal{C}}\right)_B.$$

$$\boldsymbol{c}'^T \boldsymbol{x}_B = -\sum_{i \in B} \alpha'_i x_i + \sum_{j \in B} \beta'_j x_j + \sum_{(i,j) \in A \cap B^2} \gamma'_{ij}(x_i - x_j) + \sum_{h \in H} \lambda'_h \boldsymbol{h}_B^T \boldsymbol{x}_B \quad \text{and} \quad M' = \sum_{j \in B} \beta'_j + \sum_{h \in H} \lambda'_h \boldsymbol{h}_B^T \left(\bar{x}_h^{\mathcal{C}}\right)_B.$$

As we have discussed, the problem is that the separating hyperplanes $\boldsymbol{h}_B^T \boldsymbol{x}_B \leq \boldsymbol{h}_B^T \left(\bar{x}_h^{\mathcal{C}}\right)_B$ are not actually valid. We can, however, recover their valid counterpart by lifting them back to $\boldsymbol{h}^T \boldsymbol{x} \leq \boldsymbol{h}^T \bar{x}_h^{\mathcal{C}}$. The hope is that $\boldsymbol{h}_B^T \boldsymbol{x}_B \leq \boldsymbol{h}_B^T \left(\bar{x}_h^{\mathcal{C}}\right)_B$ and $\boldsymbol{h}^T \boldsymbol{x} \leq \boldsymbol{h}^T \bar{x}_h^{\mathcal{C}}$ are not too different so that the arguments will still go through. We show that this is indeed the case.

Again, we scale $c, c', \alpha, \alpha', \beta, \beta', \gamma, \gamma', \lambda, \lambda'$ so that

$$\sum_{h \in H} (\lambda_h + \lambda'_h) = 1.$$

By adding all the constituent separating hyperplane inequalities, we get

$$\sum_{h \in H} \lambda_h \boldsymbol{h}^T \boldsymbol{x} + \sum_{h \in H} \lambda'_h \boldsymbol{h}^T \boldsymbol{x} \leq \sum_{h \in H} \lambda_h \boldsymbol{h}^T \bar{x}_h^{\mathcal{C}} + \sum_{h \in H} \lambda'_h \boldsymbol{h}^T \bar{x}_h^{\mathcal{C}}$$

Let

$$LHS \stackrel{\text{def}}{=} \sum \alpha_i x_i + \sum \alpha'_i x_i - \sum \beta_j x_j - \sum \beta'_j x_j + \sum \gamma_{ij}(x_j - x_i) + \sum \gamma'_{ij}(x_j - x_i).$$

Here we know that

$$\sum_{h \in H} \lambda_h \boldsymbol{h}^T \boldsymbol{x} + \sum_{h \in H} \lambda'_h \boldsymbol{h}^T \boldsymbol{x} = LHS + (\boldsymbol{c} + \boldsymbol{c}')^T \boldsymbol{x}_B + \sum_{h \in H} \lambda_h \boldsymbol{h}_{\bar{B}}^T \boldsymbol{x}_{\bar{B}} + \sum_{h \in H} \lambda'_h \boldsymbol{h}_{\bar{B}}^T \boldsymbol{x}_{\bar{B}}$$

$$\sum_{h \in H} \lambda_h \boldsymbol{h}^T \bar{x}_h^{\mathcal{C}} + \sum_{h \in H} \lambda'_h \boldsymbol{h}^T \bar{x}_h^{\mathcal{C}} = (M + M') + \sum_{h \in H} \lambda_h \boldsymbol{h}_{\bar{B}}^T \left(\bar{x}_h^{\mathcal{C}}\right)_{\bar{B}} + \sum_{h \in H} \lambda'_h \boldsymbol{h}_{\bar{B}}^T \left(\bar{x}_h^{\mathcal{C}}\right)_{\bar{B}} - \sum \beta_j - \sum \beta'_j$$

Combining all yields

$$\begin{aligned} & LHS + (\boldsymbol{c} + \boldsymbol{c}')^T \boldsymbol{x}_B + \sum_{h \in H} \lambda_h \boldsymbol{h}_{\bar{B}}^T \boldsymbol{x}_{\bar{B}} + \sum_{h \in H} \lambda'_h \boldsymbol{h}_{\bar{B}}^T \boldsymbol{x}_{\bar{B}} \\ \leq \; & (M + M') + \sum_{h \in H} \lambda_h \boldsymbol{h}_{\bar{B}}^T \left(\bar{x}_h^{\mathcal{C}}\right)_{\bar{B}} + \sum_{h \in H} \lambda'_h \boldsymbol{h}_{\bar{B}}^T \left(\bar{x}_h^{\mathcal{C}}\right)_{\bar{B}} - \sum \beta_j - \sum \beta'_j. \end{aligned}$$

Here $(\boldsymbol{c} + \boldsymbol{c}')^T \boldsymbol{x}_B$ can be bounded as before: $(\boldsymbol{c} + \boldsymbol{c}')^T \boldsymbol{x}_B \geq -\sqrt{n} ||\boldsymbol{c} + \boldsymbol{c}'||_2 \geq -\sqrt{n}\mathsf{gap}$. Since $M + M' \leq \mathsf{gap}$, We then obtain

$$LHS + \sum_{h \in H} \lambda_h \boldsymbol{h}_{\bar{B}}^T \boldsymbol{x}_{\bar{B}} + \sum_{h \in H} \lambda'_h \boldsymbol{h}_{\bar{B}}^T \boldsymbol{x}_{\bar{B}} \leq 2\sqrt{n}\mathsf{gap} + \sum_{h \in H} \lambda_h \boldsymbol{h}_{\bar{B}}^T \left(\bar{x}_h^{\mathcal{C}}\right)_{\bar{B}} + \sum_{h \in H} \lambda'_h \boldsymbol{h}_{\bar{B}}^T \left(\bar{x}_h^{\mathcal{C}}\right)_{\bar{B}} - \sum \beta_j - \sum \beta'_j$$

We should expect the contribution from $\boldsymbol{h}_{\bar{B}}$ to be small as h_i for $i \notin B$ is small compared to $B_1 \cup \ldots \cup B_k$. We formalize our argument in the next two lemmas.

Lemma 10.4.22. *We have* $\sum_{h \in H} \lambda_h \boldsymbol{h}_{\bar{B}}^T \left(\bar{x}_h^{\mathcal{C}}\right)_{\bar{B}} + \sum_{h \in H} \lambda'_h \boldsymbol{h}_{\bar{B}}^T \left(\bar{x}_h^{\mathcal{C}}\right)_{\bar{B}} \leq N/n^{10(k+1)-1}$.

Proof. We bound each component of $\sum_{h \in H} \lambda_h \boldsymbol{h}_{\bar{B}}^T \left(\bar{x}_h^{\mathcal{C}}\right)_{\bar{B}} + \sum_{h \in H} \lambda'_h \boldsymbol{h}_{\bar{B}}^T \left(\bar{x}_h^{\mathcal{C}}\right)_{\bar{B}}$. For $i \in \bar{B}$, we have $\mathtt{upper}(i) \leq N/n^{10(k+1)}$. By Lemma 10.4.8 $h_i \leq \mathtt{upper}(i)$. Therefore,

$$\sum_{h \in H} \lambda_h \boldsymbol{h}_i^T \left(\bar{x}_h^{\mathcal{C}}\right)_i + \sum_{h \in H} \lambda'_h \boldsymbol{h}_i^T \left(\bar{x}_h^{\mathcal{C}}\right)_i \;\leq\; \left(\sum_{h \in H} \lambda_h + \sum_{h \in H} \lambda'_h\right) N/n^{10(k+1)} = N/n^{10(k+1)}.$$

Our result then follows since

$$\sum_{h \in H} \lambda_h \boldsymbol{h}_{\bar{B}}^T \left(\bar{x}_h^{\mathcal{C}}\right)_{\bar{B}} + \sum_{h \in H} \lambda'_h \boldsymbol{h}_{\bar{B}}^T \left(\bar{x}_h^{\mathcal{C}}\right)_{\bar{B}} = \sum_{i \in \bar{B}} \left(\sum_{h \in H} \lambda_h \boldsymbol{h}_i^T \left(\bar{x}_h^{\mathcal{C}}\right)_i + \sum_{h \in H} \lambda'_h \boldsymbol{h}_i^T \left(\bar{x}_h^{\mathcal{C}}\right)_i\right).$$

□

Lemma 10.4.23. *We have* $\sum_{h\in H}\lambda_h \boldsymbol{h}_{\bar{B}}^T\boldsymbol{x}_{\bar{B}} + \sum_{h\in H}\lambda'_h \boldsymbol{h}_{\bar{B}}^T\boldsymbol{x}_{\bar{B}} \geq -N/n^{10(k+1)-1}$.

Proof. The proof is almost identical to the last lemma except that we use $h_i \geq \texttt{lower}(i)$ instead of $h_i \leq \texttt{upper}(i)$, and $\texttt{lower}(i) \geq -N/n^{10(k+1)}$. □

The two lemmas above imply that

$$LHS \leq 2\sqrt{n}\texttt{gap} - \sum\beta_j - \sum\beta_j + 2N/n^{10(k+1)-1} = \texttt{gap}' - \sum\beta_j - \sum\beta_j$$

where $\texttt{gap}' = 2\sqrt{n}\texttt{gap} + 2N/n^{10(k+1)-1}$.

Lemma 10.4.24. *Suppose x satisfies (10.1) and $LHS \leq \texttt{gap}' - \sum\beta_j - \sum\beta'_j$ with $\alpha_i, \beta_j, \gamma_{ij}, \alpha'_i, \beta'_j, \gamma'_{ij} \geq 0$.*

1. *If $\alpha_i > \texttt{gap}'$ or $\alpha'_i > \texttt{gap}'$, then $x_i < 1$.*
2. *If $\beta_j > \texttt{gap}'$ or $\beta'_j > \texttt{gap}'$, then $x_j > 0$.*
3. *If $\gamma_{ij} > \texttt{gap}'$ or $\gamma'_{ij} > \texttt{gap}'$, then $0 \leq x_j - x_i < 1$.*

Proof. The proof is exactly the same as Lemma 10.4.14 with $2\sqrt{n}\texttt{gap}$ replaced by $\texttt{gap}'$. □

From now on we may assume that

$$\max\{\alpha_i, \alpha'_i, \beta_j, \beta'_j, \gamma_{ij}, \gamma'_{ij}\} \leq \texttt{gap}'. \tag{10.9}$$

Lemma 10.4.25. *Let $\boldsymbol{y} \stackrel{\text{def}}{=} \sum_{h\in H}\lambda_h\boldsymbol{h}$ and $\boldsymbol{y}' \stackrel{\text{def}}{=} \sum_{h\in H}\lambda'_h\boldsymbol{h}$ and let $p \in \arg\max_{l\in B}\{\max\{|y_l|, |y'_l|\}$ then*

$$N \geq n^{10k+6}\|\boldsymbol{y}_B + \boldsymbol{y}'_B\|_\infty$$

assuming (10.9).

Proof. Recall that $\|\boldsymbol{c} + \boldsymbol{c}'\|_2 \leq \texttt{gap} < \texttt{gap}'$ where $\texttt{gap} = \frac{1}{n^{20k}}\min\{||\boldsymbol{c}||_2, ||\boldsymbol{c}'||_2\}$ and $\texttt{gap}' = 2\sqrt{n}\texttt{gap} + 2N/n^{10(k+1)-1}$. Now there are two cases.

Case 1: $2\sqrt{n}\texttt{gap} \geq 2N/n^{10(k+1)-1}$. Then $\texttt{gap}' \leq 4\sqrt{n}\texttt{gap}$ and we follow the same proof of Lemma 10.4.15. We have

$$\boldsymbol{c} = \boldsymbol{y}_B - \sum_i \alpha_i\mathbf{1}_i + \sum_j \beta_j\mathbf{1}_j + \sum_{(i,j)}\gamma_{ij}(\mathbf{1}_i - \mathbf{1}_j) \quad \text{and } \boldsymbol{c}' = \boldsymbol{y}'_B - \sum_i \alpha'_i\mathbf{1}_i + \sum_j \beta'_j\mathbf{1}_j + \sum_{(i,j)}\gamma'_{ij}(\mathbf{1}_i - \mathbf{1}_j)\,.$$

By (10.9) we know that $\|\boldsymbol{c} - \boldsymbol{y}_B\|_2 \leq 4n^2\texttt{gap}' \leq \frac{1}{n^{17k}}\|\boldsymbol{c}\|_2$ and $\|\boldsymbol{c}' - \boldsymbol{y}'_B\|_2 \leq 4n^2\texttt{gap}' \leq \frac{1}{n^{17k}}\|\boldsymbol{c}\|_2$. Consequently, by the triangle inequality we have that

$$\|\boldsymbol{y}_B + \boldsymbol{y}'_B\|_2 \leq \|\boldsymbol{c} + \boldsymbol{c}'\|_2 + \|\boldsymbol{c} - \boldsymbol{y}_B\|_2 + \|\boldsymbol{c}' - \boldsymbol{y}'_B\|_2 \leq 9n^2\texttt{gap}'$$

and

$$\|\boldsymbol{c}\|_2 \leq \|\boldsymbol{c} - \boldsymbol{y}_B\|_2 + \|\boldsymbol{y}_B\|_2 \leq \frac{1}{n^{17k}}\|\boldsymbol{c}\|_2 + \|\boldsymbol{y}_B\|_2 \quad \Rightarrow \quad \|\boldsymbol{c}\|_2 \leq 2\|\boldsymbol{y}_B\|_2$$

Similarly, we have that $\|\boldsymbol{c}'\|_2 \leq 2\|\boldsymbol{y}'_B\|_2$. Consequently since $\texttt{gap}' \leq \frac{1}{n^{19k}}\min\{||\boldsymbol{c}||_2, ||\boldsymbol{c}'||_2\}$, we have that

$$\|\boldsymbol{y}_B + \boldsymbol{y}'_B\|_2 \leq \frac{18}{n^{17k}}\min\left\{\|\boldsymbol{y}_B\|_2, \|\boldsymbol{y}'_B\|_2\right\}$$

and thus, invoking Lemma 10.4.11 yields $N \geq \texttt{upper}(p) \geq n^{16k}\|\boldsymbol{y}_B + \boldsymbol{y}'_B\|_\infty$, as desired.

Case 2: $2\sqrt{n}\texttt{gap} < 2N/n^{10(k+1)-1}$. Then for any $i \in B$, $|c_i + c'_i| \leq ||\boldsymbol{c} + \boldsymbol{c}'||_2 \leq \texttt{gap} < 2N/n^{10(k+1)-1}$. Since

$$\boldsymbol{y}_B + \boldsymbol{y}'_B = (\boldsymbol{c} + \boldsymbol{c}') + \sum_i \alpha_i \mathbf{1}_i - \sum_j \beta_j \mathbf{1}_j - \sum_{(i,j)} \gamma_{ij}(\mathbf{1}_i - \mathbf{1}_j) + \sum_i \alpha'_i \mathbf{1}_i - \sum_j \beta'_j \mathbf{1}_j - \sum_{(i,j)} \gamma'_{ij}(\mathbf{1}_i - \mathbf{1}_j)$$

we have

$$\|\boldsymbol{y}_B + \boldsymbol{y}'_B\|_\infty \leq 2N/n^{10(k+1)-1} + 2n^{1.5}\texttt{gap}' \leq N/n^{10k+7}.$$

□

Corollary 10.4.26. *Let P be the feasible region after running Cutting Plane on (10.1) with the projected separation oracle. Then one of the following holds:*

1. *We found a BFS $\boldsymbol{h}$ with $\boldsymbol{h}_B = 0$.*
2. *The integral points of P all lie on some hyperplane $x_i = 0, x_j = 1$ or $x_i = x_j$.*
3. *Let H be the collection of BFS's $\boldsymbol{h}$ used to construct our separating hyperplanes for P. Then there is a convex combination $\boldsymbol{y}$ of H such that for $p \in B_1 \cup \cdots \cup B_k$, we have $n^4|y_i| < \texttt{upper}(p)$ or $\texttt{lower}(p) < -n^4|y_i|$ for all i.*

Proof. As mentioned before, (1) happens if some separating hyperplane satisfies $\boldsymbol{h}_B = 0$ when running cutting plane on the non-negligible coordinates. We have (2) if some condition in Lemma 10.4.24 holds. Otherwise, we claim $y = \sum_{\boldsymbol{h}} \lambda_{\boldsymbol{h}} \boldsymbol{h} + \sum_{\boldsymbol{h}} \lambda'_{\boldsymbol{h}} \boldsymbol{h}$ is a candidate for Case 3. y is a convex combination of BFS and by Lemma 10.4.25, for the big elements $i \in B$ we have

$$|y_i| \leq N/n^{10k+6} \leq \frac{1}{n^4} \max\{\texttt{upper}(p), -\texttt{lower}(p)\}.$$

where the last inequality holds since for $p \in B_1 \cup \cdots \cup B_k$, $\max\{\texttt{upper}(p), -\texttt{lower}(p)\} \geq N/n^{10k}$.

On the other hand, for the small elements $i \notin B$, $|y_i| \leq N/n^{10(k+1)} \leq \frac{1}{n^4} \max\{\texttt{upper}(p), -\texttt{lower}(p)\}$ as desired. □

The gap is then smaller enough to add an arc for each $p \in B_1 \cup \cdots \cup B_k$ by Lemmas 10.4.9 and 10.4.10. Therefore we can add a total of $|B_1 \cup \cdots \cup B_k|/2 \geq b/4$ arcs with roughly $O(kb \log n) = \widetilde{O}(b)$ iterations of Cutting Plane, each of which takes $\widetilde{O}(n \cdot \mathrm{EO} + n^2)$. That is, the amortized cost for each arc is $\widetilde{O}(n \cdot \mathrm{EO} + n^2)$. We give a more formal time analysis in below but it should be somewhat clear why we have the desired time complexity.

Lemma 10.4.27. *Suppose there is a convex combination $\boldsymbol{y}$ of H such that for $p \in B_1 \cup \cdots \cup B_k$, we have $n^4|y_i| < \texttt{upper}(p)$ or $\texttt{lower}(p) < -n^4|y_i|$ for all i. Then we can identify at least $b/4$ new valid arcs.*

Proof. We have $|H| = O(n)$ since H is the set of BFS's used for the constraints of P which has $O(n)$ constraints. By Lemmas 10.4.9 and 10.4.10, for $p \in B_1 \cup \cdots \cup B_k$ we can add a new valid arc (p, q) or (q, p). However note that a new arc (p_1, p_2) may added twice by both p_1 and p_2. Therefore the total number of new arcs is only at least $|B_1 \cup \cdots \cup B_k|/2 \geq b/4$. □

10.4.4.4 Running Time

Not much changes to the previous runtime analysis are needed. To avoid repetition, various details already present in the corresponding part of the last section are omitted. Recall $k \leq \log n$, and of course, $b \leq n$.

For each (roughly) $O(kb \log n)$ iterations of Cutting Plane we either get $x_i = 0$,$x_i = 1$,$x_i = x_j$ or $b/4$ $x_i \leq x_j$'s. The former can happen at most n times while in the latter case, the amortized cost of each arc is $O(k \log n)$ iterations of Cutting Plane. In the worst case the overall number of iterations required is $\widetilde{O}(n^2)$. Thus our algorithm has a runtime of $\widetilde{O}(n^3 \cdot \mathrm{EO} + n^4)$ since each iteration is $\widetilde{O}(n \cdot \mathrm{EO} + n^2)$ as shown below.

Theorem 10.4.28. *Our algorithm runs in time $O(n^3 \log^2 n \cdot EO + n^4 \log^{O(1)} n)$.*

Proof. We use Corollary 10.4.26. First we note that Case 1 can actually be integrated into Case 3 since $\max\{\texttt{upper}(p), -\texttt{lower}(p)\} \geq N/n^{10k} = n^{10}N/n^{10(k+1)} \geq h_i$ for $i \notin B$.

As we have argued in the beginning of the last section, Theorem 10.4.12 with $\tau = k \log_b n$ implies that the runtime for each phase is $O(bn \log^2 n \cdot \mathrm{EO} + bn^2 \log^{O(1)} n)$. In each phase we either get $x_i = 0$, $x_i = 1$, $x_i = x_j$ (Case 2) or $b/4$ $x_i \leq x_j$'s (Case 3), the latter of which follows from Corollary 10.4.26 and Lemma 10.4.27.

Case 2 can only happen n times. Thus the total cost is at most $O(n^3 \log^2 n \cdot \mathrm{EO} + n^4 \log^{O(1)} n)$. The overhead cost is also small. Similar to before, given F and F' represented as a nonnegative combination of facets, we can check for the conditions in Lemma 10.4.24 in $O(n)$ time as there are only this many facets of P. This settles Case 2.

For case 3 the amortized cost for each arc is $O(n \log^2 n \cdot \mathrm{EO} + n^2 \log^{O(1)} n)$. Our desired runtime follows since there are only $O(n^2)$ arcs to add. Unlike Case 2 some extra care is needed to handle the overhead cost. The time needed to deduce a new arc (applying Lemmas 10.4.9 and 10.4.10 to $\boldsymbol{y}$ and $p \in B_1 \cup \cdots \cup B_k$) is still $O(n \cdot \mathrm{EO} + n^2)$. But as soon as we get a new arc, we must update A to be its transitive closure so that it is still complete. Given A complete and a new arc $(p, q) \notin A$, we can simply add the arcs from the ancestors of p to q and from p to the descendants of q. There are at most $O(n)$ arcs to add so this takes time $O(n^2)$ per arc, which is okay. □

■ 10.5 Discussion and Comparison with Previous Algorithms

We compare and contrast our algorithms with the previous ones. We focus primarily on strongly polynomial time algorithms.

Convex combination of BFS's

All of the previous algorithms maintain a convex combination of BFS's and iteratively improve over it to get a better primal solution. In particular, the new BFS's used are typically obtained by making local changes to existing ones. Our algorithms, on the other hand, considers the geometry of the existing BFS's. The weighted "influences"[3] then aggregately govern the choice of the next BFS. We believe that this is the main driving force for the speedup of our algorithms.

Scaling schemes

Many algorithms for combinatorial problems are explicitly or implicitly scaling a potential function or a parameter. In this chapter, our algorithms in some sense aim to minimize the volume of the feasible region. Scaling schemes for different potential functions and parameters were also designed in previous works [121, 118, 125, 119]. All of these functions and parameters have an explict form. On the contrary, our potential function is somewhat unusual in the sense that it has no closed form.

[3] In the terminology of Chapter 8, these weighted influences are the leverage scores.

Deducing new constraints

As mentioned in the main text, our algorithms share the same skeleton and tools for deducing new constraints with [121, 118, 125, 119]. Nevertheless, there are differences in the way these tools are employed. Our algorithms proceed by invoking them in a geometric manner, whereas previous algorithms were mostly combinatorial.

Big elements and bucketing

Our bucketing idea has roots in Iwata-Orlin's algorithm [125] but is much more sophisticated. For instance, it is sufficient for their algorithm to consider only big elements, i.e. $\texttt{upper}(i) \geq N/n^{O(1)}$. Our algorithm, on the other hand, must carefully group elements by the size of both $\texttt{upper}(i)$ and $\texttt{lower}(i)$. The speedup appears impossible without these new ideas. We do however note that it is unfair to expect such a sophisticated scheme in Iwata-Orlin's algorithm as it would not lead to a speedup. In other words, their method is fully sufficient for their purposes, and the simplicity in their case is a virtue rather than a shortcoming.

10.5.1 Open Problems

One natural open problem is improving our weakly polynomial algorithm to $O(n^2 \log M \cdot \mathrm{EO} + n^3 \log^{O(1)} n \cdot \log M)$ time. Our application of center of mass to SFM demonstrates that it should be possible.

For strongly polynomial algorithms, the existential result of Theorem 10.4.1 shows that SFM can be solved with $O(n^3 \log n \cdot \mathrm{EO})$ oracle calls. Unfortunately, our algorithm incurs an overhead of $\log n$ as there can be as many as $\log n$ buckets each time. One may try to remove this $\log n$ overhead by designing a better bucketing scheme or arguing that more arcs can be added.

The other $\log n$ overhead seem much trickier to remove. Our method currently makes crucial use of the tools developed by [121], where the $\log n$ factors in the runtime seem inevitable. We suspect that our algorithm may have an analogue similar to [231, 216], which do not carry any $\log n$ overhead in the running time.

Perhaps an even more interesting open problem is whether our algorithm is optimal (up to polylogarithmic factors). There are grounds for optimism. So far the best way of *certifying* the optimality of a given solution $S \subseteq V$ is to employ duality and express some optimal solution to the base polyhedron as a convex combination of $n+1$ BFS's. This already takes n^2 oracle calls as each BFS requires n. Thus one would expect the optimal number of oracle calls needed for SFM to be at least n^2. Our bound is not too far off from it, and anything strictly between n^2 and n^3 seems instinctively unnatural.

Chapter 11

First Order Method vs Cutting Plane Method

11.1 Introduction

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a β-smooth and α-strongly convex function. Thus, for any $x, y \in \mathbb{R}^n$, we have

$$f(x) + \nabla f(x)^\top (y - x) + \frac{\alpha}{2}|y - x|^2 \leq f(y) \leq f(x) + \nabla f(x)^\top (y - x) + \frac{\beta}{2}|y - x|^2.$$

Let $\kappa = \beta/\alpha$ be its condition number. It is a one line calculation to verify that a step of gradient descent on f will decrease (multiplicatively) the squared distance to the optimum by $1 - 1/\kappa$.

In Section 11.3, we propose a new method, which can be viewed as some combination of gradient descent and the ellipsoid method, for which the squared distance to the optimum decreases at a rate of $(1 - 1/\sqrt{\kappa})$ (and each iteration requires one gradient evaluation and two line-searches). This matches the optimal rate of convergence among the class of first-order methods, [203, 206].

In Section 11.4, we generalize this idea and show how to automatically combine different optimization algorithms. In particular, we show how to combine gradient descent and cutting plane methods to get an algorithm with convergence rate $\min(1 - 1/\kappa, 1 - 1/O(n))$. This is the first algorithm we know that can naturally achieve both κ dependence and n dependence guarantee.

In Section 11.5, we provide some numerical evidence that the new methods can be superior to Nesterov's accelerated gradient descent.

11.1.1 Preliminaries

We write $|\cdot|$ for the Euclidean norm in $\mathbb{R}^n$, and $\mathrm{B}(x, r^2) := \{y \in \mathbb{R}^n : |y - x|^2 \leq r^2\}$ (note that the second argument is the radius squared). We define the map $\texttt{line\_search} : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$ by

$$\texttt{line\_search}(x, y) = \arg\min_{t \in \mathbb{R}} f(x + t(y - x)),$$

and we denote

$$x^+ = x - \frac{1}{\beta}\nabla f(x), \text{ and } x^{++} = x - \frac{1}{\alpha}\nabla f(x).$$

Recall that by strong convexity one has

$$\forall y \in \mathbb{R}^n, f(y) \geq f(x) + \nabla f(x)^\top (y - x) + \frac{\alpha}{2}|y - x|^2,$$

which implies in particular:

$$x^* \in \mathrm{B}\left(x^{++}, \frac{|\nabla f(x)|^2}{\alpha^2} - \frac{2}{\alpha}(f(x) - f(x^*))\right).$$

Furthermore recall that by smoothness one has $f(x^+) \leq f(x) - \frac{1}{2\beta}|\nabla f(x)|^2$ which allows to *shrink* the above ball by a factor of $1 - \frac{1}{\kappa}$ and obtain the following:

$$x^* \in \mathrm{B}\left(x^{++}, \frac{|\nabla f(x)|^2}{\alpha^2}\left(1 - \frac{1}{\kappa}\right) - \frac{2}{\alpha}(f(x^+) - f(x^*))\right) \tag{11.1}$$

11.2 Intuition

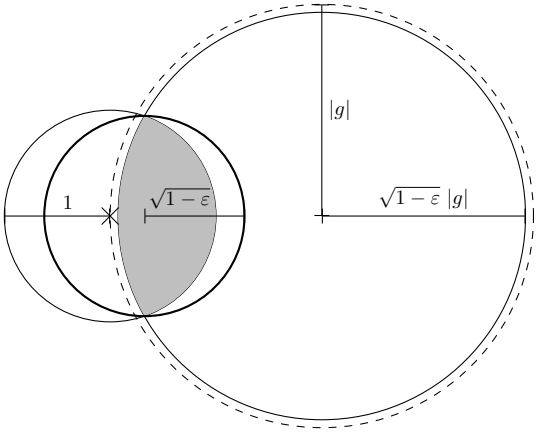


Figure 11-1: One ball shrinks.

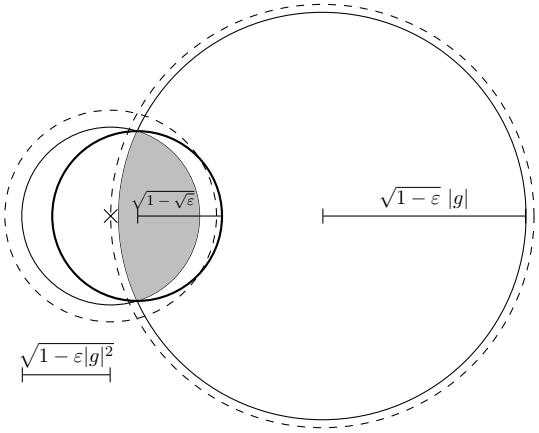


Figure 11-2: Two balls shrink.

The left diagram shows the intersection shrinks at the same rate if only one of the ball shrinks; the right diagram shows the intersection shrinks much faster if two balls shrinks at the same absolute amount.

In Section 11.2.1 we describe a geometric alternative to gradient descent (with the same convergence rate) which gives the core of our new optimal method. Then in Section 11.2.2 we explain why one can expect to accelerate this geometric algorithm.

11.2.1 A Suboptimal Algorithm

Assume that we are given a guarantee $R_0 > 0$ on the distance from some point x_0 to the optimum, that is $x^* \in \mathrm{B}(x_0, R_0^2)$. Combining this original enclosing ball for x^* and the one obtained by (11.1) (with $f(x^*) \leq f(x_0^+)$) one obtains

$$x^* \in \mathrm{B}(x_0, R_0^2) \cap \mathrm{B}\left(x_0 - \frac{1}{\alpha}\nabla f(x_0), \frac{|\nabla f(x_0)|^2}{\alpha^2}\left(1 - \frac{1}{\kappa}\right)\right).$$

If $\frac{|\nabla f(x_0)|^2}{\alpha^2} \leq R_0^2(1 - \frac{1}{\kappa})$ then the second ball already shrinks by a factor of $(1 - \frac{1}{\kappa})$. In the other case when $\frac{|\nabla f(x_0)|^2}{\alpha^2} > R_0^2(1 - \frac{1}{\kappa})$, the center of the two balls are far apart and therefore there is a much smaller ball containing the intersection of two balls. Formally, it is an easy calculation to see that for any $g \in \mathbb{R}^n$, $\varepsilon \in (0, 1)$, there exists $x \in \mathbb{R}^n$ such that

$$\mathrm{B}(0, 1) \cap \mathrm{B}(g, |g|^2(1 - \varepsilon)) \subset \mathrm{B}(x, 1 - \varepsilon). \qquad \text{(Figure 11-1)}$$

In particular the two above display implies that there exists $x_1 \in \mathbb{R}^n$ such that

$$x^* \in \mathrm{B}\left(x_1, R_0^2\left(1 - \frac{1}{\kappa}\right)\right).$$

Denote by T the map from x_0 to x_1 defined implicitely above, and let (x_k) be defined by $x_{k+1} = T(x_k)$. Then we just proved

$$|x^* - x_k|^2 \leq \left(1 - \frac{1}{\kappa}\right)^k R_0^2.$$

In other words, after $2\kappa \log(R_0/\varepsilon)$ iterations where each iteration cost one call to the gradient oracle) one obtains a point ε-close to the minimizer of f.

11.2.2 Why One Can Accelerate

Assume now that we are give a guarantee $R_0 > 0$ such that $x^* \in \mathrm{B}(x_0, R_0^2 - \frac{2}{\alpha}(f(y) - f(x^*)))$ where $f(x_0) \leq f(y)$ (say by choosing $y = x_0$). Using the fact that $f(x_0^+) \leq f(x_0) - \frac{1}{2\beta}|\nabla f(x_0)|^2 \leq f(y) - \frac{1}{2\alpha\kappa}|\nabla f(x_0)|^2$, we obtain that

$$x^* \in \mathrm{B}\left(x_0, R_0^2 - \frac{|\nabla f(x_0)|^2}{\alpha^2\kappa} - \frac{2}{\alpha}\left(f(x_0^+) - f(x^*)\right)\right)$$

which, intuitively, allows us the shrink the radius squared from R_0^2 to $R_0^2 - \frac{|\nabla f(x_0)|^2}{\alpha^2\kappa}$ using the local information at x_0. From (11.1), we have

$$x^* \in \mathrm{B}\left(x_0^{++}, \frac{|\nabla f(x_0)|^2}{\alpha^2}\left(1 - \frac{1}{\kappa}\right) - \frac{2}{\alpha}\left(f(x_0^+) - f(x^*)\right)\right).$$

Now, intersecting the above two shrunk balls and using Lemma 11.3.2 (see below and also see Figure 11-2), we obtain that there is an x_1' such that

$$x^* \in \mathrm{B}\left(x_1', R_0^2\left(1 - \frac{1}{\sqrt{\kappa}}\right) - \frac{2}{\alpha}\left(f(x_0^+) - f(x^*)\right)\right)$$

giving us an acceleration in shrinking of the radius. To carry the argument for the next iteration, we would have required that $f(x_1') \leq f(x_0^+)$ but it may not hold. Thus, we choose x_1 by a line search

$$x_1 = \texttt{line\_search}\left(x_1', x_0^+\right)$$

which ensures that $f(x_1) \leq f(x_0^+)$. To remedy the fact that the ball for the next iteration is centered at x_1' and not x_1, we observe that the line search also ensures that $\nabla f(x_1)$ is perpendicular to the line going through x_1 and x_1'. This geometric fact is enough for the algorithm to work at the next iteration as well. In the next section we describe precisely our proposed algorithm which is based on the above insights.

11.3 An Optimal Algorithm

Let $x_0 \in \mathbb{R}^n$, $c_0 = x_0^{++}$, and $R_0^2 = \left(1 - \frac{1}{\kappa}\right)\frac{|\nabla f(x_0)|^2}{\alpha^2}$. For any $k \geq 0$ let

$$x_{k+1} = \texttt{line\_search}\left(c_k, x_k^+\right),$$

and c_{k+1} (respectively R_{k+1}^2) be the center (respectively the squared radius) of the ball given by (the proof of) Lemma 11.3.2 which contains

$$\mathrm{B}\left(c_k, R_k^2 - \frac{|\nabla f(x_{k+1})|^2}{\alpha^2\kappa}\right) \cap \mathrm{B}\left(x_{k+1}^{++}, \frac{|\nabla f(x_{k+1})|^2}{\alpha^2}\left(1 - \frac{1}{\kappa}\right)\right).$$

Algorithm 27: Minimum Enclosing Ball of the Intersection to Two Balls

Input: a ball centered at x_A with radius R_A and a ball centered at x_B with radius R_B.
if $|x_A - x_B|^2 \geq \left|R_A^2 - R_B^2\right|$ **then**
  $c = \frac{1}{2}(x_A + x_B) - \frac{R_A^2 - R_B^2}{2|x_A - x_B|^2}(x_A - X_B)$.
  $R^2 = R_B^2 - \frac{\left(|x_A - x_B|^2 + R_B^2 - R_A^2\right)^2}{4|x_A - x_B|^2}$.
else if $|x_A - x_B|^2 < R_A^2 - R_B^2$ **then**
  $c = x_B$. $R = R_B$.
else
  $c = x_A$. $R = R_A$.
end
Output: a ball centered at c with radius R.

Algorithm 28: Geometric Descent Method (GeoD)

Input: parameters α and initial points x_0.
$x_0^+ = \texttt{line\_search}(x_0, x_0 - \nabla f(x_0))$.
$c_0 = x_0 - \alpha^{-1}\nabla f(x_0)$.
$R_0^2 = \frac{|\nabla f(x_0)|^2}{\alpha^2} - \frac{2}{\alpha}\left(f(x_0) - f(x_0^+)\right)$.
for $i \leftarrow 1, 2, \cdots$ **do**
  Combining Step:
  $x_k = \texttt{line\_search}(x_{k-1}^+, c_{k-1})$.
  Gradient Step:
  $x_k^+ = \texttt{line\_search}(x_k, x_k - \nabla f(x_k))$.
  Ellipsoid Step:
  $x_A = x_k - \alpha^{-1}\nabla f(x_k)$. $R_A^2 = \frac{|\nabla f(x_k)|^2}{\alpha^2} - \frac{2}{\alpha}\left(f(x_k) - f(x_k^+)\right)$.
  $x_B = c_{k-1}$. $R_B^2 = R_{k-1}^2 - \frac{2}{\alpha}\left(f(x_{k-1}^+) - f(x_k^+)\right)$.
  Let $B(c_k, R_k^2)$ is the minimum enclosing ball of $B(x_A, R_A^2) \cap B(x_B, R_B^2)$.
end
Output: x_T.

The formulas for c_{k+1} and R_{k+1}^2 are given in Algorithm 27.

Theorem 11.3.1. *For any $k \geq 0$, one has $x^* \in \mathrm{B}(c_k, R_k^2)$, $R_{k+1}^2 \leq \left(1 - \frac{1}{\sqrt{\kappa}}\right) R_k^2$, and thus*

$$|x^* - c_k|^2 \leq \left(1 - \frac{1}{\sqrt{\kappa}}\right)^k R_0^2.$$

Proof. We will prove a stronger claim by induction that for each $k \geq 0$, one has

$$x^* \in \mathrm{B}\left(c_k, R_k^2 - \frac{2}{\alpha}\left(f(x_k^+) - f(x^*)\right)\right).$$

The case $k = 0$ follows immediately by (11.1). Let us assume that the above display is true for some $k \geq 0$. Then using $f(x^*) \leq f(x_{k+1}^+) \leq f(x_{k+1}) - \frac{1}{2\beta}|\nabla f(x_{k+1})|^2 \leq f(x_k^+) - \frac{1}{2\beta}|\nabla f(x_{k+1})|^2$, one gets

$$x^* \in \mathrm{B}\left(c_k, R_k^2 - \frac{|\nabla f(x_{k+1})|^2}{\alpha^2\kappa} - \frac{2}{\alpha}\left(f(x_{k+1}^+) - f(x^*)\right)\right).$$

Furthermore by (11.1) one also has

$$\mathrm{B}\left(x_{k+1}^{++}, \frac{|\nabla f(x_{k+1})|^2}{\alpha^2}\left(1-\frac{1}{\kappa}\right) - \frac{2}{\alpha}\left(f(x_{k+1}^{+}) - f(x^*)\right)\right).$$

Thus it only remains to observe that the squared radius of the ball given by Lemma 11.3.2 which encloses the intersection of the two above balls is smaller than $\left(1-\frac{1}{\sqrt{\kappa}}\right)R_k^2 - \frac{2}{\alpha}(f(x_{k+1}^{+}) - f(x^*))$. We apply Lemma 11.3.2 after moving c_k to the origin and scaling distances by R_k. We set $\varepsilon = \frac{1}{\kappa}$, $g = \frac{|\nabla f(x_{k+1})|}{\alpha}$, $\delta = \frac{2}{\alpha}\left(f(x_{k+1}^{+}) - f(x^*)\right)$ and $a = x_{k+1}^{++} - c_k$. The line search step of the algorithm implies that $\nabla f(x_{k+1})^\top (x_{k+1} - c_k) = 0$ and therefore, $|a| = |x_{k+1}^{++} - c_k| \geq |\nabla f(x_{k+1})|/\alpha = g$ and Lemma 11.3.2 applies to give the result. □

Lemma 11.3.2. *Let $a \in \mathbb{R}^n$ and $\varepsilon \in (0,1), g \in \mathbb{R}_+$. Assume that $|a| \geq g$. Then there exists $c \in \mathbb{R}^n$ such that for any $\delta \geq 0$,*

$$\mathrm{B}(0, 1-\varepsilon g^2 - \delta) \cap \mathrm{B}(a, g^2(1-\varepsilon) - \delta) \subset \mathrm{B}\left(c, 1 - \sqrt{\varepsilon} - \delta\right).$$

Proof. First observe that if $g^2 \leq 1/2$ then one can take $c = a$ since $\frac{1}{2}(1-\varepsilon) \leq 1 - \sqrt{\varepsilon}$. Thus we assume now that $g^2 > 1/2$, and note that we can also clearly assume that $n = 2$. Consider the segment joining the two points at the intersection of the two balls under consideration. We denote c for the point at the intersection of this segment and $[0, a]$, and $x = |c|$ (that is $c = x\frac{a}{|a|}$). A simple picture reveals that x satisfies

$$1 - \varepsilon g^2 - \delta - x^2 = g^2(1-\varepsilon) - \delta - (|a| - x)^2 \quad \Leftrightarrow \quad x = \frac{1 + |a|^2 - g^2}{2|a|}.$$

When $x \leq |a|$, neither of the balls covers more than half of the other ball and hence the intersection of the two balls is contained in the ball $\mathrm{B}\left(x\frac{a}{|a|}, 1 - \varepsilon g^2 - \delta - x^2\right)$ (See figure 11-2). Thus it only remains to show that $x \leq |a|$ and that $1 - \varepsilon g^2 - \delta - x^2 \leq 1 - \sqrt{\varepsilon} - \delta$. The first inequality is equivalent to $|a|^2 + g^2 \geq 1$ which follows from $|a|^2 \geq g^2 \geq 1/2$. The second inequality to prove can be written as

$$\varepsilon g^2 + \frac{(1 + |a|^2 - g^2)^2}{4|a|^2} \geq \sqrt{\varepsilon},$$

which is straightforward to verify (recall that $|a|^2 \geq g^2 \geq 1/2$). □

Algorithm 28 we give is more agressive than Theorem 11.3.1, for instance, using line search instead of fixed step size. The correctness of this version follows from a similar proof as Theorem 11.3.1.

This algorithm does not require the smoothness parameter and the number of iterations; and it guarantees the function value is strictly decreasing. They are useful properties for machine learning applications because the only required parameter α is usually given. Furthermore, we believe that the integration of zeroth and first order information about the function makes our new method particularly well-suited in practice.

■ 11.4 Politician

In this section, we discuss how to in general combine different optimization algorithms. In standard black-box convex optimization [201, 206, 40] first-order methods interact with an *oracle*: given a query point x, the oracle reports the value and gradient of the underlying objective function f at x. Here, we propose to replace the oracle by a *politician*. Instead of answering the original query x the politician changes the question and answers a new query y which is guaranteed to be better than the original

query x in the sense that $f(y) \leq f(x)$. The newly selected query y also depends on the history of queries that were made to the politician. Formally we introduce the following definition (for sake of simplicty we write $\nabla f(x)$ for either a gradient or a subgradient of f at x).

Definition 11.4.1. Let $\mathcal{X} \subset \mathbb{R}^n$ and $f : \mathcal{X} \to \mathbb{R}$. A politician Φ for f is a mapping from $\mathcal{X} \times \cup_{k=0}^{\infty}(\mathcal{X} \times \mathbb{R} \times \mathbb{R}^n)^k$ to $\mathcal{X}$ such that for any $k \geq 0, x \in \mathcal{X}, h \in (\mathcal{X} \times \mathbb{R} \times \mathbb{R}^n)^k$ one has $f(\Phi(x,h)) \leq f(x)$. Furthermore when queried at x with history h a politician for f also output $f(\Phi(x,h))$ and $\nabla f(\Phi(x,h))$ (in order to not overload notation we do not include these outputs in the range of Φ).

Let us clarify the interaction of a first-order method with a politician. Note that we refer to the couple (first-order method, politician) as the *algorithm*. Let $M : \cup_{k=0}^{\infty}(\mathcal{X} \times \mathbb{R} \times \mathbb{R}^n)^k \to \mathcal{X}$ be a first-order method and Φ a politician for some function $f : \mathcal{X} \to \mathbb{R}$. The course of the algorithm (M, Φ) then goes as follows: at iteration $k+1$ one first calculates the method's query point $x_{k+1} = M(h_k)$ (with $h_0 = \emptyset$), then one calculates the politician's new query point $y_{k+1} = \Phi(x_{k+1}, h_k)$ and the first order information at this point $(f(y_{k+1}), \nabla f(y_{k+1}))$, and finally one updates the history with this new information $h_{k+1} = (h_k, (y_{k+1}, f(y_{k+1}), \nabla f(y_{k+1})))$. Note that a standard oracle simply corresponds to a politician $\mathcal{O}$ for f such that $\mathcal{O}(x,h) = (x, f(x), \nabla f(x))$ (in particular the algorithm $(M, \mathcal{O})$ is the usual algorithm corresponding to the first-order method M).

The philosophy of the above definition is that it gives in some sense an automatic way to combine different optimization algorithms. Say for example that we wish to combine the ellipsoid method with gradient descent. One way to do so is to design an "ellipsoidal politician": the politician keeps track of a feasible ellipsoidal region based on the previously computed gradients, and when asked with the query x the politician chooses as a new query y the result of a line-search on the line between x and the center of current ellipsoid. Gradient descent with this ellipsoidal politician would then replace the step $x \leftarrow x - \eta \nabla f(x)$ by $x \leftarrow y - \eta \nabla f(y)$. The hope is that in practice such a combination would integrate the fast incremental progress of gradient descent with the geometrical progress of the ellipsoid method.

In this section, we focus on unconstrained convex optimization. We are particularly interested in situations where calculating a (sub)gradient has superlinear complexity (i.e., $\gg n$) such as in logistic regression and semidefinite programming. In such cases it is natural to try to make the most out of the computed gradients by incorporating geometric reasoning (such as in the ellipsoid method). We do so by introducing the *geometric politician* (Section 11.4.2), which is based on a combination of the ideas in previous section with standard cutting plane machinery (through the notion of a "center" of a set, see Section 11.4.3). For a given first order method M, we denote by $M+$ the algorithm obtained by running M with the geometric politician. We demonstrate empirically (Section 11.5) the effectiveness of the geometric politician on various standard first-order methods for convex optimization (gradient descent, Nesterov's accelerated gradient descent, non-linear conjugate gradient, BFGS). In particular we show that BFGS+ is a surprisingly robust and parameter-free algorithm with state of the art performance across a wide range of problems (both smooth and non-smooth).

11.4.1 Affine Invariant Politician

As mentioned above we assume that the complexity of computing the map $x \mapsto \nabla f(x)$ is superlinear. This implies that we can afford to have a politician such that the complexity of computing the map $(x,h) \mapsto \Phi(x,h)$ is $O(n \times \text{poly}(k))$ (we think of the number of iterations k as typically much smaller than the dimension n). We show in this section that this condition is (essentially) automatically satisfied as long as the politician is affine invariant in the following sense (we use a slight abuse of language and refer to a map $f \mapsto \Phi_f$, where Φ_f is a politician for f, as a politician):

Definition 11.4.2. A politician $f \mapsto \Phi_f$ is called affine invariant if for any function f and any affine map $T : \mathbb{R}^m \to \mathbb{R}^n$ such that $T(x) = z + Lx$ for some matrix L, $k \geq 0, x \in \mathbb{R}^m, (y_i, v_i, g_i) \in \mathbb{R}^m \times \mathbb{R} \times \mathbb{R}^n$, one has

$$T(\Phi_{f\circ T}(x, (y_i, v_i, L^\top g_i)_{i\in[k]})) = \Phi_f(T(x), (T(y_i), v_i, g_i)_{i\in[k]}).$$

We say that an affine invariant politician has cost $\psi : \mathbb{N} \to \mathbb{N}$ if for any $f : \mathbb{R}^k \to \mathbb{R}$ the map $(x, h) \in \mathbb{R}^k \times (\mathbb{R}^k \times \mathbb{R} \times \mathbb{R}^k)^k \mapsto \Phi_f(x, h)$ can be computed in time $\psi(k)$.

Proposition 11.4.3. *Let Φ be an affine invariant politician with cost ψ. Then for any $f : \mathbb{R}^n \to \mathbb{R}$, $(y_i, v_i, g_i) \in \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}^n, i \in [k]$ and $x, y_i \in y_1 + \mathrm{Span}(g_1, \cdots, g_k)$ one can compute $\Phi_f(x, (y_i, v_i, g_i)_{i\in[k]}) \in \mathbb{R}^n$ in time $\psi(k) + O(nk^2)$.*

Proof. Let G be the $n \times k$ matrix with i^{th} column given by g_i. We consider the QR decomposition of G which can be computed in time $O(nk^2)$, that is Q is an $n \times k$ matrix and R a $k \times k$ matrix such that $G = QR$ and $Q^\top Q = \mathrm{I}_k$. Let T be the affine map defined by $T = y_1 + Q$. Note that since $x \in y_1 + \mathrm{Span}(g_1, \cdots, g_k)$ one has $x = T(Q^\top(x - y_1))$ (and similarly for y_i). Thus by affine invariance one has

$$\begin{aligned}\Phi_f(x, (y_i, v_i, g_i)) &= \Phi_f(T(Q^\top(x - y_1)), (T(Q^\top(y_i - y_1)), v_i, g_i)) \\ &= y_1 + Q\Phi_{f\circ T}(Q^\top(x - y_1), (Q^\top(y_i - y_1), v_i, R_i)),\end{aligned}$$

where R_i is the i^{th} column of R. Furthermore by definition of the cost ψ and since $f \circ T$ is defined on $\mathbb{R}^k$ we see that this last quantity can be computed in time $\psi(k) + O(nk^2)$, thus concluding the proof. □

The above proposition shows that with an affine invariant politician and a first order method M verifying for any $(y_i, v_i, g_i)_{i\in[k]} \in (\mathbb{R}^n \times \mathbb{R} \times \mathbb{R}^n)^k$,

$$M((y_i, v_i, g_i)_{i\in[k]}) \in y_1 + \mathrm{Span}(y_1, \cdots, y_k, g_1, \cdots, g_k),$$

one can run k steps of the corresponding algorithm in time $O(nk^2 + k\psi(k))$ plus the time to compute the k function values and gradients of the underlying function f to be optimized. Note that one gets a time of $O(nk^2)$ instead of $O(nk^3)$ as one can store the QR decomposition from one step to the next, and updating the decomposition only cost $O(nk)$.

11.4.2 Geometric Politician

We describe in this section the *geometric politician* which is based on ideas developed in Section 11.3. A key observation in that Section is that if f is a α-strongly convex function minimized at x^* then one has for any x,

$$\left\|x^* - x - \frac{1}{\alpha}\nabla f(x)\right\|^2 \leq \frac{\|\nabla f(x)\|^2}{\alpha^2} - \frac{2}{\alpha}\left(f(x) - f(x^*)\right).$$

This motivates the following definition:

$$\mathrm{B}(x, \alpha, \mathrm{fval}) := \left\{z \in \mathbb{R}^n : \left\|z - x - \frac{1}{\alpha}\nabla f(x)\right\|^2 \leq \frac{\|\nabla f(x)\|^2}{\alpha^2} - \frac{2}{\alpha}\left(f(x) - \mathrm{fval}\right)\right\}.$$

In particular given the first order information at $y_1, \cdots, y_k$ one knows that the optimum x^* lies in the region $R_k \subset \mathbb{R}^n$ defined by

$$R_k = \bigcap_{i\in[k]} B(y_i, \alpha, \mathrm{fval}) \text{ where } \mathrm{fval} = \min_{i\in[k]} f(y_i). \tag{11.2}$$

Algorithm 29: Geometric Politician

Parameter: An upper bound on the strong convexity parameter α. (Can be $+\infty$.)
Input: Query x, past queries and the corresponding first order information $(y_i, f(y_i), \nabla f(y_i))_{i\in[k]}$.
Let $\text{fval} = \min_{i\in[k]} f(y_i)$ and the feasible region $R_k(\alpha) = \bigcap_{i\in[k]} \mathrm{B}(y_i, \alpha, \text{fval})$.
if $R_k(\alpha) = \emptyset$ **then**
  Let α be the largest number such that $R_k(\alpha) \neq \emptyset$.
  $\alpha \leftarrow \alpha/4$.
end
Let $y_{k+1} = \arg\min_{y\in\{tx+(1-t)c(R_k(\alpha)),t\in\mathbb{R}\}} f(y)$
where $c(R_k(\alpha))$ is the volumetric center of $R_k(\alpha)$ (see Section 11.4.3).
Output: y_{k+1}, $f(y_{k+1})$ and $\nabla f(y_{k+1})$.

Now suppose that given this first order information at $y_1, \cdots, y_k$ the first order method asks to query x. How should we modify this query in order to take into account the geometric information that $x^* \in R_k$? First observe that for any z, $\mathrm{B}(z, \alpha, \text{fval})$ is contained in a halfspace that has z on its boundary (in the limiting case $\alpha \to 0$ the set $\mathrm{B}(z, \alpha, f(z))$ is exactly a halfspace). In particular if the next query point y_{k+1} is the center of gravity of R_k then we have that the volume of R_{k+1} is at most $1 - 1/e$ times the volume of R_k (see [111]), thus leading to an exponential convergence rate. However the region R_k can be very large initially, and the center of gravity might have a large function value and gradient, which means that R_k would be intersected with a large sphere (possibly so large that it is close to a halfspace). On the other hand the first order method recommends to query x, which we can think of as a local improvement of y_k, which should lead to a much smaller sphere. The issue is that the position of this sphere might be such that the intersection with R_k is almost as large as the sphere itself. In order to balance between the geometric and function value/gradient considerations we propose for the new query to do a line search between the center of R_k and the recommended query x. The geometric politician follows this recipe with two important modifications: (i) there are many choices of centers that would guarantee an exponential convergence rate while being much easier to compute than the center of gravity, and we choose here to consider the *volumetric center*, see Section 11.4.3 for the definition and more details about this notion; (ii) we use a simple heuristic to adapt online the strong convexity parameter α, namely we start with some large value for α and if it happens that the feasible region R_k is empty then we know that α was too large, in which case we reduce it. We can now describe formally the geometric politician, see Algorithm 29. Importantly one can verify that the geometric politician is affine invariant and thus can be implemented efficiently (see the proof of Proposition 11.4.3).

11.4.3 Volumetric Center

The volumetric barrier for a polytope was introduced in [253] to construct an algorithm with both the oracle complexity of the center of gravity method and the computational complexity of the ellipsoid method (see [Section 2.3, [40]] for more details and Chapter 8 for recent advances on this construction). Recalling that the standard logarithmic barrier F_P for the polytope $P = \{x \in \mathbb{R}^n : a_i^\top x < b_i, i \in [m]\}$ is defined by

$$F_P(x) = -\sum_{i=1}^{m} \log(b_i - a_i^\top x),$$

one defines the volumetric barrier v_P for P by

$$v_P(x) = \text{logdet}(\nabla^2 F_P(x)).$$

The volumetric center $c(P)$ is then defined as the minimizer of v_P. In the context of the geometric politician (see Algorithm 29) we are dealing with an intersection of balls rather than an intersection of halfspaces. More precisely the region of interest is of the form:

$$R = \bigcap_{i=1}^{k} \{x \in \mathbb{R}^n : \|x - c_i\| \leq r_i\}.$$

For such a domain the natural self-concordant barrier to consider is:

$$F_R(x) = -\frac{1}{2}\sum_{i=1}^{k} \log\left(r_i^2 - \|x - c_i\|^2\right).$$

The volumetric barrier is defined as before by

$$v_R(x) = \text{logdet}(\nabla^2 F_R(x)),$$

and the volumetric center of R is the minimizer of v_R. It is shown in [12] that v_R is a self-concordant barrier which means that the center can be updated (when a new ball is added to R) via few iterations of Newton's method. Often in practice, it takes less than 5 iterations to update the minimizer of a self-concordant barrier [103, 26] when we add a new constraint. Hence, the complexity merely depends on how fast we can compute the gradient and Hessian of F_R and v_R.

Proposition 11.4.4. *For the analytic barrier F_R, we have that*

$$\begin{aligned}\nabla F_R(x) =& A^\top 1_{k\times 1},\\ \nabla^2 F_R(x) =& 2A^\top A + \lambda^{(1)} I\end{aligned}$$

where d is a vector defined by $\left(r_i^2 - \|x - c_i\|^2\right)^{-1}$, A is a $k\times n$ matrix with i^{th} row given by $d_i(x)(x-c_i)$, $\lambda^{(p)} = \sum_{i\in[k]} d_i^p(x)$ and $1_{k\times 1}$ is a $k\times 1$ matrix with all entries being 1.

For the volumetric center, we have that

$$\begin{aligned}\nabla v_R(x) =& \left(\left(2\text{Tr}H^{-1}\right)\text{I} + 4H^{-1}\right)A^\top d + 8A^\top\sigma,\\ \nabla^2 v_R(x) =& 48A^\top\Sigma A - 64A^\top\left(AH^{-1}A^\top\right)^{(2)}A\\ &+\left(8\text{Tr}(D\Sigma) + 2\lambda^{(2)}\text{Tr}(H^{-1})\right)I + 4\lambda^{(2)}H^{-1}\\ &+8\text{Tr}(H^{-1})A^\top DA + 16\text{sym}(\left(A^\top DAH^{-1}\right))\\ &-4\text{Tr}(H^{-2})A^\top DJDA - 8H^{-1}A^\top DJDAH^{-1}\\ &-8\text{sym}(()A^\top DJDAH^{-2}) - 8\left(d^\top AH^{-1}A^\top d\right)H^{-1}\\ &-16\text{sym}(\left(A^\top \boldsymbol{Diag}\left(AH^{-2}A^\top\right)JDA\right))\\ &-32\text{sym}(\left(A^\top \boldsymbol{Diag}(AH^{-1}A^\top d)AH^{-1}\right))\end{aligned}$$

where $H = \nabla^2 F_R(x)$, $\sigma_i = e_i^\top AH^{-1}A^\top e_i$, e_i is the indicator vector with i^{th} coordinate, J is a $k\times k$ matrix with all entries being 1, $\text{sym}(()B) = B + B^\top$, $\boldsymbol{Diag}(v)$ is a diagonal matrix with $\boldsymbol{Diag}(v)_{ii} = v_i$, $D = \boldsymbol{Diag}(d)$, $\Sigma = \boldsymbol{Diag}(\sigma)$, and $B^{(2)}$ is the Schur square of B defined by $B^{(2)}_{ij} = B^2_{ij}$.

The above proposition shows that one step of Newton method for analytic center requires 1 dense matrix multiplication and solving 1 linear system; and for volumetric center, it requires 5 dense matrix multiplications, 1 matrix inversion and solving 1 linear system if implemented correctly. Although the analytic center is a more popular choice for "geometrical" algorithms, we choose volumetric center here because it gives a better convergence rate [253, 20] and the extra cost $\psi(k)$ is negligible to the cost of updating QR decomposition nk.

11.5 Experiments

In this section, we compare the geometric politician against two libraries for first order methods, minFunc [230] and TFOCS [32]. Both are popular MATLAB libraries for minimizing general smooth convex functions. Since the focus of this chapter is all about how to find a good step direction, we use the exact line search (up to machine accuracy) whenever possible. This eliminates the effect of different line searches and reduces the number of algorithms we need to test. TFOCS is the only algorithm we use which does not use line search because they do not provide such option. To compensate on the unfairness to TFOCS, we note that the algorithm TFOCS uses is accelerated gradient descent and hence we implement the Gonzaga-Karas's accelerated gradient descent [108], which is specifically designed to be used with exact line search. Another reason we pick this variant of accelerated gradient descent is because we found it to be the fastest variant of accelerated gradient descent (excluding the geometric descent in Section 11.3) for our tested data (Gonzaga and Karas also observed that on their own dataset).

The algorithms to be tested are the following:

- [SD] Steepest descent algorithm in minFunc.
- [Nes] Accelerated gradient descent, General Scheme 2.2.6 in [206].
- [TFOCS] Accelerated gradient descent in TFOCS.
- [GK] Gonzaga-Karas's of Accelerated Gradient Descent (Section 11.5.1).
- [Geo] Geometric Descent (Section 11.3).
- [CG] Non-Linear Conjugate Gradient in minFunc.
- [BFGS] Broyden–Fletcher–Goldfarb–Shanno algorithm in minFunc.
- [PCG] Preconditioned Non-Linear Conjugate Gradient in minFunc.
- [$\emptyset$+] Geometric Politician itself (Section 11.5.1).
- [GK+] Using GK with Geometric Oracle (Section 11.5.1).
- [BFGS+] Using BFGS with Geometric Oracle (Section 11.5.1).

We only tested the geometric oracle on GK and BFGS because they are respectively the best algorithms in theory and practice on our tested data. The $\emptyset$+ algorithm is used as the control group to test if the geometric politician by itself is sufficient to achieve good convergence rate. We note that all algorithms except Nes are parameter free; each step of SD, Nes, TFOCS, GK, Geo, CG takes $O(n)$ time and each step of BFGS, PCG, $\emptyset$+, GK+ and BFGS+ takes roughly $O(nk)$ time for k^{th} iteration.

Algorithm 30: $\emptyset+$

Input: x_0.
for $k \leftarrow 1, 2, \cdots$ **do**
  Set $x_{k+1} \leftarrow \Phi_f(x_k, (x_i, f(x_i), \nabla f(x_i))_{i\in[k]})$.
end

Algorithm 31: Gonzaga-Karas's variant of Accelerated Gradient Descent

Input: x_1.
$\gamma = 2\alpha$, $v_0 = x_0$ and $y_0 = x_0$.
for $k \leftarrow 1, 2, \cdots$ **do**
  $y_k \leftarrow \Phi_f(y_{k-1})$.
  $x_{k+1} = \texttt{line\_search}(y_k, -\nabla f(y_k))$.
  if $\alpha \geq \gamma/1.02$ *and we are using first order oracle* **then** $\alpha = \gamma/2$. (*);
  if $\alpha \geq \frac{\|\nabla f(y_k)\|^2}{2(f(y_k)-f(x_{k+1}))}$ **then** $\alpha = \frac{\|\nabla f(y_k)\|^2}{20(f(y_k)-f(x_{k+1}))}$.;
  $G = \gamma\left(\frac{\alpha}{2}\|v_k - y_k\|^2 + \langle \nabla f(y_k), v_k - y_k\rangle\right)$.
  $A = G + \frac{1}{2}\|\nabla f(y_k)\|^2 + (\alpha - \gamma)(f(x_k) - f(y_k))$.
  $B = (\alpha - \gamma)(f(x_{k+1}) - f(x_k)) - \gamma(f(y_k) - f(x_k)) - G$.
  $C = \gamma(f(x_{k+1}) - f(x_k))$.
  $\beta = \frac{-B+\sqrt{B^2-4AC}}{2A}$, $\gamma = (1-\beta)\gamma + \beta\alpha$.
  $v_{k+1} = \frac{1}{\gamma}((1-\beta)\gamma v_k + \beta(\alpha y_k - \nabla f(y_k))$.
end

11.5.1 Details of Implementations

The first algorithm we implement is the $\emptyset+$ algorithm which simply repeatedly call the politician. As we will see, this algorithm is great for non smooth problems but not competitive for smooth problems.

The second algorithm we implement is the accelerated gradient descent proposed by Gonzaga and Karas [108]. This algorithm uses line search to learn the the smoothness parameter and strong convexity parameter, see Algorithm 31. We disable the line (*) in the algorithm if Φ_f is a politician instead of an oracle because $\gamma \geq \alpha$ does not hold for the strong convexity parameter α if Φ_f is not an oracle.

The third algorithm we implemented is the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm. This algorithm uses the gradients to reconstruct the Hessian and use it to approximate Newton's method, see Algorithm 32. We note that another natural way to employ the politician with BFGS is to set $x_{k+1} = \texttt{line\_search}(\Phi_f(x_k), p)$ and this runs faster in practice; however, this algorithm computes two gradients per iteration (namely $\nabla f(x_k)$ and $\nabla f(\Phi_f(x_k))$) while we restrict ourselves to algorithms which compute one gradient per iteration.

11.5.2 Quadratic Function

We consider the function

$$f(x) = (x - c)^\top D(x - c), \tag{11.3}$$

where D is a diagonal matrix with entries uniformly sampled from $[0, 1]$ and c is a random vector with entries uniformly sampled from the normal distribution $N(0, 1)$. Since this is a quadratic function, CG, BFGS and BFGS+ are equivalent and optimal, namely, they output the minimum point in the

Algorithm 32: BFGS

Input: x_1.
for $k \leftarrow 1, 2, \cdots$ **do**
  $p = -\nabla f(x_k)$.
  for $i \leftarrow k-1, \cdots, 1$ **do**
    $\alpha_i \leftarrow \langle s_i, p\rangle / \langle s_i, y_i\rangle$.
    $p = p - \alpha_i y_i$.
  end
  $p = \langle s_{k-1}, y_{k-1}\rangle / \langle y_{k-1}, y_{k-1}\rangle\, p$.
  for $i \leftarrow 1, \cdots, k-1$ **do**
    $\beta_i \leftarrow \langle y_i, p\rangle / \langle s_i, y_i\rangle$.
    $p = p + (\alpha_i - \beta_i) y_i$.
  end
  $x_{k+1} = \Phi_f\left(\texttt{line\_search}(x_k, p)\right)$.
  $s_k = x_{k+1} - x_k,\ y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$.
end

span of all previous gradients.

11.5.3 Variant of Nesterov's Worst Function

[206] introduced the function

$$f(x) = (1 - x[1])^2 + \sum_{k=1}^{n-1} (x[k] - x[k+1])^2$$

and used it to give a lower bound for all first-order methods. To distinguish the performance between CG, BFGS and BFGS+, we consider the following non-quadratic variant

$$f(x) = g(1 - x[1]) + \sum_{k=1}^{n-1} g(x[k] - x[k+1]) \tag{11.4}$$

for some function g to be defined. If we pick $g(x) = |x|$ then all first order methods takes at least n iterations to minimize f exactly. On the other hand with $g(x) = \max(|x| - 0.1, 0)$ one of the minimizer of f is $(1, \frac{9}{10}, \frac{8}{10}, \cdots, \frac{1}{10}, 0, 0, \cdots, 0)$, and thus it takes at least 11 iterations for first order methods to minimize f in this case. We "regularize" the situation a bit and consider the function

$$g(x) = \begin{cases} \sqrt{(x-0.1)^2 + 0.001^2} - 0.001 & \text{if } x \geq 0.1 \\ \sqrt{(x+0.1)^2 + 0.001^2} - 0.001 & \text{if } x \leq -0.1 \\ 0 & \text{otherwise} \end{cases}.$$

Since this function is far from quadratic, our algorithms ($\emptyset$+, GK+, BFGS+) converge much faster. This is thus a nice example where the geometric politician helps a lot because the underlying dimension of the problem is small.

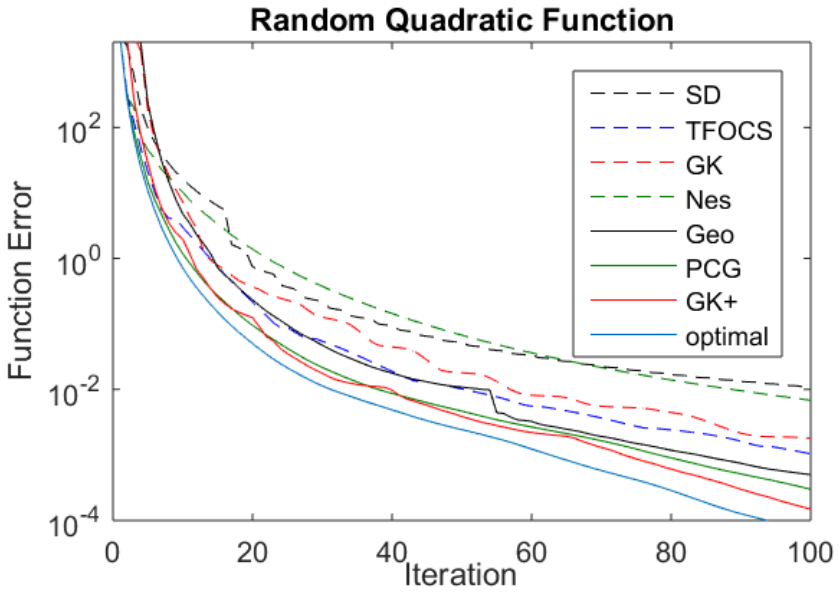


Figure 11-3: Comparison of first-order methods for the function (11.3) with $n = 10000$.

11.5.4 Binary Regression with Smoothed Hinge Loss

We consider the binary classification problem on the datasets from [48]. The problem is to minimize the regularized empirical risk:

$$f_t(x) = \frac{1}{n}\sum_{i=1}^{n} \varphi_t(b_i a_i^T x) + \frac{\lambda}{2}|x|^2 \tag{11.5}$$

where $a_i \in \mathbb{R}^d$, $b_i \in \mathbb{R}$ are given by the datasets, λ is the regularization coefficient, φ_t is the smoothed hinge loss defined by

$$\varphi_t(z) = \begin{cases} 0 & \text{if } z \le -1 \\ z + 1 - \frac{t}{2} & \text{if } z \ge -1 + t \\ \frac{1}{2t}(z+1)^2 & \text{otherwise} \end{cases}$$

and t is the smoothness parameter. The usual choice for t is 1, here we test both $t = 1$ and $t = 10^{-4}$. The latter case is to test how well the algorithms perform when the function is non-smooth.

We note that for this problem it would be natural to compare ourselves with SGD (stochastic gradient descent) or more refined stochastic algorithms such as SAG [159] or SVRG [129]. However since the focus of this chapter is on general black-box optimization we stick to comparing only to general methods. It is an interesting open problem to extend our algorithms to the stochastic setting, see Section 11.6.

In figures 11-5 and 11-6, we show the performance profile for problems in the LIBSVM datasets (and with different values for the regularization parameter λ). More precisely for a given algorithm we plot $x \in [1, 10]$ versus the fraction of datasets that the algorithm can solve (up to a certain prespecified accuracy) in a number of iterations which is at most x times the number of iterations of the best algorithm for this dataset. Figure 11-5 shows the case $t = 1$ with the targeted accuracy 10^{-6}; Figure 11-6 shows the case $t = 10^{-4}$ with the targeted accuracy 10^{-3}. We see that TFOCS is slower than SD for many problems, this is simply because SD uses the line search while TFOCS does not, and this makes a huge difference for simple problems. Among algorithms taking $O(n)$ time per iteration, CG and Geo perform the best, while for the $O(nk)$ algorithms we see that BFGS, BFGS+

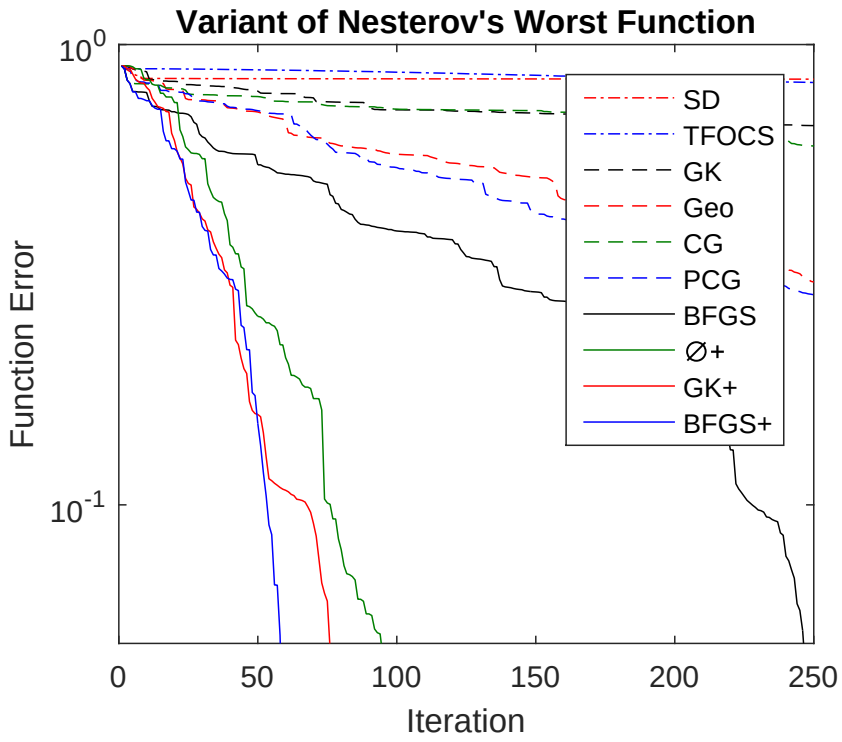


Figure 11-4: Comparison of first-order methods for the function (11.4) with $n = 10000$.

and GK+ perform the best. The gap in performance is particularly striking in the non-smooth case where BFGS+ is the fastest algorithm on almost all problems and all other methods (except GK+) are lagging far behind (for 20% of the problems all other methods take 10 times more iterations than BFGS+ and GK+).

Finally in figures 11-7 and 11-8 we test five algorithms on three specific datasets (respectively in the smooth and non-smooth case). In both figures we see that BFGS+ performs the best for all three datasets. BFGS performs second for smooth problems while GK+ performs second for nonsmooth problems.

11.5.5 Summary

The experiments show that BFGS+ and BFGS perform the best among all methods for smooth test problems while BFGS+ and GK+ perform the best for nonsmooth test problems. The first phenomenon is due to the optimality of these algorithm for quadratic problems. We leave the explanation for the second phenomenon as an open problem. At least, the experiments show that this is not due to the geometric oracle itself since $\emptyset+$ is much slower, and this is not due to the original algorithm since GK performs much worse than GK+ for those problems. Overall these experiments are very promising for the geometric oracle as a replacement of quasi Newton method for non-smooth problems and as a general purpose solver due to its robustness.

11.6 Discussion

First order methods generally involve only very basic operations at each step (addition, scalar multiplication). In last section, we formalize each step's operations (besides the gradient calculation) as

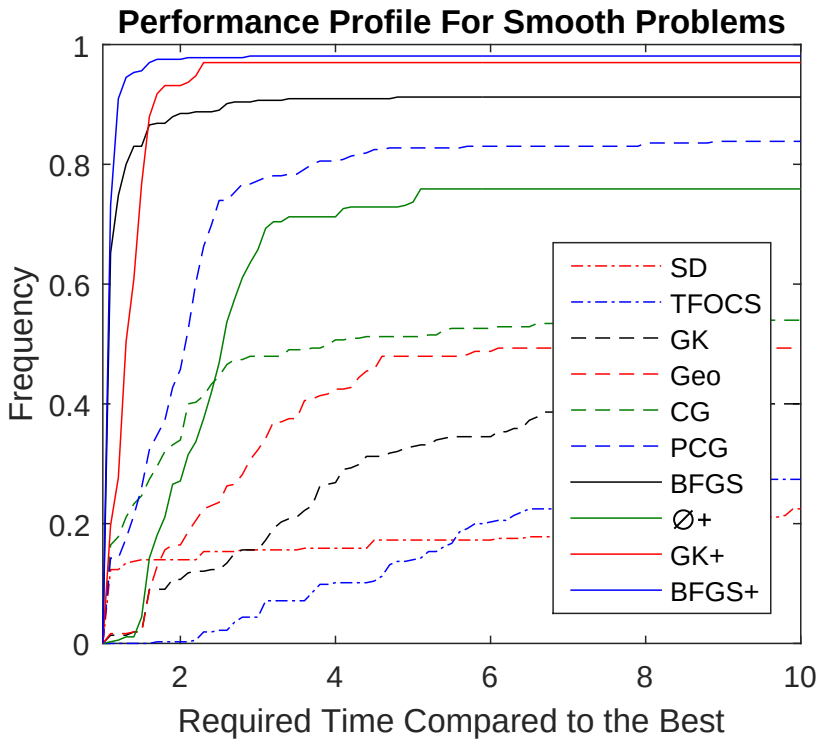


Figure 11-5: Performance profile on problem (11.5) with $t = 1$ and $\lambda = 10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}$.

the work of the politician. We showed that the cost per step of an affine invariant politician $\psi(k)$ is negligible compared to the gradient calculation (which is $\Omega(n)$). This opens up a lot of possibilities: instead of basic addition or scalar multiplication one can imagine computing a center of gravity, solving a linear program, or even searching over an exponential space (indeed, say $k < 30$ and $n > 10^{10}$, then $2^k < n$). Our experiments demonstrate the effectiveness of this strategy. On the other hand from a theoretical point of view a lot remains to be done. For example one can prove results of the following flavor:

Theorem 11.6.1. *Let f such that $\alpha \mathrm{I} \preceq \nabla^2 f(x) \preceq \beta \mathrm{I}, \forall x \in \mathbb{R}^n$ and let $\kappa = \beta/\alpha$. Suppose that in the Geometric Politician we replace the volumetric center by the center of gravity or the center of the John ellipsoid. Let y_k be the output of the k^{th} step of SD+ with some initial point x_0. Then, we have that*

$$f(y_k) - f(x^*) \leq \kappa \left(1 - \frac{1}{\Theta(\min(n \log(\kappa), \kappa))}\right)^k (f(x_0) - f(x^*)) .$$

and

$$f(y_k) - f(x^*) \leq \frac{2\beta R^2}{k+4}$$

where $R = \max_{f(x) \leq f(x_0)} \|x - x^*\|$.

This claim says that, up to a logarithmic factor, SD+ enjoys simultaneously the incremental progress of gradient descent and the geometrical progress of cutting plane methods. There are three caveats in this claim:

- We use the center of gravity or the center of the John ellipsoid instead of the volumetric center. Note however that it is well-known that the volumetric center is usually more difficult to analyze, [253, 20].

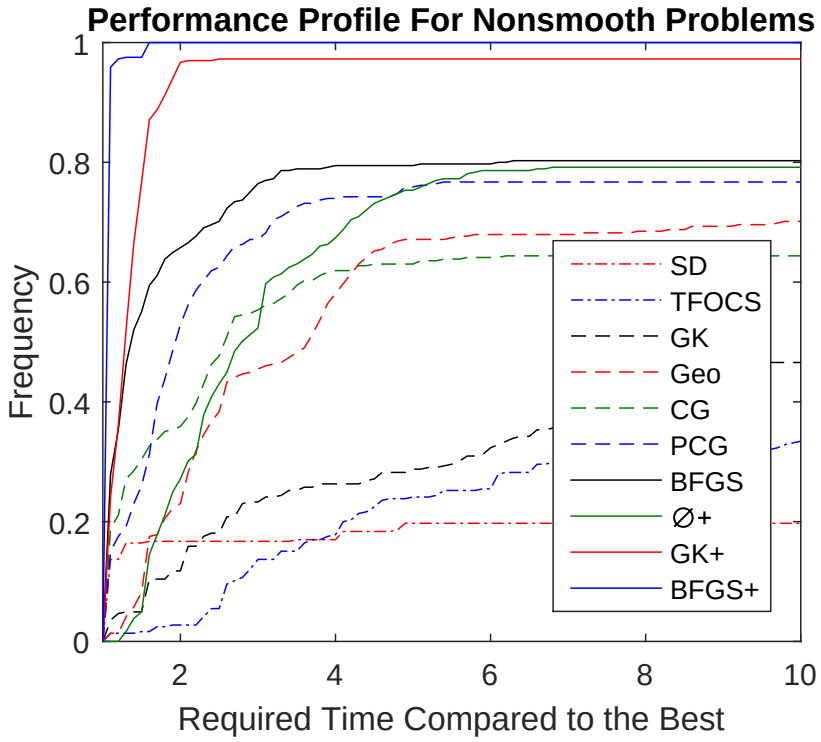


Figure 11-6: Performance profile on problem (11.5) with $t = 10^{-4}$ and $\lambda = 10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}$.

- The extraneous $\log(\kappa)$ comes from the number of potential restart when we decrease α. Is there a better way to learn α that would not incur this additional logarithmic term?

- Theorem 11.3.1 shows essentially that one can combine the ellipsoid method with gradient descent to achieve the optimal $1 - \sqrt{1/\kappa}$ rate. Can we prove such a result for SD+?

The geometric politician could be refined in many ways. Here are two simple questions that we leave for future work:

- One can think that gradient descent stores 1 gradient information, accelerated gradient descent stores 2 gradient information, and our method stores all past gradient information. We believe that neither 1, 2 nor all is the correct answer. Instead, the algorithm should dynamically decide the number of gradients to store based on the size of its memory, the cost of computing gradients, and the information each gradient reveals.

- Is there a stochastic version of our algorithm? How well would such a method compare with state of the art stochastic algorithms such as SAG [159] and SVRG [129]?

■ 11.7 Appendix: Convergence of SD+

Let f such that $\alpha \mathrm{I} \preceq \nabla^2 f(x) \preceq \beta \mathrm{I}$ for all $x \in \mathbb{R}^n$ and let $\kappa = \beta/\alpha$. Let y_k be the output of the k^{th} step of SD+ (where the volumetric center is replaced by the center of gravity or the center of the John ellipsoid) with some initial point x_0. We prove two rates of convergence for SD+, one with the condition number κ, and one with the ambient dimension n. We start by the former.

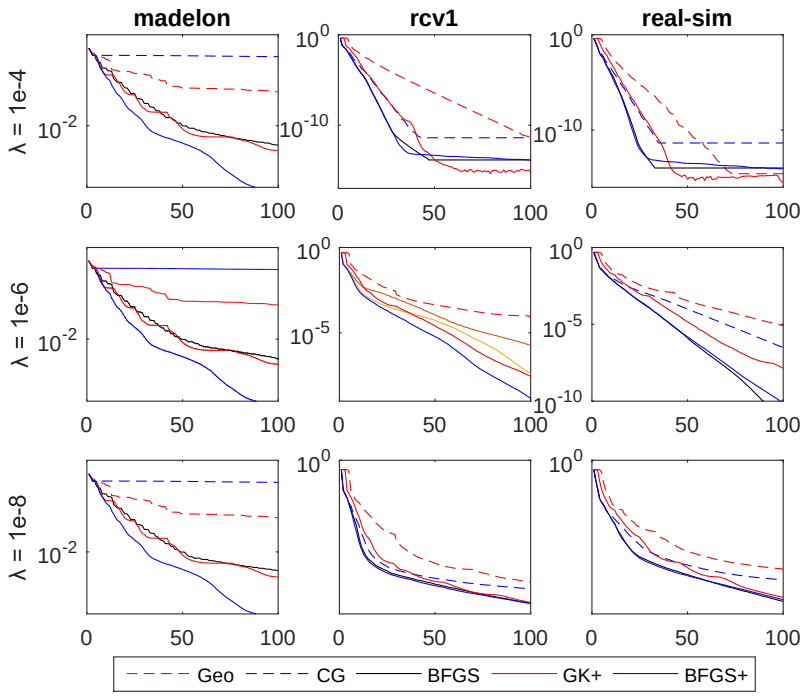


Figure 11-7: Comparison between Geo, CG, BFGS, GK+, BFGS+ on problem (11.5) with $t = 1$ and $\lambda = 10^{-4}, 10^{-6}, 10^{-8}$.

Theorem 11.7.1. *One has*

$$f(y_k) - f(x^*) \leq \left(1 - \frac{1}{\kappa}\right)^k (f(x_0) - f(x^*))$$

and

$$f(y_k) - f(x^*) \leq \frac{2\beta r^2}{k+4}$$

where $r = \max_{f(x)\leq f(x_0)} \|x - x^*\|$.

Proof. Let $\delta_k = f(y_k) - f(x^*)$. Since f is α-strongly convex we have that

$$\delta_k \leq \frac{1}{2\alpha}\|\nabla f(y_k)\|^2.$$

Due to the decrease guarantee of politicians and the line search in steepest descent, we have that $f(y_{k+1}) \leq f(x_{k+1}) \leq f(y_k) - \frac{1}{2\beta}\|\nabla f(y_k)\|^2$ and hence

$$\delta_k - \delta_{k+1} \geq \frac{1}{2\beta}\|\nabla f(y_k)\|^2 \geq \frac{\delta_k}{\kappa}. \tag{11.6}$$

Hence, we have $\delta_{k+1} \leq \left(1 - \frac{1}{\kappa}\right)\delta_k$ and this gives the first inequality.

To obtain a rate independent of α we instead use the following estimate

$$\delta_k \leq \langle \nabla f(y_k), y_k - x^* \rangle \leq \|\nabla f(y_k)\| \cdot \|y_k - x^*\|.$$

Using the decrease guarantee of politicians and line search we have that $f(y_k) \leq f(x_k) \leq f(y_{k-1}) \leq$

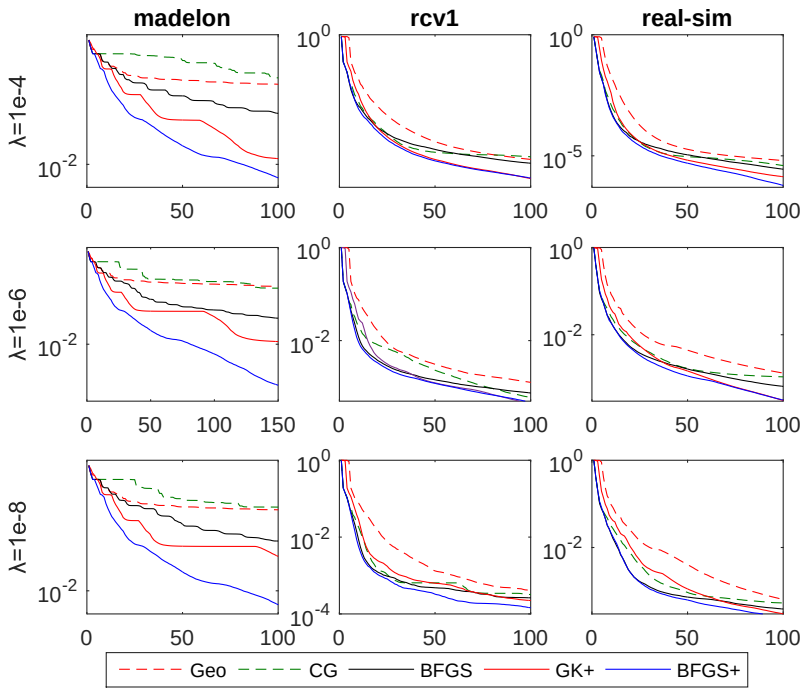


Figure 11-8: Comparison between Geo, CG, BFGS, GK+, BFGS+ on problem (11.5) with $t = 10^{-4}$ and $\lambda = 10^{-4}, 10^{-6}, 10^{-8}$.

$\cdots \leq f(x_0)$, and thus by definition of R:

$$\|y_k - x^*\| \leq r.$$

Due to the line search in steepest descent again, we have that

$$\delta_k - \delta_{k+1} \geq \frac{1}{2\beta}\|\nabla f(y_k)\|^2 \geq \frac{1}{2\beta}\left(\frac{\delta_k}{r}\right)^2.$$

Since $\delta_k \geq \delta_{k+1}$, we have

$$\frac{1}{\delta_{k+1}} - \frac{1}{\delta_k} = \frac{\delta_k - \delta_{k+1}}{\delta_k \delta_{k+1}} \geq \frac{\delta_k - \delta_{k+1}}{\delta_k^2} \geq \frac{1}{2\beta r^2}.$$

So, by induction, we have that $\frac{1}{\delta_k} \geq \frac{1}{\delta_0} + \frac{k}{2\beta r^2}$. Now, we note that

$$\delta_0 \leq \langle \nabla f(x^*), x_0 - x^* \rangle + \frac{\beta}{2}\|x_0 - x^*\|^2 \leq \frac{\beta r^2}{2}.$$

Thus, we have that

$$\delta_k \leq \frac{2\beta r^2}{k+4}.$$

□

We now turn to the dimension dependent analysis of SD+. We first show a simple geometric result, namely that if an intersection of spheres has a "small" volume then the intersection must lie close close to the boundary of one of the spheres.

Lemma 11.7.2. *Let $R = \cap_{i=1}^k \{x \in \mathbb{R}^n : \|x - c_i\| \leq r_i\}$, $D = \max_{i\in[k]} r_i$, and ω_n the volume of the unit ball in $\mathbb{R}^n$. Then, there exists $i \in [k]$ such that for all $x \in R$,*

$$\|x - c_i\|^2 \geq r_i^2 - 24k^2 \left(\frac{\mathrm{vol}R}{D^n \omega_n}\right)^{1/n} D^2.$$

Proof. Since $-\log(1-\|x\|^2)$ is a 1-self concordant barrier function, $2F_R$ is a k-self concordant function. Let y be the minimizer of F_R. Let $E = \{x \in \mathbb{R}^n : x^\top \nabla^2 (2F_R)(y) x \leq 1\}$. Theorem 4.2.6 in [206] shows that

$$y + E \subset R \subset y + (k + 2\sqrt{k})E. \tag{11.7}$$

In particular, this shows that $\mathrm{vol}E \leq \mathrm{vol}R$. We have that

$$\left(\det \nabla^2 F_R(y)\right)^{1/2} = \frac{\omega_n}{2^{n/2}} \frac{1}{\mathrm{vol}E} \geq \frac{\omega_n}{2^{n/2}} \frac{1}{\mathrm{vol}R}.$$

By the AM-GM inequality, we have that

$$\frac{\mathrm{Tr}\nabla^2 F_R(y)}{n} \geq \frac{1}{2} \left(\frac{\omega_n}{\mathrm{vol}R}\right)^{2/n}. \tag{11.8}$$

By Proposition 11.4.3, we have that $\nabla^2 F_R(y) = 2A^\top A + \lambda^{(1)} \mathrm{I}$ and hence,

$$\begin{aligned} \mathrm{Tr}\nabla^2 F_R(y) &= 2\mathrm{Tr}A^\top A + n\lambda^{(1)} \\ &= 2\sum_{i=1}^k \frac{\|y - c_i\|^2}{(r_i^2 - \|y - c_i\|^2)^2} + n \sum_{i=1}^k \frac{1}{r_i^2 - \|y - c_i\|^2}. \end{aligned}$$

Applying (11.8), we have that

$$\frac{2}{n} \sum_{i=1}^k \frac{\|y - c_i\|^2}{(r_i^2 - \|y - c_i\|^2)^2} + \sum_{i=1}^k \frac{1}{r_i^2 - \|y - c_i\|^2} \geq \frac{1}{2} \left(\frac{\omega_n}{\mathrm{vol}R}\right)^{2/n}$$

So, there exists i such that

$$\frac{\|y - c_i\|^2}{(r_i^2 - \|y - c_i\|^2)^2} \geq \frac{n}{8k} \left(\frac{\omega_n}{\mathrm{vol}R}\right)^{2/n} \text{ or } \frac{1}{r_i^2 - \|y - c_i\|^2} \geq \frac{1}{4k} \left(\frac{\omega_n}{\mathrm{vol}R}\right)^{2/n}.$$

Using $\|y - c_i\| \leq r_i \leq D$ and $\mathrm{vol}R \leq D^n \omega_n$, we have that

$$\begin{aligned} r_i^2 - \|y - c_i\|^2 &\leq \max\left(\sqrt{\frac{8k}{n}} \left(\frac{\mathrm{vol}R}{\omega_n}\right)^{1/n} \|y - c_i\|, 4k \left(\frac{\mathrm{vol}R}{\omega_n}\right)^{2/n}\right) \\ &\leq 4k \left(\frac{\mathrm{vol}R}{D^n \omega_n}\right)^{1/n} D^2. \end{aligned}$$

Therefore, the width of the ellipsoid E in the direction $y - c_i$ is at most

$$r_i - \|y - c_i\| \leq r_i - \sqrt{r_i^2 - 4k \left(\frac{\mathrm{vol}R}{D^n \omega_n}\right)^{1/n} D^2}.$$

The right hand side of (11.7) shows that, for all $x \in R$, we have

$$\begin{aligned}\|x - c_i\| &\geq r_i - (1+k+2\sqrt{k})\left(r_i - \sqrt{r_i^2 - 4k\left(\frac{\mathrm{vol}R}{D^n\omega_n}\right)^{1/n} D^2}\right)\\ &\geq r_i - (1+k+2\sqrt{k})\left(r_i - r_i\left(1 - 4k\left(\frac{\mathrm{vol}R}{D^n\omega_n}\right)^{1/n}\frac{D^2}{r_i^2}\right)\right)\\ &\geq \left[1 - 12k^2\left(\frac{\mathrm{vol}R}{D^n\omega_n}\right)^{1/n}\frac{D^2}{r_i^2}\right] r_i.\end{aligned}$$

Hence, we have

$$\|x - c_i\|^2 \geq \left[1 - 24k^2\left(\frac{\mathrm{vol}R}{D^n\omega_n}\right)^{1/n}\frac{D^2}{r_i^2}\right] r_i^2.$$

□

Finally, equipped with the above geometrical result, we can bound the convergence of SD+ using the dimension n. We start with a lemma taking care of the adaptivity to the strong convexity in the geometric politician.

Lemma 11.7.3. *In the first $k = \Theta(n\log(\frac{\kappa n}{\varepsilon}))$ iterations, either SD+ restarts the estimate of the strong convexity or*

$$f(y_k) - f(x^*) \leq \varepsilon\left(f(x_0) - f(x^*)\right).$$

Proof. The decrease guarantee and the smoothness imply that

$$\frac{\|\nabla f(y_k)\|^2}{2\beta} \leq f(y_k) - f(x^*) \leq f(x_0) - f(x^*).$$

Therefore, all the spheres found by the geometric politician have radius squared at most D^2 where, denoting $\overline{\alpha}$ for the convexity upper bound the algorithm is currently using,

$$D^2 = \max_{k\geq 1}\frac{\|\nabla f(y_k)\|^2}{\overline{\alpha}^2} \leq \frac{2\beta(f(x_0) - f(x^*))}{\overline{\alpha}^2}.$$

Lemma 11.7.2 shows that for any step k, there is $i \in [k]$ such that for all $x \in R_k$,

$$\|x - c_i\|^2 \geq r_i^2 - \frac{48\beta k^2}{\overline{\alpha}^2}\left(\frac{\mathrm{vol}R_k}{D^n\omega_n}\right)^{1/n}(f(x_0) - f(x^*)).$$

Let $k = \Theta(n\log(\frac{\kappa n}{\varepsilon}))$ and recall the discussion in Section 11.4.2 about the volume decrease of the geometric politician with the center of gravity (the same discussion applies to the John ellipsoid). We see that if the algorithm does not restart $\overline{\alpha}$ within the first k iterations then we have

$$\frac{\mathrm{vol}R_k}{D^n\omega_n} = \left(O\left(\frac{\varepsilon}{\kappa^2 k^2}\right)\right)^n,$$

and hence (for an appropriate numerical constant in k)

$$\|x - c_i\|^2 \geq r_i^2 - \frac{\varepsilon(f(x_0) - f(x^*))}{\overline{\alpha}\kappa}. \tag{11.9}$$

Recall from (11.6) that

$$f(y_{k+1}) \leq f(y_k) - \frac{f(y_k) - f(x^*)}{\kappa},$$

and therefore we have (by the improvement of the previous balls):

$$R_{k+1} \subset \left\{ \|x - c_i\|^2 \leq r_i^2 - \frac{2(f(y_k) - f(x^*))}{\overline{\alpha}\kappa} \right\} \cap R_k.$$

However, from (11.9), we know that either the above intersection is empty or $f(y_k) - f(x^*) < \varepsilon(f(x_0) - f(x^*))$. This proves the statement. □

Theorem 11.7.4. *We have that*

$$f(y_k) - f(x^*) \leq \kappa \left(1 - \frac{1}{\Theta(n \log(\kappa))}\right)^k (f(x_0) - f(x^*)).$$

Proof. If $\kappa < n$, the statements follows from Theorem 11.7.1. Hence, we can assume $\kappa \geq n$.

Set $T = \Theta(n \log(\frac{n\kappa}{\varepsilon}) \log(\kappa))$, Lemma 11.7.3 shows that for every $\Theta(n \log(\frac{n\kappa}{\varepsilon}))$ iteration, the algorithm either finds y such that

$$f(y) - f(x^*) \leq \varepsilon (f(x_0) - f(x^*))$$

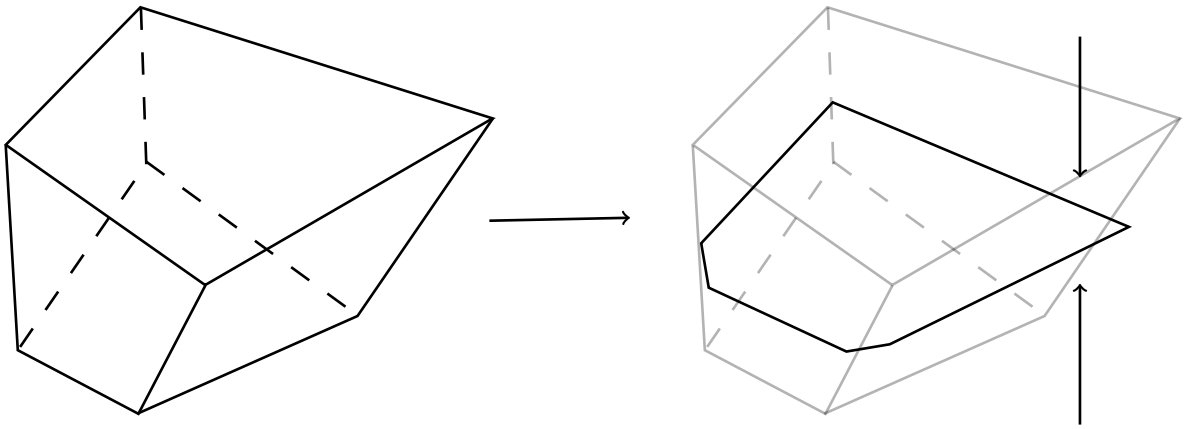
or decreases α_k by a constant where α_k is the convexity upper bound the algorithm is using at k^{th} iteration. Note that $\alpha_1 \leq \beta$ because of the line search, and thus the algorithm can restart α_k at most $\log(\kappa)$ many times. Hence, after T iterations, we must have

$$f(y_T) - f(x^*) \leq \varepsilon (f(x_0) - f(x^*)),$$

thus concluding the proof. □

Part III

Collapsing



Chapter 12

Undirected MaxFlow In Almost-Linear Time

12.1 Introduction

In this chapter, we study the maximum flow problem. Given a graph $G = (V, E)$ in which each edge $e \in E$ is assigned a nonnegative capacity $\boldsymbol{\mu}_e$, the **maximum s-t flow problem** asks us to find a flow $\boldsymbol{f}$ that routes as much flow as possible from a source vertex s to a sink vertex t while sending at most $\boldsymbol{\mu}_e$ units of flow over each edge e. Its generalization, the **maximum concurrent multicommodity flow problem**, supplies k source-sink pairs (s_i, t_i) and asks for the maximum α such that we may simultaneously route α units of flow between each source-sink pair. That is, it asks us to find flows $\boldsymbol{f}_1, \ldots, \boldsymbol{f}_k$ (which we think of as corresponding to k different commodities) such that $\boldsymbol{f}_i$ sends α units of flow from s_i to t_i, and $\sum_i |\boldsymbol{f}_i(e)| \leq \boldsymbol{\mu}_e$ for all $e \in E$.

These problems lie at the core of graph algorithms and combinatorial optimization, and they have been extensively studied over the past 60 years [232, 4]. They have found a wide range of theoretical and practical applications [5], and they are widely used as key subroutines in other algorithms (see [18, 233]).

In this chapter, we introduce a new framework for approximately solving flow problems in capacitated, undirected graphs and apply it to provide asymptotically faster algorithms for the maximum s-t flow and maximum concurrent multicommodity flow problems. For graphs with n vertices and m edges, it allows us to find an ε-approximately maximum s-t flows in time $O(m^{1+o(1)}\varepsilon^{-2})$, improving on the previous best bound of $\widetilde{O}(mn^{1/3}\text{poly}(1/\varepsilon))$[51]. Applying the same framework in the multicommodity setting solves a maximum concurrent multicommodity flow problem with k commodities in $O(m^{1+o(1)}\varepsilon^{-2}k^2)$ time, improving on the existing bound of $\widetilde{O}(m^{4/3}\text{poly}(k, \varepsilon^{-1}))$[140].

We believe that both our general framework and several of the pieces necessary for its present instantiation are of independent interest, and we hope that they will find other applications. These include:

- a non-Euclidean generalization of gradient descent, bounds on its performance, and a way to use this to reduce approximate maximum flow and maximum concurrent flow to oblivious routing;
- the definition and efficient construction of *flow sparsifiers*; and
- the construction of a new oblivious routing scheme that can be implemented extremely efficiently.

We have aimed to make our algorithm fairly modular and have thus occasionally worked in slightly more generality than is strictly necessary for the problem at hand. This has slightly increased the length of the exposition, but we believe that it clarifies the high-level structure of the argument, and it will hopefully facilitate the application of these tools in other settings.

12.1.1 Related Work

For the first several decades of its study, the fastest algorithms for the maximum flow problem were essentially all deterministic algorithms based on combinatorial techniques, such as augmenting paths, blocking flows, preflows, and the push-relabel method. These culminated in the work of Goldberg and Rao [104], which computes exact maximum flows in time $O(\min(n^{2/3}, m^{1/2}) \log(n^2/m) \log U)$ on graphs with edge weights in $\{0, \ldots, U\}$. We refer the reader to [104] for a survey of these results.

More recently, a collection of new techniques based on randomization, spectral graph theory and numerical linear algebra, graph decompositions and embeddings, and iterative methods for convex optimization have emerged. These have allowed researchers to provide better provable algorithms for a wide range of flow and cut problems, particularly when one aims to obtain approximately optimal solutions on undirected graphs.

Our algorithm draws extensively on the intellectual heritage established by these works. In this section, we will briefly review some of the previous advances that inform our algorithm. We do not give a comprehensive review of the literature, but instead aim to provide a high-level view of the main tools that motivated the present work, along with the limitations of these tools that had to be overcome. For simplicity of exposition, we primarily focus on the maximum s-t flow problem for the remainder of the introduction.

Sparsification In [33], Benczur and Karger showed how to efficiently approximate any graph G with a sparse graph G' on the same vertex set. To do this, they compute a carefully chosen probability p_e for each $e \in E$, sample each edge e with probability p_e, and include e in G' with its weight increased by a factor of $1/p_e$ if it is sampled. Using this, they obtain, in nearly linear time, a graph G' with $O(n \log n/\varepsilon^2)$ edges such that the total weight of the edges crossing any cut in G' is within a multiplicative factor of $1 \pm \varepsilon$ of the weight crossing the corresponding cut in G. In particular, the Max-Flow Min-Cut Theorem implies that the value of the maximum flow on G' is within a factor of $1 \pm \varepsilon$ of that of G.

This is an extremely effective tool for approximately solving cut problems on a dense graph G, since one can simply solve the corresponding problem on the sparsified graph G'. However, while this means that one can approximately compute the *value* of the maximum s-t flow on G by solving the problem on G', it is not known how to use the maximum s-t flow on G' to obtain an actual approximately maximum flow on G. Intuitively, this is because the weights of edges included in G' are larger than they were in G, and the sampling argument does not provide any guidance about how to route flows over these edges in the original graph G.

Iterative algorithms based on linear systems and electrical flows In 2010, Christiano *et al.*[51] described a new linear algebraic approach to the problem that found ε-approximately maximum s-t flows in time $\widetilde{O}(mn^{1/3}\text{poly}(1/\varepsilon))$. They treated the edges of G as electrical resistors and then computed the *electrical flow* that would result from sending electrical current from s to t in the corresponding circuit. They showed that these flows can be computed in nearly-linear time using fast Laplacian linear system solvers [147, 146, 141, 163], which we further discuss below. The electrical flow obeys the flow conservation constraints, but it could violate the capacity constraints. They then adjusted the resistances of edges to penalize the edges that were flowing too much current and repeated the process. Kelner, Miller, and Peng [140] later showed how to use more general objects that they called *quadratically coupled flows* to use a similar approach to solve the maximum concurrent multicommodity flow problem in time $\widetilde{O}(m^{4/3}\text{poly}(k, 1/\varepsilon))$.

Following this, Lee, Rao, and Srivastava [161] proposed another iterative algorithm that uses electrical flows, but in a way that was substantially different than in [51]. Instead of adjusting the

resistances of the edges in each iteration to correct overflowing edges, they keep the resistances the same but compute a new electrical flow to reroute the excess current. They explain how to interpret this as gradient descent in a certain space, from which a standard analysis would give an algorithm that runs in time $\widetilde{O}(m^{3/2}\text{poly}(1/\varepsilon))$. By replacing the standard gradient descent step with Nesterov's accelerated gradient descent method [206] and using a regularizer to make the penalty function smoother, they obtain an algorithm that runs in time $\widetilde{O}(mn^{1/3}\text{poly}(1/\varepsilon))$ in unweighted graphs.

In all of these algorithms, the superlinear running times arise from an intrinsic $\Theta(\sqrt{m})$ factor introduced by using electrical flows, which minimize an ℓ_2 objective function, to approximate the maximum congestion, which is an ℓ_∞ quantity.

Fast solvers for Laplacian linear systems In their breakthrough paper [243], Spielman and Teng showed how to solve Laplacian systems in nearly-linear time. (This was later sped up and simplified by Koutis, Miller, and Peng [147, 146] and Kelner, Orecchia, Sidford, and Zhu [141].) Their algorithm worked by showing how to approximate the Laplacian $\mathcal{L}_G$ of a graph G with the Laplacian $\mathcal{L}_H$ of a much simpler graph H such that one could use the ability to solve linear systems in $\mathcal{L}_H$ to accelerate the solution of a linear system in $\mathcal{L}_G$. They then applied this recursively to solve the linear systems in $\mathcal{L}_H$. In addition to providing the electrical flow primitive used by the algorithms described above, the structure of their recursive sequence of graph simplifications provides the motivating framework for much of the technical content of our oblivious routing construction.

Oblivious routing In an *oblivious routing scheme*, one specifies a linear operator taking any demand vector to a flow routing these demands over the edges of G. Given a collection of demand vectors, one can produce a multicommodity flow meeting these demands by routing each demand vector using this pre-specified operator, independently of the others. The competitive ratio of such an operator is the worst possible ratio between the congestion incurred by a set of demands in this scheme and the congestion of the best multicommodity flow routing these demands.

In [225], Räcke showed how to construct an oblivious routing scheme with a competitive ratio of $O(\log n)$. His construction worked by providing a probability distribution over trees T_i such that G embeds into each T_i with congestion at most 1, and such that the corresponding convex combination of trees embeds into G with congestion $O(\log n)$. In a sense, one can view this as showing how to approximate G by a probability distribution over trees. Using this, he was able to show how to obtain polylogarithmic approximations for a variety of cut and flow problems, given only the ability to solve these problems on trees.

We note that such an oblivious routing scheme clearly yields a logarithmic approximation to the maximum flow and maximum concurrent multicommodity flow problems. However, Räcke's construction took time substantially superlinear time, making it too slow to be useful for computing approximately maximum flows. Furthermore, it only gives a logarithmic approximation, and it is not clear how to use this a small number of times to reduce the error to a multiplicative ε.

In a later paper [182], Madry applied a recursive technique similar to the one employed by Spielman and Teng in their Laplacian solver to accelerate many of the applications of Räcke's construction at the cost of a worse approximation ratio. Using this, he obtained almost-linear-time polylogarithmic approximation algorithms for a wide variety of cut problems.

Unfortunately, his algorithm made extensive use of sparsification, which, for the previously mentioned reasons, made it unable to solve the corresponding flow problems. This meant that, while it could use flow-cut duality to find a polylogarithmic approximation of the value of a maximum flow, it could not construct a corresponding flow or repeatedly apply such a procedure a small number of times to decrease the error to a multiplicative ε.

In simultaneous, independent work [233], Jonah Sherman used somewhat different techniques to find another almost-linear-time algorithm for the (single-commodity) maximum flow problem. His approach is essentially dual to ours: Our algorithm maintains a flow that routes the given demands throughout its execution and iteratively works to improve its congestion. Our main technical tools thus consist of efficient methods for finding ways to route flow in the graph while maintaining flow conservation. Sherman, on the other hand, maintains a flow that does *not* route the given demands, along with a bound on the congestion required to route the excess flow at the vertices. He then uses this to iteratively work towards achieving flow conservation. (In a sense, our algorithm is more in the spirit of augmenting paths, whereas his is more like preflow-push.) As such, his main technical tools are efficient methods for producing dual objects that give congestion bounds. Objects meeting many of his requirements were given in the work of Madry [182] (whereas there were no previous constructions of flow-based analogues, requiring us to start from scratch); leveraging these allows him to avoid some of the technical complexity required by our approach. We believe that these papers nicely complement each other, and we enthusiastically refer the reader to Sherman's paper.

12.1.2 Our Approach

In this section, we give a high-level description of how we overcome the obstacles described in the previous section. For simplicity, we suppose for the remainder of this introduction that all edges have capacity 1.

The problem is thus to send as many units of flow as possible from s to t without sending more than one unit over any edge. It will be more convenient for us to work with an equivalent congestion minimization problem, where we try to find the unit s-t flow $\boldsymbol{f}$ (i.e., a flow sending one unit from s to t) that minimizes $\|f\|_\infty = \max_e |\boldsymbol{f}_e|$. If we begin with some initial unit s-t flow $\boldsymbol{f}_0$, the goal will be thus be to find the circulation $\boldsymbol{c}$ to add to $\boldsymbol{f}_0$ that minimizes $\|\boldsymbol{f}_0 + \boldsymbol{c}\|_\infty$.

We give an iterative algorithm to approximately find such a $\boldsymbol{c}$. There are $2^{O(\sqrt{\log n \log\log n})}/\varepsilon^2$ iterations, each of which adds a circulation to the present flow and runs in $m \cdot 2^{O(\sqrt{\log n \log\log n})}$ time. Constructing this scheme consists of two main parts: an iterative scheme that reduces the problem to the construction of a projection matrix with certain properties; and the construction of such an operator.

The iterative scheme: Non-Euclidean gradient descent

The simplest way to improve the flow would be to just perform gradient descent on the maximum congestion of an edge. There are two problems with this:

The first problem is that gradient descent depends on having a smoothly varying gradient, but the infinity norm is very far from smooth. This is easily remedied by a standard technique: we replace the infinity norm with a smoother "soft max" function. Doing this would lead to an update that would be a linear projection onto the space of circulations. This could be computed using an electrical flow, and the resulting algorithm would be very similar to the unaccelerated gradient descent algorithm in [161].

The more serious problem is the one discussed in the previous section: the difference between ℓ_2 and ℓ_∞. Gradient steps choose a direction by optimizing a local approximation of the objective function over a sphere, whereas the ℓ_∞ constraint asks us to optimize over a cube. The difference between the size of the largest sphere inside a cube and the smallest sphere containing it gives rise to an inherent $O(\sqrt{m})$ in the number of iterations, unless one can somehow exploit additional structure in the problem.

To deal with this, we introduce and analyze a non-Euclidean variant of gradient descent that

operates with respect to an arbitrary norm.[1] Rather than choosing the direction by optimizing a local linearization of the objective function over the sphere, it performs an optimization over the unit ball in the given norm. By taking this norm to be ℓ_∞ instead of ℓ_2, we are able to obtain a much smaller bound on the number of iterations, albeit at the expense of having to solve a nonlinear minimization problem at every step. The number of iterations required by the gradient descent method depends on how quickly the gradient can change over balls in the norm we are using, which we express in terms of the Lipschitz constant of the gradient in the chosen norm.

To apply this to our problem, we write flows meeting our demands as $\boldsymbol{f}_0 + \boldsymbol{c}$, as described above. We then need a parametrization of the space of circulations so that the objective function (after being smoothed using soft max) has a good bound on its Lipschitz constant. Similarly to what occurs in [141], this comes down to finding a good linear representation of the space of circulations, which we show amounts in the present setting to finding a matrix that projects into the space of circulations while meetings certain norm bounds.

Constructing a projection matrix

This reduces our problem to the construction of such a projection matrix. A simple calculation shows that any linear oblivious routing scheme A with a good competitive ratio gives rise to a projection matrix with the desired properties, and thus leads to an iterative algorithm that converges in a small number of iterations. Each of these iterations performs a matrix-vector multiplication with both $\boldsymbol{A}$ and $\boldsymbol{A}^T$.

Intuitively, this is letting us replace the electrical flows used in previous algorithms with the flows given by an oblivious routing scheme. Since the oblivious routing scheme was constructed to meet ℓ_∞ guarantees, while the electrical flow could only obtain such guarantees by relating ℓ_2 to ℓ_∞, it is quite reasonable that we should expect this to lead to a better iterative algorithm.

However, the computation involved in existing oblivious routing schemes is not fast enough to be used in this setting. Our task thus becomes constructing an oblivious routing scheme that we can compute and work with very efficiently. We do this with a recursive construction that reduces oblivious routing in a graph to oblivious routing in various successively simpler graphs.

To this end, we show that if G can be embedded with low congestion into H (existentially), and H can be embedded with low congestion into G *efficiently*, one can use an oblivious routing on H to obtain an oblivious routing on G. The crucial difference between the simplification operations we perform here and those in previous papers (e.g., in the work of Benczur-Karger [33] and Madry [182]) is that ours are accompanied by such embeddings, which enables us to transfer flows from the simpler graphs to the more complicated ones.

We construct our routing scheme by recursively composing two types of reductions, each of which we show how to implement without incurring a large increase in the competitive ratio:

- **Vertex elimination** This shows how to efficiently reduce oblivious routing on a graph $G = (V, E)$ to routing on t graphs with roughly $\widetilde{O}(|E|/t)$ vertices. To do this, we show how to efficiently embed G into t simpler graphs, each consisting of a tree plus a subgraph supported on roughly $\widetilde{O}(|E|/t)$ vertices. This follows easily from a careful reading of Madry's paper [182]. We then show that routing on such a graph can be reduced to routing on a graph with at most $\widetilde{O}(|E|/t)$ vertices by collapsing paths and eliminating leaves.

[1] This idea and analysis seems to be implicit in other work, e.g., [205] However, we could not find a clean statement like the one we need in the literature, and we have not seen it previously applied in similar settings. We believe that it will find further applications, so we state it in fairly general terms before specializing to what we need for flow problems.

- **Flow sparsification** This allows us to efficiently reduce oblivious routing on an arbitrary graph to oblivious routing on a graph with $\widetilde{O}(|V|)$ edges, which we call a flow sparsifier.
  To construct flow sparsifiers, we use local partitioning to decompose the graph into well-connected clusters that contain many of the original edges. (These clusters are not quite expanders, but they are contained in graphs with good expansion in a manner that is sufficient for our purposes.) We then sparsify these clusters using standard techniques and then show that we can embed the sparse graph back into the original graph using electrical flows. If the graph was originally dense, this results in a sparser graph, and we can recurse on the result. While the implementation of these steps is somewhat different, the outline of this construction parallels Spielman and Teng's approach to the construction of spectral sparsifiers [243, 244].

Combining these two reductions recursively yields an efficient oblivious routing scheme, and thus an algorithm for the maximum flow problem.

Finally, we show that the same framework can be applied to the maximum concurrent multicommodity flow problem. While the norm and regularization change, the structure of the argument and the construction of the oblivious routing scheme go through without requiring substantial modification.

12.2 Preliminaries

Here, we define some notations that is used throughout in this chapter.

General Notation: We let $\boldsymbol{I}$ be the identity matrix and $\boldsymbol{I}_{a\to b} \in \mathbb{R}^{b\times a}$ denote the matrix such that for all $i \le \min\{a,b\}$ we have $\boldsymbol{I}_{ii} = 1$ and $\boldsymbol{I}_{ij} = 0$ otherwise.

Graphs: Throughout this chapter we let $G = (V, E, \boldsymbol{\mu})$ denote an undirected capacitated graph with $n = |V|$ vertices, $m = |E|$ edges, and non-negative capacities $\boldsymbol{\mu} \in \mathbb{R}^E$. We let $w_e \ge 0$ denote the weight of an edge and let $r_e \stackrel{\text{def}}{=} 1/w_e$ denote the resistance of an edge. Here we make no connection between μ_e and r_e; we fix their relationship later.

Sizes: For all $a \in V$ we let $d_a \stackrel{\text{def}}{=} \sum_{\{a,b\}} w_{a,b}$ denote the *(weighted) degree* of vertex a and we let $\deg(a) \stackrel{\text{def}}{=} |\{e \in E \mid e = \{a,b\} \text{ for some } b \in V\}|$ denote its *(combinatorial) degree*. We let $\boldsymbol{D} \in \mathbb{R}^{V\times V}$ be the diagonal matrix where $\boldsymbol{D}_{a,a} = d_a$. Furthermore, for any vertex subset $S \subseteq V$ we define its *volume* by $\mathrm{vol}(S) \stackrel{\text{def}}{=} \sum_{a\in V} d_a$.

Cuts: For any vertex subset $S \subseteq V$ we denote the cut induced by S by the edge subset

$$\partial(S) \stackrel{\text{def}}{=} \{e \in E \mid e \not\subseteq S \text{ and } e \not\subseteq E \setminus S\}$$

and we denote the cost of $F \subseteq E$ by $w(F) \stackrel{\text{def}}{=} \sum_{e\in F} w_e$. We denote the *conductance* of $S \subseteq V$ by

$$\Phi(S) \stackrel{\text{def}}{=} \frac{w(\partial(S))}{\min\{\mathrm{vol}(S), \mathrm{vol}(V-S)\}}$$

and we denote the conductance of a graph by

$$\Phi(G) \stackrel{\text{def}}{=} \min_{S\subseteq V \,:S\notin\{\emptyset,V\}} \phi(S)$$

Subgraphs: For a graph $G = (V, E)$ and a vertex subset $S \subseteq V$ let $G(S)$ denote the subgraph of G consisting of vertex set S and all the edges of E with both endpoints in S, i.e. $\{(a,b) \in E \mid a, b \in S\}$. When we we refer to a graph property, such as vol or Φ, we use subscripts to specify which graph we are considering. For example $\mathrm{vol}_{G(S)}(A)$ denotes the volume of vertex set A in the subgraph of G induced by S.

Congestion: Thinking of edge vectors, $\boldsymbol{f} \in \mathbb{R}^E$, as flows we let the *congestion*[2] *of* $\boldsymbol{f}$ be given by $\text{cong}(\boldsymbol{f}) \stackrel{\text{def}}{=} \|\boldsymbol{U}^{-1}\boldsymbol{f}\|_\infty$. For any collection of flows $\{\boldsymbol{f}_i\} = \{\boldsymbol{f}_1, \ldots, \boldsymbol{f}_k\}$ we overload notation and let their *total congestion* be given by

$$\text{cong}(\{\boldsymbol{f}_i\}) \stackrel{\text{def}}{=} \left\| \boldsymbol{U}^{-1} \sum_i |\boldsymbol{f}_i| \right\|_\infty$$

Demands and Multicommodity Flow: We call a vector $\boldsymbol{\chi} \in \mathbb{R}^V$ a *demand vector* if it is the case that $\sum_{a \in V} \boldsymbol{\chi}(a) = 0$ and we say $\boldsymbol{f} \in \mathbb{R}^E$ meets the demands if $\boldsymbol{B}^T\boldsymbol{f} = \boldsymbol{\chi}$. Given a set of demands $D = \{\boldsymbol{\chi}_1, \ldots, \boldsymbol{\chi}_k\}$, i.e. $\forall i \in [k], \sum_{a \in V} \boldsymbol{\chi}_i(a) = 0$, we denote the optimal low congestion routing of these demands as follows

$$\text{OPT}(D) \stackrel{\text{def}}{=} \min_{\{\boldsymbol{f}_i\} \in \mathbb{R}^E \,:\, \{\boldsymbol{B}^T\boldsymbol{f}_i\} = \{\boldsymbol{\chi}_i\}} \text{cong}(\{\boldsymbol{f}_i\})$$

We call a set of flows $\{\boldsymbol{f}_i\}$ that meet demands $\{\boldsymbol{\chi}_i\}$, i.e. $\forall i, \boldsymbol{B}^T\boldsymbol{f}_i = \boldsymbol{\chi}_i$, a *multicommodity flow* meeting the demands.

Running Time: For matrix $\boldsymbol{A}$, we let $\mathcal{T}(\boldsymbol{A})$ denote the maximum amount of time needed to apply $\boldsymbol{A}$ or $\boldsymbol{A}^T$ to a vector.

12.3 Solving Max-Flow Using a Circulation Projection

12.3.1 Gradient Descent

In this section, we discuss the gradient descent method for general norms. Let $\|\cdot\| : \mathbb{R}^n \to \mathbb{R}$ be an arbitrary norm on $\mathbb{R}^n$ and recall that the gradient of f at $\boldsymbol{x}$ is defined to be the vector $\nabla f(\boldsymbol{x}) \in \mathbb{R}^n$ such that

$$f(\boldsymbol{y}) = f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle + o(\|\boldsymbol{y} - \boldsymbol{x}\|). \tag{12.1}$$

The gradient descent method is a greedy minimization method that updates the current vector, $\boldsymbol{x}$, using the direction which minimizes $\langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle$. To analyze this method's performance, we need a tool to compare the improvement $\langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle$ with the step size, $\|\boldsymbol{y} - \boldsymbol{x}\|$, and the quantity, $\|\nabla f(\boldsymbol{x})\|$. For ℓ_2 norm, this can be done by Cauchy Schwarz inequality and in general, we can define a new norm for $\nabla f(\boldsymbol{x})$ to make this happens. We call this the *dual norm* $\|\cdot\|^*$ defined as follows

$$\|\boldsymbol{x}\|^* \stackrel{\text{def}}{=} \max_{\boldsymbol{y} \in \mathbb{R}^n \,:\, \|\boldsymbol{y}\| \le 1} \langle \boldsymbol{y}, \boldsymbol{x} \rangle.$$

Fact 12.8.4 shows that this definition indeed yields that $\langle \boldsymbol{y}, \boldsymbol{x} \rangle \le \|\boldsymbol{y}\|^* \|\boldsymbol{x}\|$. Next, we define the fastest increasing direction $x^\#$, which is an arbitrary point satisfying the following

$$\boldsymbol{x}^\# \stackrel{\text{def}}{=} \arg\max_{\boldsymbol{s} \in \mathbb{R}} \langle \boldsymbol{x}, \boldsymbol{s} \rangle - \frac{1}{2}\|\boldsymbol{s}\|^2.$$

In the appendix, we provide some facts about $\|\cdot\|^*$ and $\boldsymbol{x}^\#$ that we will use in this section. Using the notations defined, the gradient descent method simply produces a sequence of $\boldsymbol{x}_k$ such that

$$\boldsymbol{x}_{k+1} := \boldsymbol{x}_k - t_k(\nabla f(\boldsymbol{x}_k))^\#$$

[2] Note that here and in the rest of the chapter we will focus our analysis with congestion with respect to the norm $\|\cdot\|_\infty$ and we will look at oblivious routing strategies that are competitive with respect to this norm. However, many of the results present are easily generalizable to other norms. These generalizations are outside the scope of this thesis.

where t_k is some chosen step size for iteration k. To determine what these step sizes should be we need some information about the smoothness of the function, in particular, the magnitude of the second order term in (12.1). The natural notion of smoothness for gradient descent is the Lipschitz constant of the gradient of f, that is the smallest constant L such that

$$\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n \quad : \quad \|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\|^* \leq L \cdot \|\boldsymbol{x} - \boldsymbol{y}\|.$$

In the appendix we provide an equivalent definition and a way to compute L, which is useful later.

Let $X^* \subseteq \mathbb{R}^n$ denote the set of optimal solutions to the unconstrained minimization problem $\min_{x \in \mathbb{R}^n} f$ and let f^* denote the optimal value of this minimization problem, i.e.

$$\forall \boldsymbol{x} \in X^* \ : \ f(\boldsymbol{x}) = f^* = \min_{\boldsymbol{y} \in \mathbb{R}^n} f(y) \quad \text{and} \quad \forall \boldsymbol{x} \notin X^* \ : \ f(\boldsymbol{x}) > f^*$$

We assume that X^* is non-empty. Now, we are ready to estimate the convergence rate of the gradient descent method.

Theorem 12.3.1. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex continuously differentiable function and let L be the Lipschitz constant of ∇f. For initial point $\boldsymbol{x}_0 \in \mathbb{R}^n$ we define a sequence of $\boldsymbol{x}_k$ by the update rule*

$$\boldsymbol{x}_{k+1} := \boldsymbol{x}_k - \frac{1}{L}(\nabla f(\boldsymbol{x}_k))^{\#}$$

For all $k \geq 0$, we have

$$f(\boldsymbol{x}_k) - f^* \leq \frac{2 \cdot L \cdot R^2}{k+4} \quad \text{where} \quad R \stackrel{\text{def}}{=} \max_{\boldsymbol{x} \in \mathbb{R}^n : f(\boldsymbol{x}) \leq f(\boldsymbol{x}_0)} \min_{\boldsymbol{x}^* \in X^*} \|\boldsymbol{x} - \boldsymbol{x}^*\|.$$

Proof. [3] By the Lipschitz continuity of the gradient of f and Lemma 12.8.5 we have

$$f(\boldsymbol{x}_{k+1}) \leq f(\boldsymbol{x}_k) - \frac{1}{2L} \left(\|\nabla f(\boldsymbol{x}_k)\|^*\right)^2.$$

Furthermore, by the convexity of f, we know that

$$\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n \quad : \quad f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle.$$

Using this and the fact that $f(\boldsymbol{x}_k)$ decreases monotonically with k, we get

$$f(\boldsymbol{x}_k) - f^* \leq \min_{\boldsymbol{x}^* \in X^*} \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{x}_k - \boldsymbol{x}^* \rangle \leq \min_{\boldsymbol{x}^* \in X^*} \|\nabla f(\boldsymbol{x}_k)\|^* \|\boldsymbol{x}_k - \boldsymbol{x}^*\| \leq R \|\nabla f(\boldsymbol{x}_k)\|^*.$$

Therefore, letting $\phi_k \stackrel{\text{def}}{=} f(\boldsymbol{x}_k) - f^*$, we have

$$\phi_k - \phi_{k+1} \geq \frac{1}{2L} (\|\nabla f(\boldsymbol{x}_k)\|^*)^2 \geq \frac{\phi_k^2}{2 \cdot L \cdot R^2}.$$

Furthermore, since $\phi_k \geq \phi_{k+1}$, we have

$$\frac{1}{\phi_{k+1}} - \frac{1}{\phi_k} = \frac{\phi_k - \phi_{k+1}}{\phi_k \phi_{k+1}} \geq \frac{\phi_k - \phi_{k+1}}{\phi_k^2} \geq \frac{1}{2 \cdot L \cdot R^2}.$$

So, by induction, we have that

$$\frac{1}{\phi_k} - \frac{1}{\phi_0} \geq \frac{k}{2 \cdot L \cdot R^2}.$$

[3] The structure of this specific proof was modeled after a proof in [205] for a slightly different problem.

Now, note that since $\nabla f(\boldsymbol{x}^*) = 0$, we have that

$$f(\boldsymbol{x}_0) \leq f(\boldsymbol{x}^*) + \langle \nabla f(\boldsymbol{x}^*), \boldsymbol{x}_0 - \boldsymbol{x}^* \rangle + \frac{L}{2}\|\boldsymbol{x}_0 - \boldsymbol{x}^*\|^2 \leq f(\boldsymbol{x}^*) + \frac{L}{2}R^2.$$

So, we have that $\phi_0 \leq \frac{L}{2}R^2$ and putting this all together yields that

$$\frac{1}{\phi_k} \geq \frac{1}{\phi_0} + \frac{k}{2 \cdot L \cdot R^2} \geq \frac{4}{2 \cdot L \cdot R^2} + \frac{k}{2 \cdot L \cdot R^2}.$$

□

■ 12.3.2 Maximum Flow Formulation

For an arbitrary set of demands $\boldsymbol{\chi} \in \mathbb{R}^V$ we wish to solve the following *maximum flow* problem

$$\max_{\alpha \in R, \boldsymbol{f} \in \mathbb{R}^E} \alpha \qquad \text{subject to} \qquad \boldsymbol{B}^T \boldsymbol{f} = \alpha \boldsymbol{\chi} \quad \text{and} \quad \|\boldsymbol{U}^{-1}\boldsymbol{f}\|_\infty \leq 1.$$

Equivalently, we want to compute a *minimum congestion flow*

$$\min_{\boldsymbol{f} \in \mathbb{R}^E \,:\, \boldsymbol{B}^T \boldsymbol{f} = \boldsymbol{\chi}} \|\boldsymbol{U}^{-1}\boldsymbol{f}\|_\infty.$$

where we call $\|\boldsymbol{U}^{-1}\boldsymbol{f}\|_\infty$ *the congestion* of $\boldsymbol{f}$.

Letting $\boldsymbol{f}_0 \in \mathbb{R}^E$ be some initial *feasible flow*, i.e. $\boldsymbol{B}^T \boldsymbol{f}_0 \stackrel{\text{def}}{=} \boldsymbol{\chi}$, we write the problem equivalently as

$$\min_{\boldsymbol{c} \in \mathbb{R}^E \,:\, \boldsymbol{B}^T \boldsymbol{c} = 0} \|\boldsymbol{U}^{-1}(\boldsymbol{f}_0 + \boldsymbol{c})\|_\infty$$

where the output flow is $\boldsymbol{f} = \boldsymbol{f}_0 + \boldsymbol{c}$. Although the gradient descent method is applicable to constrained optimization problems and has a similar convergence guarantee, the sub-problem involved in each iteration is a constrained optimization problem, which is quite complicated in this case. Since the domain is a linear subspace, the constraints can be avoided by projecting the variables onto this subspace.

Formally, we define a circulation projection matrix as follows.

Definition 12.3.2. A matrix $\tilde{\boldsymbol{P}} \in \mathbb{R}^{E \times E}$ is a *circulation projection matrix* if it is a projection matrix onto the circulation space, i.e. it satisfies the following

- $\forall \boldsymbol{x} \in \mathbb{R}^E$ we have $\boldsymbol{B}^T \tilde{\boldsymbol{P}} \boldsymbol{x} = \boldsymbol{0}$.
- $\forall \boldsymbol{x} \in \mathbb{R}^E$ with $\boldsymbol{B}^T \boldsymbol{x} = \boldsymbol{0}$ we have $\boldsymbol{P}\boldsymbol{x} = \boldsymbol{x}$.

Then, the problem becomes

$$\min_{\boldsymbol{c} \in \mathbb{R}^E} \|\boldsymbol{U}^{-1}(\boldsymbol{f}_0 + \tilde{\boldsymbol{P}}\boldsymbol{c})\|_\infty.$$

Applying gradient descent on this problem is similar to applying projected gradient method on the original problem. But, instead of using the orthogonal projection that is not suitable for $\|\cdot\|_\infty$, we will pick a better projection matrix.

Applying the change of basis $\boldsymbol{x} = \boldsymbol{U}^{-1}\boldsymbol{c}$ and letting $\boldsymbol{\alpha_0} = \boldsymbol{U}^{-1}\boldsymbol{f}_0$ and $\boldsymbol{P} = \boldsymbol{U}^{-1}\tilde{\boldsymbol{P}}\boldsymbol{U}$, we write the problem equivalently as

$$\min_{\boldsymbol{x} \in \mathbb{R}^E} \|\boldsymbol{\alpha_0} + \boldsymbol{P}\boldsymbol{x}\|_\infty$$

where the output maximum flow is

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{U}(\boldsymbol{\alpha}_0 + \boldsymbol{P}\boldsymbol{x})/\|\boldsymbol{U}(\boldsymbol{\alpha}_0 + \boldsymbol{P}\boldsymbol{x})\|_\infty.$$

12.3.3 An Approximate Maximum Flow Algorithm

Since the gradient descent method requires the objective function to be differentiable, we introduce a smooth version of $\|\cdot\|_\infty$ which we call smax_t. In next section, we prove that there is a convex differentiable function smax_t such that $\nabla\mathrm{smax}_t$ is Lipschitz continuous with Lipschitz constant $\frac{1}{t}$ and such that

$$\forall \boldsymbol{x} \in \mathbb{R}^E \quad : \quad \|\boldsymbol{x}\|_\infty - t\ln(2m) \le \mathrm{smax}_t(\boldsymbol{x}) \le \|\boldsymbol{x}\|_\infty \quad .$$

Now we consider the following regularized optimization problem

$$\min_{\boldsymbol{x}\in\mathbb{R}^E} g_t(\boldsymbol{x}) \quad \text{where} \quad g_t(\boldsymbol{x}) = \mathrm{smax}_t(\boldsymbol{\alpha}_0 + \boldsymbol{P}\boldsymbol{x}).$$

For the rest of this section, we consider solving this optimization problem using gradient descent under $\|\cdot\|_\infty$.

First, we bound the Lipschitz constant of the gradient of g_t.

Lemma 12.3.3. *The gradient of g_t is Lipschitz continuous with Lipschitz constant $L = \frac{\|\boldsymbol{P}\|_\infty^2}{t}$.*

Proof. By Lemma 12.8.5 and the Lipschitz continuity of $\nabla\mathrm{smax}_t$, we have

$$\mathrm{smax}_t(\boldsymbol{y}) \le \mathrm{smax}_t(\boldsymbol{x}) + \langle\nabla\mathrm{smax}_t(\boldsymbol{x}), \boldsymbol{y}-\boldsymbol{x}\rangle + \frac{1}{2t}\|\boldsymbol{y}-\boldsymbol{x}\|_\infty.$$

Seting $\boldsymbol{x} \leftarrow \boldsymbol{\alpha}_0 + \boldsymbol{P}\boldsymbol{x}$ and $\boldsymbol{y} \leftarrow \boldsymbol{\alpha}_0 + \boldsymbol{P}\boldsymbol{y}$, we have

$$\begin{aligned} g_t(\boldsymbol{y}) &\le g_t(\boldsymbol{y}) + \langle\nabla\mathrm{smax}_t(\boldsymbol{\alpha}_0 + \boldsymbol{P}\boldsymbol{x}), \boldsymbol{P}\boldsymbol{y} - \boldsymbol{P}\boldsymbol{x}\rangle + \frac{1}{2t}\|\boldsymbol{P}\boldsymbol{y} - \boldsymbol{P}\boldsymbol{x}\|_\infty^2 \\ &\le g_t(\boldsymbol{y}) + \langle\boldsymbol{P}^T\nabla\mathrm{smax}_t(\boldsymbol{\alpha}_0 + \boldsymbol{P}\boldsymbol{x}), \boldsymbol{y}-\boldsymbol{x}\rangle + \frac{1}{2t}\|\boldsymbol{P}\|_\infty^2\|\boldsymbol{y}-\boldsymbol{x}\|_\infty^2 \\ &= g_t(\boldsymbol{y}) + \langle\nabla g_t(\boldsymbol{x}), \boldsymbol{y}-\boldsymbol{x}\rangle + \frac{1}{2t}\|\boldsymbol{P}\|_\infty^2\|\boldsymbol{y}-\boldsymbol{x}\|_\infty^2. \end{aligned}$$

Hence, the result follows from Lemma 12.8.5. □

Now, we apply gradient descent to find an approximate max flow as follows.

Algorithm 33: MaxFlow

Input: any initial feasible flow $\boldsymbol{f}_0$ and $\mathrm{OPT} = \min_{\boldsymbol{x}} \|\boldsymbol{U}^{-1}\boldsymbol{f}_0 + \boldsymbol{P}\boldsymbol{x}\|_\infty$.
Let $\boldsymbol{\alpha}_0 = (\boldsymbol{I} - \boldsymbol{P})\boldsymbol{U}^{-1}\boldsymbol{f}_0$ and $\boldsymbol{x}_0 = 0$.
Let $t = \varepsilon\mathrm{OPT}/2\ln(2m)$ and $k = 300\|\boldsymbol{P}\|_\infty^4\ln(2m)/\varepsilon^2$.
Let $g_t = \mathrm{smax}_t(\boldsymbol{\alpha}_0 + \boldsymbol{P}\boldsymbol{x})$.
for $i = 1, 2, \cdots, k$ **do**
  $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i - \frac{t}{\|\boldsymbol{P}\|_\infty^2}(\nabla g_t(\boldsymbol{x}_i))^\#$. (See Lemma 12.3.5)
end
Output: $\boldsymbol{U}\left(\vec{\alpha_0} + \boldsymbol{P}\boldsymbol{x}_k\right)/\|\vec{\alpha_0} + \boldsymbol{P}\boldsymbol{x}_k\|_\infty$.

We remark that the initial flow can be obtained by BFS and the OPT value can be approximted using binary search. In Section 12.7, we will give an algorithm with better dependence on $\|\boldsymbol{P}\|$.

Theorem 12.3.4. *Let $\widetilde{\boldsymbol{P}}$ be a cycle projection matrix, let $\boldsymbol{P} = \boldsymbol{U}^{-1}\widetilde{\boldsymbol{P}}\boldsymbol{U}$, and let $\varepsilon < 1$. **MaxFlow** outputs an $(1-\varepsilon)$-approximate maximum flow in time*

$$O\left(\frac{\|\boldsymbol{P}\|_\infty^4 \ln(m)\,(\mathcal{T}(\boldsymbol{P}) + m)}{\varepsilon^2}\right).$$

Proof. First, we bound $\|\vec{\alpha_0}\|_\infty$. Let $\boldsymbol{x}^*$ be a minimizer of $\min_{\boldsymbol{x}} \|\boldsymbol{U}^{-1}\boldsymbol{f}_0 + \boldsymbol{P}\boldsymbol{x}\|_\infty$ such that $\boldsymbol{P}\vec{x}^* = \vec{x}^*$. Then, we have

$$\begin{aligned}
\|\boldsymbol{\alpha}_0\|_\infty &= \|\boldsymbol{U}^{-1}\boldsymbol{f}_0 - \boldsymbol{P}\boldsymbol{U}^{-1}\boldsymbol{f}_0\|_\infty \\
&\le \|\boldsymbol{U}^{-1}\boldsymbol{f}_0 + \boldsymbol{x}^*\|_\infty + \|\boldsymbol{x}^* + \boldsymbol{P}\boldsymbol{U}^{-1}\boldsymbol{f}_0\|_\infty \\
&= \|\boldsymbol{U}^{-1}\boldsymbol{f}_0 + \boldsymbol{x}^*\|_\infty + \|\boldsymbol{P}\boldsymbol{x}^* + \boldsymbol{P}\boldsymbol{U}^{-1}\boldsymbol{f}_0\|_\infty \\
&\le (1 + \|\boldsymbol{P}\|_\infty)\,\|\boldsymbol{U}^{-1}\boldsymbol{f}_0 + \boldsymbol{x}^*\|_\infty \\
&= (1 + \|\boldsymbol{P}\|_\infty)\,\mathrm{OPT}.
\end{aligned}$$

Second, we bound R in Theorem 12.3.1. Note that

$$g_t(\boldsymbol{x}_0) = \mathrm{smax}_t(\boldsymbol{\alpha}_0) \le \|\boldsymbol{\alpha}_0\|_\infty \le (1 + \|\boldsymbol{P}\|_\infty)\,\mathrm{OPT}.$$

Hence, the condition $g_t(\boldsymbol{x}) \le g_t(\boldsymbol{x}_0)$ implies that

$$\|\boldsymbol{\alpha}_0 + \boldsymbol{P}\boldsymbol{x}\|_\infty \le (1 + \|\boldsymbol{P}\|_\infty)\,\mathrm{OPT} + t\ln(2m).$$

For any $\boldsymbol{y} \in X^*$ let $\boldsymbol{c} = \boldsymbol{x} - \boldsymbol{P}\boldsymbol{x} + \boldsymbol{y}$ and note that $\boldsymbol{P}\boldsymbol{c} = \boldsymbol{P}\boldsymbol{y}$ and therefore $\boldsymbol{c} \in X^*$. Using these facts, we can bound R as follows

$$\begin{aligned}
R &= \max_{\boldsymbol{x}\in\mathbb{R}^E \,:\, g_t(\boldsymbol{x}) \le g_t(\boldsymbol{x}_0)} \left\{ \min_{\boldsymbol{x}^* \in X^*} \|\boldsymbol{x} - \boldsymbol{x}^*\|_\infty \right\} \\
&\le \max_{\boldsymbol{x}\in\mathbb{R}^E \,:\, g_t(\boldsymbol{x}) \le g_t(\boldsymbol{x}_0)} \|\boldsymbol{x} - \boldsymbol{c}\|_\infty \\
&\le \max_{\boldsymbol{x}\in\mathbb{R}^E \,:\, g_t(\boldsymbol{x}) \le g_t(\boldsymbol{x}_0)} \|\boldsymbol{P}\boldsymbol{x} - \boldsymbol{P}\boldsymbol{y}\|_\infty \\
&\le \max_{\boldsymbol{x}\in\mathbb{R}^E \,:\, g_t(\boldsymbol{x}) \le g_t(\boldsymbol{x}_0)} \|\boldsymbol{P}\boldsymbol{x}\|_\infty + \|\boldsymbol{P}\boldsymbol{y}\|_\infty \\
&\le 2\|\boldsymbol{\alpha}_0\|_\infty + \|\boldsymbol{\alpha}_0 + \boldsymbol{P}\boldsymbol{x}\|_\infty + \|\boldsymbol{\alpha}_0 + \boldsymbol{P}\boldsymbol{y}\|_\infty \\
&\le 2\|\boldsymbol{\alpha}_0\|_\infty + 2\|\boldsymbol{\alpha}_0 + \boldsymbol{P}\boldsymbol{x}\|_\infty \\
&\le 4\,(1 + \|\boldsymbol{P}\|_\infty)\,\mathrm{OPT} + 2t\ln(2m).
\end{aligned}$$

From Lemma 12.3.3, we know that the Lipschitz constant of ∇g_t is $\|\boldsymbol{P}\|_\infty^2/t$. Hence, Theorem 12.3.1 shows that

$$\begin{aligned}
g_t(\boldsymbol{x}_k) &\le \min_{\boldsymbol{x}} g_t(\boldsymbol{x}) + \frac{2\cdot L\cdot R^2}{k+4} \\
&\le \mathrm{OPT} + \frac{2\cdot L\cdot R^2}{k+4}.
\end{aligned}$$

So, we have

$$\begin{aligned}
\|\boldsymbol{\alpha}_0 + \boldsymbol{P}\boldsymbol{x}_k\|_\infty &\le g_t(\boldsymbol{x}_k) + t\ln(2m) \\
&\le \mathrm{OPT} + t\ln(2m) + \frac{2\|\boldsymbol{P}\|_\infty^2}{t(k+4)}\left(4\,(1 + \|\boldsymbol{P}\|_\infty)\,\mathrm{OPT} + 2t\ln(2m)\right)^2.
\end{aligned}$$

Using $t = \varepsilon \mathrm{OPT}/2\ln(2m)$ and $k = 300\|\boldsymbol{P}\|_\infty^4 \ln(2m)/\varepsilon^2$, we have

$$\|\boldsymbol{\alpha}_0 + \boldsymbol{P}\boldsymbol{x}_k\|_\infty \leq (1+\varepsilon)\mathrm{OPT}.$$

Therefore, $\boldsymbol{\alpha}_0 + \boldsymbol{P}\boldsymbol{x}_k$ is an $(1-\varepsilon)$ approximate maximum flow.

Now, we estimate the running time. In each step 5, we are required to compute $(\nabla g(\boldsymbol{x}_k))^\#$. The gradient

$$\nabla g(\boldsymbol{x}) = \boldsymbol{P}^T \nabla \mathrm{smax}_t(\boldsymbol{\alpha}_0 + \boldsymbol{P}\boldsymbol{x})$$

can be computed in $O(\mathcal{T}(\boldsymbol{P}) + m)$ using the formula of the gradient of smax_t, applications of $\boldsymbol{P}$ and $\boldsymbol{P}^T$. Lemma 12.3.5 shows that the $\#$ operator can be computed in $O(m)$. □

Lemma 12.3.5. *In $\|\cdot\|_\infty$, the $\#$ operator is given by the explicit formula*

$$\left(\boldsymbol{x}^\#\right)_e = \mathrm{sign}(x_e)\|\boldsymbol{x}\|_1 \quad \textit{for } e \in E.$$

Proof. Recall that

$$\boldsymbol{x}^\# \quad = \quad \arg\max_{\boldsymbol{s}\in\mathbb{R}} \langle \boldsymbol{x}, \boldsymbol{s}\rangle - \frac{1}{2}\|\vec{s}\|_\infty^2.$$

It is easy to see that for all $e \in E$, $||\boldsymbol{x}^\#||_\infty = \left|\left(\boldsymbol{x}^\#\right)_e\right|$. In particular, we have

$$\left(\boldsymbol{x}^\#\right)_e = \mathrm{sign}(x_e)||\boldsymbol{x}^\#||_\infty.$$

Fact 12.8.3 shows that $||\boldsymbol{x}^\#||_\infty = \|\boldsymbol{x}\|_1$ and the result follows. □

12.3.4 Properties of soft max

In this section, we define smax_t and discuss its properties. Formally, the regularized convex function can be found by smoothing technique using convex conjugate [208] [35, Sec 5.4]. For simplicity and completeness, we define it explicitly and prove its properties directly. Formally, we define

$$\forall \boldsymbol{x} \in \mathbb{R}^E, \forall t \in \mathbb{R}_{>0} \quad : \quad \mathrm{smax}_t(\boldsymbol{x}) \stackrel{\mathrm{def}}{=} t\ln\left(\frac{\sum_{e\in E}\exp\left(\frac{x_e}{t}\right) + \exp\left(-\frac{x_e}{t}\right)}{2m}\right).$$

For notational simplicity, for all $\boldsymbol{x}$ where this vector is clear from context, we define $\boldsymbol{c}$ and $\boldsymbol{s}$ as follows

$$\forall e \in E \quad : \quad \boldsymbol{c}_e \stackrel{\mathrm{def}}{=} \exp\left(\frac{x_e}{t}\right) + \exp\left(-\frac{x_e}{t}\right) \quad \text{and} \quad \boldsymbol{s}_e \stackrel{\mathrm{def}}{=} \exp\left(\frac{x_e}{t}\right) - \exp\left(-\frac{x_e}{t}\right),$$

where the letters are chosen due to the very close resemblance to hyperbolic sine and hyperbolic cosine.

Lemma 12.3.6.

$$\forall \boldsymbol{x} \in \mathbb{R}^n \quad : \quad \nabla \mathrm{smax}_t(\boldsymbol{x}) = \frac{1}{\boldsymbol{1}^T\boldsymbol{c}}\boldsymbol{s}$$

$$\forall \boldsymbol{x} \in \mathbb{R}^n \quad : \quad \nabla^2 \mathrm{smax}_t(\boldsymbol{x}) = \frac{1}{t\left(\boldsymbol{1}^T\boldsymbol{c}\right)}\left[\boldsymbol{diag}(\boldsymbol{c}) - \frac{\boldsymbol{s}\boldsymbol{s}^T}{\boldsymbol{1}^T\boldsymbol{c}}\right]$$

Proof. For all $i \in E$ and $\boldsymbol{x} \in \mathbb{R}^E$, we have

$$\begin{aligned}\frac{\partial}{\partial x_i}\mathrm{smax}_t(\boldsymbol{x}) &= \frac{\partial}{\partial x_i}\left(t\ln\left(\frac{\sum_{e\in E}\exp\left(\frac{x_e}{t}\right)+\exp\left(-\frac{x_e}{t}\right)}{2m}\right)\right)\\ &= \frac{\exp\left(\frac{x_i}{t}\right)-\exp\left(-\frac{x_i}{t}\right)}{\sum_{e\in E}\exp\left(\frac{x_e}{t}\right)+\exp\left(-\frac{x_e}{t}\right)}.\end{aligned}$$

For all $i, j \in E$ and $\boldsymbol{x} \in \mathbb{R}^E$, we have

$$\begin{aligned}\frac{\partial^2}{\partial x_i \partial x_j}\mathrm{smax}_t(\boldsymbol{x}) &= \frac{\partial^2}{\partial x_i \partial x_j}\left(t\ln\left(\frac{\sum_{e\in E}\exp\left(\frac{x_e}{t}\right)+\exp\left(-\frac{x_e}{t}\right)}{2m}\right)\right)\\ &= \frac{\partial}{\partial j}\left[\frac{\exp\left(\frac{x_i}{t}\right)-\exp\left(-\frac{x_i}{t}\right)}{\sum_{e\in E}\exp\left(\frac{x_e}{t}\right)+\exp\left(-\frac{x_e}{t}\right)}\right]\\ &= \frac{1}{t}\frac{\left(\mathbf{1}^T\boldsymbol{c}\right)\mathbf{1}_{i=j}\left(\boldsymbol{c}_i\right)-\boldsymbol{s}_i\boldsymbol{s}_j}{\left(\mathbf{1}^T\boldsymbol{c}\right)^2}.\end{aligned}$$

□

Lemma 12.3.7. *The function* smax_t *is a convex continuously differentiable function and it has Lipschitz continuous gradient with Lipschitz constant* $1/t$ *and*

$$\|\boldsymbol{x}\|_\infty - t\ln(2m) \le \mathrm{smax}_t(\boldsymbol{x}) \le \|\boldsymbol{x}\|_\infty$$

for $\boldsymbol{x} \in \mathbb{R}^E$.

Proof. By the formulation of the Hessian, for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^E$, we have

$$\boldsymbol{y}^T\left(\nabla^2\mathrm{smax}_t(\boldsymbol{x})\right)\boldsymbol{y} \le \frac{\sum_i c_i \boldsymbol{y}_i^2}{t(\mathbf{1}^T\boldsymbol{c})} \le \frac{\sum_i c_i(\max_j \boldsymbol{y}_j^2)}{t(\mathbf{1}^T\boldsymbol{c})} \le \frac{1}{t}\|y\|_\infty^2.$$

On the other side, for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^E$, we have by $s_i \le |s_i| \le c_i$ and Cauchy Schwarz shows that

$$\boldsymbol{y}^T\boldsymbol{s}\boldsymbol{s}^T\boldsymbol{y} \le (\mathbf{1}^T|s|)(\boldsymbol{y}^T\boldsymbol{diag}(|s|)\boldsymbol{y}). \le (\mathbf{1}^T\boldsymbol{c})(\boldsymbol{y}^T\boldsymbol{diag}(\boldsymbol{c})\boldsymbol{y}).$$

and hence

$$0 \le \boldsymbol{y}^T\left(\nabla^2\mathrm{smax}_t(\boldsymbol{x})\right)\boldsymbol{y}.$$

Thus, the first part follows from Lemma 12.8.6. For the later part, we have

$$\|\boldsymbol{x}\|_\infty \ge t\ln\left(\frac{\sum_{e\in E}\exp\left(\frac{x_e}{t}\right)+\exp\left(-\frac{x_e}{t}\right)}{2m}\right) \ge t\ln\left(\frac{\exp\left(\frac{\|\boldsymbol{x}\|_\infty}{t}\right)}{2m}\right) = \|\boldsymbol{x}\|_\infty - \ln(2m).$$

□

12.4 Oblivious Routing

In the previous sections, we saw how a circulation projection matrix can be used to solve max flow. In the next few sections, we show how to efficiently construct a circulation projection matrix to obtain an almost linear time algorithm for solving max flow.

Our proof focuses on the notion of (linear) oblivious routings. Rather than constructing the circulation projection matrix directly, we show how the efficient construction of an oblivious routing

algorithm with a good competitive ratio immediately allows us to produce a circulation projection matrix.

In the remainder of this section, we formally define oblivious routings and prove the relationship between oblivious routing and circulation projection matrices (Section 12.4.1), provide a high level overview of our recursive approach and state the main theorems we will prove in later sections (Section 12.4.2). Finally, we prove the main theorem about our almost-linear-time construction of circulation projection with norm $2^{O(\sqrt{\log(n)\log\log(n)})}$ assuming the proofs in the later sections (Section 12.4.3).

12.4.1 From Oblivious Routing to Circulation Projection

Here we provide definitions and prove basic properties of *oblivious routings*, that is, fixed mappings from demands to flows that meet the input demands. While non-linear algorithms could be considered, we restrict our attention to linear oblivious routing strategies and use the term oblivious routing to refer to the linear subclass for the remainder of this chapter.[4]

Definition 12.4.1 (Oblivious Routing)**.** An *oblivious routing* on graph $G = (V, E)$ is a linear operator $\boldsymbol{A} \in \mathbb{R}^{E\times V}$ such that for all demands $\boldsymbol{\chi} \in \mathbb{R}^V$, $\boldsymbol{B}^T\boldsymbol{A}\boldsymbol{\chi} = \boldsymbol{\chi}$. We call $\boldsymbol{A}\boldsymbol{\chi}$ the *routing* of $\boldsymbol{\chi}$ by $\boldsymbol{A}$.

Given an oblivious routing strategy $\boldsymbol{A}$ and a set of demands $D = \{\boldsymbol{\chi}_1, \ldots, \boldsymbol{\chi}_k\}$, one can construct a multicommodity flow satisfying all the demands in D by using $\boldsymbol{A}$ to route each demand individually, obliviously to the existence of the other demands. We measure the *competitive ratio* of such an oblivious routing strategy to be the ratio of the worst relative congestion of such a routing to the minimal-congestion routing of the demands.

Definition 12.4.2 (Competitive Ratio)**.** The *competitive ratio of oblivious routing* $\boldsymbol{A} \in \mathbb{R}^{E\times V}$, denoted $\rho(\boldsymbol{A})$, is given by

$$\rho(\boldsymbol{A}) \stackrel{\text{def}}{=} \max_{\{\boldsymbol{\chi}_i\} \,:\, \forall i\ \boldsymbol{\chi}_i \perp \mathbf{1}} \frac{\text{cong}(\{\boldsymbol{A}\boldsymbol{\chi}_i\})}{\text{OPT}(\{\boldsymbol{\chi}_i\})}$$

At times, it will be more convenient to analyze an oblivious routing as a linear algebraic object rather a combinatorial algorithm; towards this end, we note that the competitive ratio of a linear oblivious routing strategy can be gleaned from the operator norm of a related matrix (see also [158] and [139]). Below, we state and prove a generalization of this result to weighted graphs that will be vital to relating $\boldsymbol{A}$ to $\widetilde{\boldsymbol{P}}$.

Lemma 12.4.3. *For any oblivious routing* $\boldsymbol{A}$*, we have* $\rho(\boldsymbol{A}) = \|\boldsymbol{U}^{-1}\boldsymbol{A}\boldsymbol{B}^T\boldsymbol{U}\|_\infty$

Proof. For a set of demands D, let D_∞ be the set of demands that results by taking the routing of every demand in D by $\text{OPT}(D)$ and splitting it up into demands on every edge corresponding to the flow sent by $\text{OPT}(D)$. Now, clearly $\text{OPT}(D) = \text{OPT}(D_\infty)$ since routing D can be used to route D_∞ and vice versa, and clearly $\text{cong}(\boldsymbol{A}D) \le \text{cong}(\boldsymbol{A}D_\infty)$ by the linearity of $\boldsymbol{A}$ (routing D_∞ simply doesn't reward $\boldsymbol{A}$ routing for cancellations). Using this and the fact that $D_\infty \subseteq D$, we have:

$$\begin{aligned}\rho_p(\boldsymbol{A}) &= \max_D \frac{\text{cong}(\{\boldsymbol{A}D\})}{\text{OPT}(D)} = \max_{D_\infty} \frac{\text{cong}(\boldsymbol{A}D_\infty)}{\text{OPT}(D_\infty)} = \max_{\boldsymbol{x}\in\mathbb{R}^E} \frac{\|\sum_{e\in E} \boldsymbol{x}_e \left|\boldsymbol{U}^{-1}\boldsymbol{A}\boldsymbol{\chi}_e\right|\|_\infty}{\|\boldsymbol{U}^{-1}\boldsymbol{x}\|_\infty} \\ &= \max_{\boldsymbol{x}\in\mathbb{R}^E} \frac{\|\left|\boldsymbol{U}^{-1}\boldsymbol{A}\boldsymbol{B}^T\right|\boldsymbol{x}\|_\infty}{\|\boldsymbol{U}^{-1}\boldsymbol{x}\|_\infty} = \max_{\boldsymbol{x}\in\mathbb{R}^E} \frac{\|\left|\boldsymbol{U}^{-1}\boldsymbol{A}\boldsymbol{B}^T\boldsymbol{U}\right|\boldsymbol{x}\|_\infty}{\|\boldsymbol{x}\|_\infty}.\end{aligned}$$

□

[4] Note that the oblivous routing strategies considered in [139] [158] [225] are all linear oblivious routing strategies.

To make this lemma easily applicable in a variety of settings, we make use of the following easy-to-prove lemma.

Lemma 12.4.4 (Operator Norm Bounds). *For all $\boldsymbol{A} \in \mathbb{R}^{n \times m}$, we have that*

$$\|\boldsymbol{A}\|_\infty = \| \, |\boldsymbol{A}| \, \|_\infty = \| \, |\boldsymbol{A}| \, \mathbf{1}\|_\infty = \max_{i \in n} \|\boldsymbol{A}^T \mathbf{1}_i\|_1.$$

The previous two lemmas make the connection between oblivious routings and circulation projection matrices clear. Below, we prove it formally.

Lemma 12.4.5 (Oblivious Routing to Circulation Projection). *For oblivious routing $\boldsymbol{A} \in \mathbb{R}^{E \times V}$ the matrix $\widetilde{\boldsymbol{P}} \stackrel{\text{def}}{=} \boldsymbol{I} - \boldsymbol{A}\boldsymbol{B}^T$ is a circulation projection matrix such that $\|\boldsymbol{U}^{-1}\widetilde{\boldsymbol{P}}\boldsymbol{U}\|_\infty \leq 1 + \rho(\boldsymbol{A})$.*

Proof. First, we verify that $\text{im}(\widetilde{\boldsymbol{P}})$ is contained in cycle space:

$$\forall \boldsymbol{x} \in \mathbb{R}^E \quad : \quad \boldsymbol{B}^T \widetilde{\boldsymbol{P}} \boldsymbol{x} = \boldsymbol{B}^T \boldsymbol{x} - \boldsymbol{B}^T \boldsymbol{A} \boldsymbol{B}^T \boldsymbol{x} = \mathbf{0}.$$

Next, we check that $\widetilde{\boldsymbol{P}}$ is the identity on cycle space

$$\forall \boldsymbol{x} \in \mathbb{R}^E \text{ s.t. } \boldsymbol{B}^T \boldsymbol{x} = \mathbf{0} \quad : \quad \widetilde{\boldsymbol{P}} \boldsymbol{x} = \boldsymbol{x} - \boldsymbol{A}\boldsymbol{B}^T \boldsymbol{x} = \boldsymbol{x}.$$

Finally, we bound the ℓ_∞-norm of the scaled projection matrix:

$$\|\boldsymbol{U}^{-1}\widetilde{\boldsymbol{P}}\boldsymbol{U}\|_\infty = \|\boldsymbol{I} - \boldsymbol{U}^{-1}\boldsymbol{A}\boldsymbol{B}^T\boldsymbol{U}\|_\infty \leq 1 + \rho(\boldsymbol{A}).$$

□

■ 12.4.2 A Recursive Approach by Embeddings

We construct an oblivious routing for a graph recursively. Given a generic, possibly complicated, graph, we show how to reduce computing an oblivious routing on this graph to computing an oblivious routing on a simpler graph on the same vertex set. A crucial concept in these constructions will be the notion of an embedding, which will allow us to relate the competitive ratios of an oblivious routing algorithms over graphs on the same vertex sets but different edge sets.

Definition 12.4.6 (Embedding). Let $G = (V, E, \boldsymbol{\mu})$ and $G' = (V, E', \boldsymbol{\mu}')$ denote two undirected capacitated graphs on the same vertex set with incidence matrices $\boldsymbol{B} \in \mathbb{R}^{E \times V}$ and $\boldsymbol{B}' \in \mathbb{R}^{E' \times V}$ respectively. An *embedding* from G to G' is a matrix $\boldsymbol{M} \in \mathbb{R}^{E' \times E}$ such that $\boldsymbol{B}'^T \boldsymbol{M} = \boldsymbol{B}^T$.

In other words, an embedding is a map from flows in one graph G to flows in another graph G' that preserves the demands met by the flow. We can think of an embedding as a way of routing any flow in graph G into graph G' that has the same vertex set, but different edges. We will be particularly interested in embeddings that increase the congestion of the flow by a small amount going from G to G'.

Definition 12.4.7 (Embedding Congestion). Let $\boldsymbol{M} \in \mathbb{R}^{E' \times E}$ be an embedding from $G = (V, E, \boldsymbol{\mu})$ to $G' = (V, E', \boldsymbol{\mu}')$ and let $\boldsymbol{U} \in \mathbb{R}^{E \times E}$ and $\boldsymbol{U}' \in \mathbb{R}^{E' \times E'}$ denote the capacity matrices of G and G' respectively. The *congestion* of embedding $\boldsymbol{M}$ is given by

$$\text{cong}(\boldsymbol{M}) \stackrel{\text{def}}{=} \max_{\boldsymbol{x} \in \mathbb{R}^E} \frac{\|\boldsymbol{U}'^{-1}\boldsymbol{M}\boldsymbol{x}\|_\infty}{\|\boldsymbol{U}^{-1}\boldsymbol{x}\|_\infty} = \|\boldsymbol{U}'^{-1}|\boldsymbol{M}|\boldsymbol{U}\mathbf{1}\|_\infty \quad .$$

We say G *embeds into* G' with congestion α if there exists an embedding $\boldsymbol{M}$ from G to G' such that $\text{cong}(\boldsymbol{M}) \leq \alpha$.

Embeddings potentially allow us to reduce computing an oblivious routing in a complicated graph to computing an oblivious routing in a simpler graph. Specifically, if we can embed a complicated graph in a simpler graph and we can efficiently embed the simple graph in the original graph, both with low congestion, then we can just focus on constructing oblivious routings in the simpler graph. We prove this formally as follows.

Lemma 12.4.8 (Embedding Lemma). *Let $G = (V, E, \boldsymbol{\mu})$ and $G' = (V, E', \boldsymbol{\mu}')$ denote two undirected capacitated graphs on the same vertex sets, let $\boldsymbol{M} \in \mathbb{R}^{E' \times E}$ denote an embedding from G into G', let $\boldsymbol{M}' \in \mathbb{R}^{E \times E'}$ denote an embeding from G' into G, and let $\boldsymbol{A}' \in \mathbb{R}^{E' \times V}$ denote an oblivious routing algorithm on G'. Then $\boldsymbol{A} \stackrel{\text{def}}{=} \boldsymbol{M}'\boldsymbol{A}'$ is an oblivious routing algorithm on G and*

$$\rho(\boldsymbol{A}) \leq \operatorname{cong}(\boldsymbol{M}) \cdot \operatorname{cong}(\boldsymbol{M}') \cdot \rho(\boldsymbol{A}').$$

Proof. For all $\boldsymbol{x} \in \mathbb{R}^V$ we have by definition of embeddings and oblivious routings that

$$\boldsymbol{B}^T\boldsymbol{A}\boldsymbol{x} = \boldsymbol{B}^T\boldsymbol{M}'\boldsymbol{A}'\boldsymbol{x} = \boldsymbol{B}^T\boldsymbol{x}.$$

To bound $\rho(A)$, we let $\boldsymbol{U}$ denote the capacity matrix of G and $\boldsymbol{U}'$ denote the capacity matrix of G'. Using Lemma 12.4.3, we get

$$\rho(\boldsymbol{A}) = \|\boldsymbol{U}^{-1}\boldsymbol{A}\boldsymbol{B}^T\boldsymbol{U}\|_\infty = \|\boldsymbol{U}^{-1}\boldsymbol{M}'\boldsymbol{A}'\boldsymbol{B}^T\boldsymbol{U}\|_\infty$$

Using that $\boldsymbol{M}$ is an embedding and therefore $\boldsymbol{B}'^T\boldsymbol{M} = \boldsymbol{B}^T$, we get

$$\rho(\boldsymbol{A}) = \|\boldsymbol{U}^{-1}\boldsymbol{M}'\boldsymbol{A}'\boldsymbol{B}'^T\boldsymbol{M}\boldsymbol{U}\|_\infty \leq \|\boldsymbol{U}^{-1}\boldsymbol{M}'\boldsymbol{U}'\|_\infty \cdot \|\boldsymbol{U}'^{-1}\boldsymbol{A}'\boldsymbol{B}'^T\boldsymbol{U}'\|_\infty \cdot \|\boldsymbol{U}'^{-1}\boldsymbol{M}\boldsymbol{U}\|_\infty$$

By the definition of competitive ratio and congestion, we obtain the result. □

Note how in this lemma we only use the embedding from G to G' to certify the quality of flows in G', we do not actually need to apply this embedding in the reduction.

Using this concept, we construct oblivious routings via recursive application of two techniques. First, in Section 12.5, we show how to take an arbitrary graph $G = (V, E)$ and approximate it by a *sparse graph* $G' = (V, E')$ (i.e. one in which $|E'| = \widetilde{O}(|V|)$) such that flows in G can be routed in G' with low congestion and that there is an $\widetilde{O}(1)$ embedding from G' to G that can be applied in $\widetilde{O}(|E|)$ time. We call such a construction a *flow sparsifiers* and prove the following theorem.

Theorem 12.4.9. *Let $G = (V, E, \boldsymbol{\mu})$ be an undirected capacitated graph. In $\widetilde{O}(|E| \log U)$ time we can construct a graph G' on the same vertex set with at most $\widetilde{O}(|V| \log U)$ edges and capacity ratio at most $U \cdot \operatorname{poly}(|V|)$. Moreover, given an oblivious routing $\boldsymbol{A}'$ on G', in $\widetilde{O}(|E| \log U)$ time we can construct an oblivious routing $\boldsymbol{A}$ on G such that*

$$\mathcal{T}(\boldsymbol{A}) = \widetilde{O}(|E| \log U + \mathcal{T}(\boldsymbol{A}')) \quad \text{and} \quad \rho(\boldsymbol{A}) = \widetilde{O}(\rho(\boldsymbol{A}')).$$

Next, in Section 12.6 we show how to embed a graph into a collection of graphs consisting of trees plus extra edges. Then, we will show how to embed these graphs into better structured graphs consisting of trees plus edges so that by simply removing degree 1 and degree 2 vertices we are left with graphs with fewer vertices. Formally, we prove the following.

Theorem 12.4.10. *Let $G = (V, E, \boldsymbol{\mu})$ be an undirected capacitated graph with capacity ratio U. For all $t > 0$ in $\widetilde{O}(t \cdot |E|)$ time we can compute graphs $G_1, \ldots, G_t$ each with at most $\widetilde{O}(\frac{|E| \log(U)}{t})$ vertices, at most $|E|$ edges, and capacity ratio at most $|V| \cdot U$. Moreover, given oblivious routings $\boldsymbol{A}_i$ for each*

G_i, in $\widetilde{O}(t \cdot |E|)$ time we can compute an oblivious routing $\boldsymbol{A}$ on G such that

$$\mathcal{T}(\boldsymbol{A}) = \widetilde{O}(t \cdot |E| + \sum_{i=1}^{t} \mathcal{T}(\boldsymbol{A}_i)) \quad \text{and} \quad \rho(\boldsymbol{A}) = \widetilde{O}(\max_i \rho(\boldsymbol{A}_i)).$$

In the next section we show that the careful application of these two ideas along with a powerful primitive for routing on constant sized graphs suffices to produce an oblivious routing with the desired properties.

12.4.3 Efficient Oblivious Routing Construction Proof

First, we provide the lemma that will serve as the base case of our recursion. In particular, we show that electric routing can be used to obtain a routing algorithm with constant competitive ratio for constant-size graphs.

Lemma 12.4.11 (Base Case). *Let $G = (V, E, \boldsymbol{\mu})$ be an undirected capacitated graph and let us assign weights to edges so that $\boldsymbol{W} = \boldsymbol{U}^2$. For $\mathcal{L} \stackrel{\text{def}}{=} \boldsymbol{B}^T \boldsymbol{W} \boldsymbol{B}$ we have that $\boldsymbol{A} \stackrel{\text{def}}{=} \boldsymbol{W} \boldsymbol{B} \mathcal{L}^+$ is an oblivious routing on G with $\rho(\boldsymbol{A}) \leq \sqrt{|E|}$ and $\mathcal{T}(\mathcal{L}^+) = \widetilde{O}(|E|)$.*

Proof. To see that $\boldsymbol{A}$ is an oblivious routing strategy we note that for any demands $\boldsymbol{\chi} \in \mathbb{R}^V$ we have $\boldsymbol{B}^T \boldsymbol{A} = \mathcal{L}\mathcal{L}^+ = \boldsymbol{I}$. To see bound $\rho(\boldsymbol{A})$ we note that by Lemma 12.4.3 and standard norm inequalities we have

$$\rho(\boldsymbol{A}) = \max_{\boldsymbol{x} \in \mathbb{R}^E} \frac{\|\boldsymbol{U}^{-1} \boldsymbol{W} \boldsymbol{B} \mathcal{L}^+ \boldsymbol{B}^T \boldsymbol{U} \boldsymbol{x}\|_\infty}{\|\boldsymbol{x}\|_\infty} \leq \max_{\boldsymbol{x} \in \mathbb{R}^E} \frac{\|\boldsymbol{U} \boldsymbol{B} \mathcal{L}^+ \boldsymbol{B}^T \boldsymbol{U} \boldsymbol{x}\|_2}{\frac{1}{\sqrt{|E|}} \|\boldsymbol{x}\|_2} = \sqrt{|E|} \cdot \|\boldsymbol{U} \boldsymbol{B} \mathcal{L}^+ \boldsymbol{B}^T \boldsymbol{U}\|_2$$

The result follows from the fact in [242] that $\boldsymbol{\Pi} \stackrel{\text{def}}{=} \boldsymbol{U} \boldsymbol{B} \mathcal{L}^+ \boldsymbol{B}^T \boldsymbol{U}$ is an orthogonal projection, and therefore $\|\boldsymbol{\Pi}\|_2 \leq 1$, and the fact in [243, 147, 146, 141] that $\mathcal{T}(\mathcal{L}^+) = \widetilde{O}(|E|)$. □

Assuming Theorem 12.4.9 and Theorem 12.4.10, which we prove in the next two sections, we prove that low-congestion oblivious routings can be constructed efficiently.

Theorem 12.4.12. *Given an undirected capacitated graph $G = (V, E, \boldsymbol{\mu})$ with capacity ratio U. Assume $U = \text{poly}(|V|)$. We can construct an oblivious routing algorithm $\boldsymbol{A}$ on G in time*

$$O(|E| 2^{O(\sqrt{\log |V| \log \log |V|})})$$

such that

$$\mathcal{T}(\boldsymbol{A}) = |E| 2^{O(\sqrt{\log |V| \log \log |V|})} \quad \text{and} \quad \rho(\boldsymbol{A}) = 2^{O(\sqrt{\log |V| \log \log |V|})}.$$

Proof. Let c be the constant hidden in the exponent terms, including $\tilde{O}(\cdot)$ and $\text{poly}(\cdot)$ in Theorem 12.4.9 and Theorem 12.4.10. Apply Theorem 12.4.9 to construct a sparse graph $G^{(1)}$, then apply Theorem 12.4.10 with $t = \left\lceil 2^{\sqrt{\log |V| \log \log |V|}} \right\rceil$ to get t graphs $G_1^{(1)}, \cdots G_t^{(1)}$ such that each graphs have at most $O\left(\frac{1}{t} |E| \log^{2c} |V| \log^{2c} U\right)$ vertices and at most $U \cdot |V|^{2c}$ capacity ratio.

Repeat this process on each $G_i^{(1)}$, it produces t^2 graphs $G_1^{(2)}, \cdots, G_{t^2}^{(2)}$. Keep doing this until all graphs G_i produced have $O(1)$ vertices. Let k be the highest level we go through in this process. Since at the k-th level the number of vertices of each graph is at most $O\left(\frac{c^k}{t^k} |E| \log^{2kc} |V| \log^{2k}(U |V|^{2ck})\right)$ vertices, we have $k = O\left(\sqrt{\frac{\log |V|}{\log \log |V|}}\right)$.

On each graph G_i, we use Theorem 12.4.11 to get an oblivious routing algorithm $\boldsymbol{A}_i$ for each G_i with

$$\mathcal{T}(\boldsymbol{A}_i) = \tilde{O}(1) \quad \text{and} \quad \rho(\boldsymbol{A}_i) = \tilde{O}(1).$$

Then, Theorem 12.4.10 and 12.4.9 shows that we have an oblivious routing algorithm $\boldsymbol{A}$ for G with

$$\mathcal{T}(\boldsymbol{A}) = O(tk|E|\log^{ck}(|V|)\log^{2k}(U|V|^{2ck})) \quad \text{and} \quad \rho(\boldsymbol{A}) = O(\log^{2kc}|V|\log^{k}(U|V|^{2ck})).$$

The result follows from $k = O\left(\sqrt{\frac{\log|V|}{\log\log|V|}}\right)$ and $t = \left\lceil 2^{\sqrt{\log|V|\log\log|V|}}\right\rceil$. □

Using Theorem 12.4.12, Lemma 12.4.5 and Theorem 12.3.4, we have the following almost linear time max flow algorithm on undirected graph.

Theorem 12.4.13. *Given an undirected capacitated graph $G = (V, E, \boldsymbol{\mu})$ with capacity ratio U. Assume $U = \text{poly}(|V|)$. There is an algorithm finds an $(1-\varepsilon)$ approximate maximum flow in time*

$$O\left(\frac{|E|2^{O\left(\sqrt{\log|V|\log\log|V|}\right)}}{\varepsilon^2}\right).$$

■ 12.5 Flow Sparsifiers

In order to prove Theorem 12.4.9, i.e. reduce the problem of efficiently computing a competitive oblivious routing on a dense graph to the same problem on a sparse graph, we introduce a new algorithmic tool called *flow sparsifiers*. [5] A flow sparsifier is an efficient cut-sparsification algorithm that also produces an efficiently-computable low-congestion embedding mapping the sparsified graph back to the original graph.

Definition 12.5.1 (Flow Sparsifier). An algorithm is a (h, ε, α)-*flow sparsifier* if on input graph $G = (V, E, \mu)$ with capacity ratio U it outputs a graph $G' = (V, E', \mu')$ with capacity ratio $U' \leq U \cdot \text{poly}(|V|)$ and an embedding $\boldsymbol{M} : \mathbb{R}^{E'} \to \mathbb{R}^{E}$ of G' into G with the following properties:

- **Sparsity:** G' is h-sparse, i.e.
$$|E'| \leq h$$
- **Cut Approximation:** G' is an ε-cut approximation of G, i.e.
$$\forall S \subseteq V \quad : \quad (1-\varepsilon)\mu(\partial_G(S)) \leq \mu'(\partial_{G'}(S)) \leq (1+\varepsilon)\mu(\partial_G(S))$$
- **Flow Approximation:** $\boldsymbol{M}$ has congestion at most α, i.e.
$$\text{cong}(\boldsymbol{M}) \leq \alpha.$$
- **Efficiency:** The algorithm runs in $\tilde{O}(m \log U)$ time and $\mathcal{T}(\boldsymbol{M})$ is also $\tilde{O}(m \log U)$.

Flow sparsifiers allow us to solve a multi-commodity flow problem on a possibly dense graph G by converting G into a sparse graph G' and solving the flow problem on G', while suffering a loss of a factor of at most α in the congestion when mapping the solution back to G using $\boldsymbol{M}$.

[5] Note that our flow sparsifiers aim to reduce the number of edges, and are different from the flow sparsifiers of Leighton and Moitra [167], which work in a different setting and reduce the number of vertices.

Theorem 12.5.2. *Consider a graph $G = (V, E, \mu)$ and let $G' = (V, E', \mu')$ be given by an (h, ε, α)-flow sparsifier of G. Then, for any set of k demands $D = \{\boldsymbol{\chi}_1, \boldsymbol{\chi}_2, \ldots, \boldsymbol{\chi}_k\}$ between vertex pairs of V, we have:*

$$\mathrm{OPT}_{G'}(D) \leq \frac{O(\log k)}{1-\varepsilon} \cdot \mathrm{OPT}_G(D). \tag{12.2}$$

Given the optimum flow $\{f_i^\star\}$ over G', we have

$$\mathrm{cong}_G(\{\boldsymbol{M} f_i^\star\}) \leq \alpha \cdot \mathrm{OPT}_{G'}(D) \leq \frac{O(\alpha \log k)}{1-\varepsilon} \cdot \mathrm{OPT}_G(D).$$

Proof. By the flow-cut gap theorem of Aumann and Rabani [21], we have that, for any set of k demands D on V we have:

$$\mathrm{OPT}_G(D) \geq O\left(\frac{1}{\log k}\right) \cdot \max_{S \subset V} \frac{D(\partial(S))}{\mu(\partial_G(S))}.$$

where $D(\partial(S))$ denotes the total amount of demand separated by the cut between S and $\bar{S}$. As any cut $S \subseteq V$ in G' has capacity $\mu'(\partial_{G'}(S)) \geq (1-\varepsilon)\mu(\partial_G(S))$, we have:

$$\mathrm{OPT}_{G'}(D) \leq \max_{S \subset V} \frac{D(\partial(S))}{\mu'(\partial_{G'}(S))} \leq \frac{1}{1-\varepsilon} \cdot \max_{S \subset V} \frac{D(\partial(S))}{\mu(\partial_G(S))} \leq \frac{O(\log k)}{1-\varepsilon} \cdot \mathrm{OPT}_G(D).$$

The second part of the theorem follows as a consequence of the definition of the congestion of the embedding $\boldsymbol{M}$. □

Our flow sparsifiers should be compared with the cut-based decompositions of Räcke [225]. Räcke constructs a probability distribution over trees and gives explicit embeddings from G to this distribution and backwards, achieving a congestion of $O(\log n)$. However, this distribution over tree can include up to $O(m \log n)$ trees and it is not clear how to use it to obtain an almost linear time algorithm. Flow sparsifiers answer this problem by embedding G into a single graph G', which is larger than a tree, but still sparse. Moreover, they provide an explicit efficient embedding of G' into G. Interestingly, the embedding from G to G' is not necessary for our notion of flow sparsifier, and is replaced by the cut-approximation guarantee. This requirement, together with the application of the flow-cut gap [21], lets us argue that the optimal congestion of a k-commodity flow problem can change at most by a factor of $O(\log k)$ between G and G'.

Main Theorem on Flow Sparsifiers and Proof of Theorem 12.4.9

The main goal of this section will be to prove the following theorem:

Theorem 12.5.3. *For any constant $\varepsilon \in (0, 1)$, there is an $(\tilde{O}(n \log U), \varepsilon, \tilde{O}(1))$-flow sparsifier.*

Assuming Theorem 12.5.3, we can now prove Theorem 12.4.9, the main theorem necessary for edge reduction in our construction of low-congestion projections.

Proof of Theorem 12.4.9. We apply the flow sparsifier of Theorem 12.5.3 to $G = (V, E, \boldsymbol{\mu})$ and obtain output $G' = (V, E', \boldsymbol{\mu}')$ with embedding $\boldsymbol{M}$. By the definition of flow sparsifier, we know that the capacity ratio U' of G' is at most $U \cdot \mathrm{poly}(|V|)$, as required. Moreover, again by Theorem 12.5.3, G' has at most $\tilde{O}(|V| \log U)$ edges. Given an oblivious routing $\boldsymbol{A'}$ on G' consider the oblivious routing $\boldsymbol{A} \stackrel{\text{def}}{=} \boldsymbol{M}\boldsymbol{A'}$. By the definition of flow sparsifier, we have that $\mathcal{T}(\boldsymbol{M}) = \widetilde{O}(|E| \log U)$. Hence $\mathcal{T}(\boldsymbol{A}) = \mathcal{T}(\boldsymbol{M}) + \mathcal{T}(\boldsymbol{A'}) = \widetilde{O}(|E| \log U) + \mathcal{T}(\boldsymbol{A'})$. To complete the proof, we bound the competitive ratio $\rho(\boldsymbol{A})$. Using the same argument as in Lemma 12.4.3, we can write $\rho(\boldsymbol{A})$ as

$$\rho(\boldsymbol{A}) = \max_D \frac{\mathrm{cong}_G(\{\boldsymbol{A}D\})}{\mathrm{OPT}_G(D)} \leq \max_{D_\infty} \frac{\mathrm{cong}_G(\boldsymbol{A}D_\infty)}{\mathrm{OPT}_G(D_\infty)},$$

where D_∞ is the set of demands that result by taking the routing of every demand in D by $\mathrm{OPT}(D)$ and splitting it up into demands on every edge corresponding to the flow sent by $\mathrm{OPT}(D)$. Notice that D_∞ has at most $|E|$ demands that are routed between pairs of vertices in V. Then, because G' is an ε-cut approximation to G, the flow-cut gap of Aumann and Rabani [21] guarantees that

$$\mathrm{OPT}_G(D_\infty) \geq \frac{1}{O(\log n)}\mathrm{OPT}_{G'}(D_\infty).$$

As a result, we obtain:

$$\begin{aligned}\rho(\boldsymbol{A}) &\leq O(\log n)\cdot \max_{D_\infty}\frac{\mathrm{cong}_G(\boldsymbol{A}D_\infty)}{\mathrm{OPT}_{G'}(D_\infty)} = O(\log n)\cdot \max_{D_\infty}\frac{\mathrm{cong}_G(\boldsymbol{M}\boldsymbol{A}'D_\infty)}{\mathrm{OPT}_{G'}(D_\infty)}\\ &\leq O(\log n)\cdot \mathrm{cong}(\boldsymbol{M})\cdot \max_{D_\infty}\frac{\mathrm{cong}_{G'}(\boldsymbol{A}'D_\infty)}{\mathrm{OPT}_{G'}(D_\infty)} \leq \widetilde{O}(\rho(\boldsymbol{A}')).\end{aligned}$$

□

Techniques

We will construct flow sparsifiers by taking as a starting point the construction of spectral sparsifiers of Spielman and Teng [244]. Their construction achieves a sparsity of $\widetilde{O}\left(\frac{n}{\varepsilon^2}\right)$ edges, while guaranteeing an ε-spectral approximation. As the spectral approximation implies the cut approximation, the construction in [244] suffices to meet the first two conditions in Definition 12.5.1. Moreover, their algorithm also runs in time $\tilde{O}(m)$, meeting the fourth condition. Hence, to complete the proof of Theorem 12.5.3, we will modify the construction of Spielman and Teng to endow their sparsifier G' with an embedding $\boldsymbol{M}$ onto G of low congestion that can be both computed and invoked efficiently. The main tool we use in constructing $\boldsymbol{M}$ is the notion of electrical-flow routing and the fact that electrical-flow routing schemes achieve a low competitive ratio on near-expanders and subsets thereof [139, 158].

To exploit this fact and construct a flow sparsifier, we follow Spielman and Teng [244] and partition the input graph into vertex sets, where each sets induces a near-expanders and most edges of the graph do not cross set boundaries. We then sparsify these induced subgraphs using standard sparsification techniques and iterate on the edges not in the subgraphs. As each iteration removes a constant fraction of the edges, by using standard sparsification techniques, we immediately obtain the sparsity and cut approximation properties. To obtain the embedding $\boldsymbol{M}$ with $\mathrm{cong}(\boldsymbol{M}) = \widetilde{O}(1)$, we prove a generalization of results in [139, 158] and show that the electrical-flow routing achieves a low competitive ratio on near-expanders and subsets thereof.

In the next two subsections, we introduce the necessary concept about electrical-flow routing and prove that it achieves low competitive ratio over near-expanders (and subsets of near-expanders).

12.5.1 Subgraph Routing

Given an oblivious routing strategy $\boldsymbol{A}$, we may be interested only in routing demands coming from a subset of edge $F \subseteq E$. In this setting, given a set of demands D routable in F, we let $\mathrm{OPT}^F(D)$ denote the minimal congestion achieved by any routing restricted to only sending flow on edges in F and we measure the F-competitive ratio of $\boldsymbol{A}$ by

$$\rho^F(\boldsymbol{A}) \stackrel{\text{def}}{=} \max_{D \text{ routable in } F}\frac{\mathrm{cong}(\boldsymbol{A}D)}{\mathrm{OPT}^F(D)}$$

Note that $\boldsymbol{A}$ may use all the edges in G but $\rho^F(\boldsymbol{A})$ compares it only against routings that are restricted to use only edges in F. As before, we can upper bound the F-competitive ratio $\rho^F(\boldsymbol{A})$ by operator

norms.

Lemma 12.5.4. *Let $\mathbf{1}_F \in \mathbb{R}^E$ denote the indicator vector for set F (i.e. $\mathbf{1}_F(e) = 1$ if $e \in F$ and $\mathbf{1}_F(e) = 0$) and let $\boldsymbol{I}_F \stackrel{\text{def}}{=} diag(\mathbf{1}_F)$. For any $F \subseteq E$ we have*

$$\rho^F(\boldsymbol{A}) = \|\boldsymbol{U}^{-1}\boldsymbol{A}\boldsymbol{B}^T\boldsymbol{U}\boldsymbol{I}_F\|_\infty$$

Proof. We use the same reasoning as in the non-subgraph case. For a set of demands $D = \{\boldsymbol{\chi}_i\}$, we consider D^F_∞, the demands on the edges in F used by $\text{OPT}^F(D)$. Then, it is the case that $\text{OPT}^F(D) = \text{OPT}^F(D_\infty)$ and we know that cost of obliviously routing D_P is greater than the cost of obliviously routing D. Therefore, we have:

$$\begin{aligned}
\rho^F &= \max_{\boldsymbol{x}\in\mathbb{R}^E \,:\, \boldsymbol{I}_{E\setminus F}\boldsymbol{x}=0} \frac{\| \sum_{e\in E} |\boldsymbol{U}^{-1}\boldsymbol{A}\boldsymbol{B}^T\mathbf{1}_e\boldsymbol{x}_e| \|_\infty}{\|\boldsymbol{U}^{-1}\boldsymbol{x}\|_\infty} \\
&= \max_{\boldsymbol{y}\in\mathbb{R}^E \,:\, \boldsymbol{I}_{E\setminus F}\boldsymbol{y}=0} \frac{\| \sum_{e\in E} |\boldsymbol{U}^{-1}\boldsymbol{A}\boldsymbol{B}^T\boldsymbol{U}\mathbf{1}_e\boldsymbol{y}_e| \|_\infty}{\|\boldsymbol{y}\|_\infty} \\
&= \max_{\boldsymbol{y}\in\mathbb{R}^E} \frac{\| \sum_{e\in E} |\boldsymbol{U}^{-1}\boldsymbol{A}\boldsymbol{B}^T\boldsymbol{U}\boldsymbol{I}_F\mathbf{1}_e\boldsymbol{y}_e| \|_\infty}{\|\boldsymbol{y}\|_\infty} = \| \left|\boldsymbol{U}^{-1}\boldsymbol{A}\boldsymbol{B}^T\boldsymbol{U}\boldsymbol{I}_F\right| \|_\infty = \|\boldsymbol{U}^{-1}\boldsymbol{A}\boldsymbol{B}^T\boldsymbol{U}\boldsymbol{I}_F\|_\infty \\
&\qquad\qquad \text{(Having } y_e \neq 0 \text{ for } e \in E \setminus F \text{ decreases the ratio.)}
\end{aligned}$$

□

12.5.2 Electrical-Flow Routings

In this section, we define the notion of electrical-flow routing and prove the results necessary to construct flow sparsifiers. Recall that $\boldsymbol{R}$ is the diagonal matrix of resistances and the Laplacian $\mathcal{L}$ is defined as $\boldsymbol{B}^T\boldsymbol{R}^{-1}\boldsymbol{B}$. For the rest of this section, we assume that resistances are set as $\boldsymbol{R} = \boldsymbol{U}^{-1}$.

Definition 12.5.5. Consider a graph $G = (V, E, \mu)$ and set the edge resistances as $r_e = \frac{1}{\mu_e}$ for all $e \in E$. The oblivious electrical-flow routing strategy is the linear operator $\boldsymbol{A}_\mathcal{E}$ defined as

$$\boldsymbol{A}_\mathcal{E} \stackrel{\text{def}}{=} \boldsymbol{R}^{-1}\boldsymbol{B}\mathcal{L}^+,$$

In words, the electrical-flow routing strategy is the routing scheme that, for each demand $\boldsymbol{\chi}$ sends the electrical flow with boundary condition $\boldsymbol{\chi}$ on the graph G with resistances $\boldsymbol{R} = \boldsymbol{U}^{-1}$.

For the electrical-flow routing strategy $\boldsymbol{A}_\mathcal{E}$, the upper bound on the competitive ratio $\rho(\boldsymbol{A}_\mathcal{E})$ in Lemma 12.4.3 can be rephrased in terms of the voltages induced on G by electrically routing an edge $e \in E$. This interpretation appears in [139, 158].

Lemma 12.5.6. *Let $\boldsymbol{A}_\mathcal{E}$ be the electrical-flow routing strategy. For an edge $e \in E$, we let the voltage vector $\boldsymbol{v}_e \in \mathbb{R}^V$ be given by $\boldsymbol{v}_e \stackrel{\text{def}}{=} \mathcal{L}^+\boldsymbol{\chi}_e$. We then have*

$$\rho(\boldsymbol{A}_\mathcal{E}) = \max_{e\in E} \sum_{(a,b)\in E} \frac{|v_e(a) - v_e(b)|}{r_{ab}}.$$

Proof. We have:

$$\rho(\boldsymbol{A}_\mathcal{E}) = \|\boldsymbol{B}\mathcal{L}^+\boldsymbol{B}^T\boldsymbol{R}^{-1}\|_\infty = \max_{e\in E} \|\boldsymbol{R}^{-1}\boldsymbol{B}\mathcal{L}^+\boldsymbol{B}^T\mathbf{1}_e\|_1 = \max_{e\in E} \sum_{(a,b)\in E} \frac{|v_e(a) - v_e(b)|}{r_{ab}}.$$

□

The same reasoning can be extended to the subgraph-routing case to obtain the following lemma.

Lemma 12.5.7. *For $F \subseteq E$ and $\boldsymbol{R} = \boldsymbol{U}^{-1}$ we have*

$$\rho^F(\boldsymbol{A}_\mathcal{E}) = \max_{e \in E} \sum_{(a,b)\in F} \frac{|v_e(a) - v_e(b)|}{r_{ab}}.$$

Proof. As before, we have:

$$\begin{aligned}\rho^F(\boldsymbol{A}_\mathcal{E}) &= \|\boldsymbol{B}\mathcal{L}^+\boldsymbol{B}^T\boldsymbol{R}^{-1}\boldsymbol{I}_F\|_\infty && \text{(By Lemma 12.5.4)}\\ &= \max_{e\in E}\|\boldsymbol{I}_F\boldsymbol{R}^{-1}\boldsymbol{B}\mathcal{L}^+\boldsymbol{B}^T\mathbf{1}_e\|_1 = \max_{e\in E}\sum_{(a,b)\in F}\frac{|v_e(a)-v_e(b)|}{r_{ab}}\end{aligned}$$

□

12.5.2.1 Bounding the Congestion

In this section, we prove that we can bound the F-competitive ratio of the oblivious electrical-routing strategy as long as the edges F that the optimum flow is allowed to route over are contained within an induced expander $G(U) = (U, E(U))$ for some $U \subseteq V$. Towards this we provide and prove the following lemma. This is a generalization of a similar lemma proved in [139].

Lemma 12.5.8. *For weighted graph $G = (V, E, w)$ with integer weights and vertex subset $U \subseteq V$ the following holds:*

$$\rho^F(\boldsymbol{A}_\mathcal{E}) \leq \frac{8\log(\mathrm{vol}(G(U)))}{\Phi(G(U))^2}$$

Proof. By Lemma 12.5.7, for every edge $e \in E$,

$$\rho^F(\boldsymbol{A}_\mathcal{E}) \leq \|\boldsymbol{I}_{E(U)}\boldsymbol{R}^{-1}\boldsymbol{B}\mathcal{L}^+\boldsymbol{\chi}_e\|_1$$

Fix any edge $e \in E$ and let $v \stackrel{\text{def}}{=} \mathcal{L}^+\boldsymbol{\chi}_e$. Recall that with this definition

$$\|\boldsymbol{I}_{E(U)}\boldsymbol{R}^{-1}\boldsymbol{B}\mathcal{L}^+\boldsymbol{\chi}_e\|_1 = \sum_{(a,b)\in E(U)}\frac{|v(a)-v(b)|}{r_{ab}} = \sum_{(a,b)\in E(U)} w_{ab}\cdot|v(a)-v(b)| \tag{12.3}$$

We define the following vertex subsets:

$$\forall x \in \mathbb{R} \quad : \quad S_x^{\leq} \stackrel{\text{def}}{=} \{a \in U \mid v(a) \leq x\} \quad \text{and} \quad S_x^{\geq} \stackrel{\text{def}}{=} \{a \in U \mid v(a) \geq x\}$$

Since adding a multiple of the all-ones vector to v does not change the quantity of interest in Equation 12.3, we can assume without loss of generality that

$$\mathrm{vol}_{G(U)}(S_0^{\geq}) \geq \frac{1}{2}(\mathrm{vol}(G(U))) \quad \text{and} \quad \mathrm{vol}_{G(U)}(S_0^{\leq}) \geq \frac{1}{2}(\mathrm{vol}(G(U))).$$

For any vertex subset $S \subseteq U$, we denote the flow out of S and the weight out of S by

$$f(S) \stackrel{\text{def}}{=} \sum_{e=(a,b)\in E(U)\bigcap\partial(S)} w_e|v(a)-v(b)|, \quad \text{and} \quad w(S) \stackrel{\text{def}}{=} \sum_{e\in E(U)\bigcap\partial(S)} w_e.$$

At this point, we define a collections of subsets $\{C_i \in S_0^{\geq}\}$. For an increasing sequence of real numbers

$\{c_i\}$, we let $C_i \stackrel{\text{def}}{=} S^{\geq}_{c_i}$ and we define the sequence $\{c_i\}$ inductively as follows:

$$c_0 \stackrel{\text{def}}{=} 0 \quad , \quad c_i \stackrel{\text{def}}{=} c_{i-1} + \Delta_{i-1} \quad \text{, and} \quad \Delta_i \stackrel{\text{def}}{=} 2\frac{f(C_i)}{w(C_i)} \quad .$$

In words, the c_{i+1} equals the sum of c_i and an increase Δ_i which depends on how much the cut $\delta(C_i) \cap E(U)$ was congested by the electrical flow.

Now, $l_i \stackrel{\text{def}}{=} w(\partial_{E(U)}(C_{i-1}) - \partial_{E(U)}(C_i))$, i.e. the weight of the edges in $E(U)$ cut by C_{i-1} but not by C_i. We get

$$\begin{aligned}
\text{vol}(C_{i+1}) &\leq \text{vol}(C_i) - l_i \\
&\leq \text{vol}(C_i) - \frac{w(C_i)}{2} && \text{(By choice of } l_i \text{ and } \Delta_i\text{)} \\
&\leq \text{vol}(C_i) - \frac{1}{2}\text{vol}(C_i)\Phi(G(U)) && \text{(Definition of conductance)}
\end{aligned}$$

Applying this inductively and using our assumption on $\text{vol}(S^{\geq}_0)$ we have that

$$\text{vol}(C_i) \leq \left(1 - \frac{1}{2}\Phi(G(U))\right)^i \text{vol}(C_0) \leq \frac{1}{2}\left(1 - \frac{1}{2}\Phi(G(U))\right)^i \text{vol}(G(U))$$

Since $\phi(G(U)) \in (0,1)$, for $i+1 = \frac{2\log(\text{vol}(G(U)))}{\Phi(G(U))}$ we have that $\text{vol}(S_i) \leq \frac{1}{2}$. Since $\text{vol}(S_i)$ decreases monotonically with i, if we let r be the smallest value such that $C_{r+1} = \emptyset$, we must have

$$r \leq \frac{2 \cdot \log(\text{vol}(G(U)))}{\Phi(G(U))}$$

Since v corresponds to a unit flow, we know that $f(C_i) \leq 1$ for all i. Moreover, by the definition of conductance we know that $w(C_i) \geq \Phi(G(U)) \cdot \text{vol}(C_i)$. Therefore,

$$\Delta_i \leq \frac{2}{\Phi(G(U)) \cdot \text{vol}(C_i)}.$$

We can now bound the contribution of $C^{\geq}_0$ to the volume of the linear embedding v. In the following, for a vertex $a \in V$, we let $d(a) \stackrel{\text{def}}{=} \sum_{e=\{a,b\} \in E(U)} w_e$ be the degree of a in $E(U)$.

$$\begin{aligned}
\sum_{a \in C^{\geq}_0} d(a)v(a) &= \sum_{i=0}^{r} \left[\sum_{a \in C_i - C_{i+1}} d(a)v(a) \right] \\
&\leq \sum_{i=0}^{r} \left[\sum_{a \in C_i - C_{i+1}} d(a) \left(\sum_{j=0}^{i} \Delta_j \right) \right] && \text{(By definition of } C_i\text{)} \\
&\leq \sum_{i=0}^{r} \left[(\text{vol}(C_i) - \text{vol}(C_{i+1})) \cdot \left(\sum_{j=0}^{i} \Delta_j \right) \right] \\
&= \sum_{i=0}^{r} \text{vol}(C_i)\Delta_i \leq \frac{2r}{\Phi(G(U))} && \text{(Rearrangement and fact that } \text{vol}(C_{r+1}) = 0\text{)}
\end{aligned}$$

By repeating the same argument on $S^{\leq}_0$, we get that $\sum_{a \in S^{\leq}_0} d(a)v(a) \leq \frac{2r}{\Phi(G(U))}$. Putting this all

together yields

$$\|\boldsymbol{I}_{E(U)}\boldsymbol{R}^{-1}\boldsymbol{B}\mathcal{L}^{+}\boldsymbol{\chi}_e\| = \sum_{(a,b)\in G(U)} w_{ab}\cdot|v(a)-v(b)| \leq \sum_{a\in G(U)} d(a)v(a) \leq \frac{4r}{\Phi(G(U))}$$

□

From this lemma and Lemma 12.5.7, the following is immediate:

Lemma 12.5.9. *Let $F \subseteq E$ be contained within some vertex induced subgraph $G(U)$, then for $\boldsymbol{R} = \boldsymbol{U}^{-1}$ we have*

$$\rho^{F}(\boldsymbol{R}^{-1}\boldsymbol{B}\mathcal{L}^{+}) \leq \rho^{E(U)}(\boldsymbol{R}^{-1}\boldsymbol{B}\mathcal{L}^{+}) \leq \frac{8\log(\mathrm{vol}(G(U)))}{\Phi(G(U))^2}.$$

12.5.3 Construction and Analysis of Flow Sparsifiers

In the remainder of this section we show how to produce an efficient $O(\log^c)$-flow sparsifier for some fixed constant c, proving Theorem 12.5.3. In this chapter, we make no attempt to optimize the value of c. For the rest of this section, we again assume that we choose the resistance of an edge to be the the inverse of its capacity, i.e. $\boldsymbol{U} = \boldsymbol{W} = \boldsymbol{R}^{-1}$.

As discussed before, our approach follows closely that of Spielman and Teng [244] to the construction of spectral sparsifiers. The first step of this line of attack is to reduce the problem to the unweighted case.

Lemma 12.5.10. *Given an (h,ε,α)-flow-sparsifier algorithm for unweighted graphs, it is possible to construct an $(h\cdot\log U,\varepsilon,\alpha)$-flow-sparsifier algorithm for weighted graphs $G=(V,E,\mu)$ with capacity ratio U obeying*

$$U = \frac{\max_{e\in E}\mu_e}{\min_{e\in E}\mu_e} = \mathrm{poly}(|V|).$$

Proof. We write each edge in binary so that $G = \sum_{i=0}^{\log U} 2^i G_i$ for some unweighted graphs $\{G_i = (V,E_i\}_{i\in[\log U]}\}$, where $|E_i| \leq m$ for all i. We now apply the unweighted flow-sparsifier to each G_i in turn to obtain graphs $\{G_i'\}$. We let $G' \stackrel{\text{def}}{=} \sum_{i=0}^{\log U} 2^i G_i'$ be the weighted flow-sparsified graph. By the assumption on the unweighted flow-sparsifier, each G_i' is h-sparse, so that G' must have at most $h\cdot\log U$ edges. Similarly, G' is an ε-cut approximation of G, as each G_i' is an ε-cut approximation of the corresponding G_i. Letting $\boldsymbol{M}_i$ be the embedding of G_i' into G_i, we can consider the embedding $\boldsymbol{M} = \sum_{i=0}^{\log U} 2^i\boldsymbol{M}_i$ of G' into G. As each $\boldsymbol{M_i}$ has congestion bounded by α, it must be the case that $\boldsymbol{M}$ also has congestion bounded by α. The time to run the weighted flow sparsifier and to invoke $\boldsymbol{M}$ is now $\tilde{O}(m)\cdot\log U = \tilde{O}(m)$ by our assumption on U. □

The next step is to construct a routine which flow-sparsifies a constant fraction of the edges of E. This routine will then be applied iteratively to produce the final flow-sparsifier.

Lemma 12.5.11. *On input an unweighted graph $G=(V,E)$, there is an algorithm that runs in $\tilde{O}(m)$ and computes a partition of E into $(F,\bar{F})$, an edge set $F'\subseteq F$ with weight vector $w_{F'}^{\rightarrow}\in\mathbb{R}^E, \mathrm{support}(w_{F'}) = F'$, and an embedding $\boldsymbol{H}:\mathbb{R}^{F'}\to\mathbb{R}^E$ with the following properties:*

1. *F contains most of the volume of G, i.e.*

$$|F| \geq \frac{|E|}{2};$$

2. *F' contains only $\tilde{O}(n)$ edges, i.e. $|F'|\leq\tilde{O}(n)$.*

3. *The weights $w_{F'}$ are bounded*

$$\forall e \in F' \quad , \quad \frac{1}{\mathrm{poly}(n)} \leq w_{F'}(e) \leq n.$$

4. *The graph $H' = (V, F', w_{F'})$ is an ε-cut approximation to $H = (V, F)$, i.e. for all $S \subseteq V$:*

$$(1-\varepsilon)|\partial_H(S)| \leq w_{F'}(\partial_{H'}(S)) \leq (1+\varepsilon)|\partial_H(S)|.$$

5. *The embedding $\boldsymbol{H}$ from $H = (V, F', w_{F'})$ to G has bounded congestion*

$$\mathrm{cong}(\boldsymbol{H}) = \widetilde{O}(1).$$

and can be applied in time $\tilde{O}(m)$.

Given Lemma 12.5.10 and Lemma 12.5.11, it is straightforward to complete the proof of Theorem 12.5.3.

Proof. Using Lemma 12.5.10, we reduce the objective of Theorem 12.5.3 to running a $(\widetilde{O}(n), \varepsilon, \widetilde{O}(1))$-flow sparsifier on $\log U$ unweighted graphs. To construct this unweighted flow sparsifier, we apply Lemma 12.5.11 iteratively as follows. Starting with the instance unweighted graph $G_1 = (V, E_1)$, we run the algorithm of Lemma 12.5.11 on the current graph $G_t = (V, E_t)$ to produce the sets F_t and F'_t, the weight vector $w_{F'_t}$ and the embedding $\boldsymbol{H_t} : \mathbb{R}^{F'_t} \to R^E$. To proceed to the next iteration, we then define $E_{t+1} \stackrel{\text{def}}{=} E_t \setminus F_t$ and move on to G_{t+1}.

By Lemma 12.5.11, at every iteration t, $|F_t| \geq \frac{1}{2} \cdot |E_t|$, so that $|E_{t+1}| \leq \frac{1}{2} \cdot |E_t|$. This shows that there can be at most $T \leq \log(|E_1|) = O(\log n)$ iterations.

After the last iteration T, we have effectively partitioned E_1 into disjoint subsets $\{F_t\}_{t\in[T]}$, where each F_t is well-approximated but the weighted edgeset F'_t. We then output the weighted graph $G' = (V, E' \stackrel{\text{def}}{=} \cap_{t=1}^{T} F'_t, w' \stackrel{\text{def}}{=} \sum_{t=1}^{T} w_{F'_t})$, which is the sum of the disjoint weighted edges sets $\{F'_t\}_{t\in[T]}$. We also output the embedding $\boldsymbol{M} : \mathbb{R}^{E'} \to \mathbb{R}^E$ from G' to G, defined as the direct sum

$$\boldsymbol{M} = \bigoplus_{t=1}^{T} \boldsymbol{H_t}.$$

In words, $\boldsymbol{M}$ maps an edge $e' \in E'$ by finding t for which $e' \in F'_t$ and applying the corresponding $\boldsymbol{H_t}$.

We are now ready to prove that this algorithm with output G' and $\boldsymbol{M}$ is an efficient $(\widetilde{O}(n), \varepsilon, \widetilde{O}(n))$-flow sparsifier. To bound the capacity ratio U' of G', we notice that

$$U' \leq \max_t \frac{\max_{e\in F'_t} w_{F'_t}(e)}{\min_{e\in F'_t} w_{F'_t}(e)} \leq \mathrm{poly}(n),$$

where we used the fact that the sets F'_t are disjoint and the guarantee on the range of $w_{F'_t}$.

Next, we bound the sparsity of G'. By Lemma 12.5.11, F'_t contains at most $\widetilde{O}(n)$ edges. As a result, we get the required bound

$$|E'| = \sum_{t=1}^{T} |F'_t| \leq \widetilde{O}(Tn) = \widetilde{O}(n).$$

For the cut approximation, we consider any $S \subseteq V$. By the cut guarantee of Lemma 12.5.11, we have that, for all $t \in [T]$,

$$(1-\varepsilon)|\partial_G(S) \cap F_t| \leq w_{F'_t}(\partial_G(S) \cap F'_t) \leq (1+\varepsilon)|\partial_G(S) \cap F_t|.$$

Summing over all t, as $E' = \dot{\bigcup} F'_t$ and $E = \dot{\bigcup} F_t$, we obtain the required approximation

$$(1-\varepsilon)|\partial_G(S)| \leq w'(\partial_{G'}(S)) \leq (1+\varepsilon)|\partial_G(S)|.$$

The congestion of $\boldsymbol{M}$ can be bounded as follows

$$\mathrm{cong}(\boldsymbol{M}) \leq \sum_{t=1}^{T} \mathrm{cong}(\boldsymbol{H_t}) = \widetilde{O}(T) = \widetilde{O}(1).$$

To conclude the proof, we address the efficiency of the flow sparsifier. The algorithm applies the routine of Lemma 12.5.11 for $T = \widetilde{O}(1)$ times and hence runs in time $\widetilde{O}(m)$, as required. Invoking the embedding $\boldsymbol{M}$ requires invoking each of the T embeddings $\boldsymbol{H_t}$. This takes time $\widetilde{O}(Tm) = \widetilde{O}(m)$. □

12.5.3.1 Flow Sparsification of Unweighted Graphs: Proof of Lemma 12.5.11

In this subsection, we prove Lemma 12.5.11. Our starting point is the following decomposition statement, which shows that we can form a partition of an unweighted graph where most edges do not cross the boundaries and the subgraphs induced within each set of this partition are near-expanders. The following lemma is implicit in Spielman and Teng's local clustering approach to spectral sparsification [244].

Lemma 12.5.12 (Decomposition Lemma). *For an unweighted graph $G = (V, E)$, in $\tilde{O}(m)$-time we can produce a partition $V_1, \ldots, V_k$ of V and a collection of sets $S_1, \ldots, S_k \subseteq V$ with the following properties:*

- *For all i, S_i is contained in V_i.*
- *For all i, there exists a set T_i with $S_i \subseteq T_i \subseteq V_i$, such that*

$$\Phi(G(T_i)) \geq \Omega\left(\frac{1}{\log^2 n}\right).$$

- *At least half of the edges are found within the sets $\{S_i\}$, i.e.*

$$\sum_{i=1}^{k} |E(S_i)| = \sum_{i=1}^{k} |\{e = \{a, b\} : a \in S_i, b \in S_i\}| \geq \frac{1}{2}|E|.$$

To design an algorithm satisfying the requirements of Lemma 12.5.11, we start by appling the Decomposition Lemma to our unweighted input graph $G = (V, E)$ to obtain the partition $\{V_i\}_{i\in[k]}$ and the sets $\{S_i\}_{i\in[k]}$. We let $G_i \stackrel{\text{def}}{=} (S_i, E(S_i))$. To reduce the number of edges, while preseving cuts, we apply a spectral sparsification algorithm to each G_i. Concretely, by applying the spectral sparsification by effective resistances of Spielman and Srivastava [242] to each G_i, we obtain weighted graphs $G'_i = (S_i, E'_i \subseteq E(S_i), w'_i)$ in time $\sum_{i=1}^{k} \widetilde{O}(|E(S_i)|) \leq \widetilde{O}(|E|)$ with $|E'_i| \leq \widetilde{O}(|S_i|)$ and the property that cuts are preserved[6] for all i:

$$\forall S \subseteq S_i \quad , \quad (1-\varepsilon) \cdot |\partial_{G_i}(S)| \leq w'_i(\partial_{G'_i}(S)) \leq (1+\varepsilon) \cdot |\partial_{G_i}(S)|.$$

Moreover, the spectral sparsification of [242] constructs the weights $\{w'_i(e)\}_{e\in E'_i}$ such that

$$\forall e \in E'_i \quad , \quad \frac{1}{\mathrm{poly}(n)} \leq \frac{1}{\mathrm{poly}(|S_i|)} \leq w'_i(e) \leq |S_i| \leq n.$$

[6] The spectral sparsification result actually yields the stronger spectral approximation guarantee, but for our purposes the cut guarantee suffices.

To complete the description of the algorithm, we output the partition $(F, \bar{F})$ of E, where

$$F \stackrel{\text{def}}{=} \bigcup_{i=1}^{k} E(S_i).$$

We also output the set of weighted sparsified edges F'.

$$F' \stackrel{\text{def}}{=} \bigcup_{i=1}^{k} E'_i.$$

The weight $w_{F'}(e)$ of edge $e \in F'$ is given by finding i such that $e \in E'_i$ and setting $w_{F'}(e) = w'_i(e)$.

We now depart from Spielman and Teng's construction by endowing our F' with an embedding onto G. The embedding $\boldsymbol{H} : \mathbb{R}^{F'} \to \mathbb{R}^{E}$ of the graph $H = (V, F', w_{F'})$ to G is constructed by using the oblivious electrical-flow routing of $E(S_i)$ into $G(V_i)$. More specifically, as the sets $\{V_i\}$ partition V, the embedding $\boldsymbol{H}$ can be expressed as the following direct sum over the orthogonal subspaces $\{\mathbb{R}^{E(V_i)\times E'_i}\}$.

$$\boldsymbol{H} \stackrel{\text{def}}{=} \left(\bigoplus_{i=1}^{k} \boldsymbol{B}_{E(V_i)} \mathcal{L}^{+}_{G(V_i)} \boldsymbol{B}^{T}_{E(V_i)} \boldsymbol{I}_{(E(V_i),E'_i)} \right),$$

where $\boldsymbol{I}_{(E(V_i),E'_i)}$ is the identity mapping of the edges $E'_i \subseteq E(V_i)$ of F' over V_i to the edges $E(V_i)$ of V_i in G. Notice that there is no dependence on the resistances over G as G is unweighted.

This complete the description of the algorithm. We are now ready to give the proof of Lemma 12.5.11.

[Proof of Lemma 12.5.11]. The algorithm described above performs a decomposition of the input graph $G = (V, E)$ in time $\widetilde{O}(m)$ by the Decomposition Lemma. By the result of Spielman and Srivastava [242], each G_i is sparsified in time $\widetilde{O}(|E(S_i)|)$. Hence, the sparsification step requires time $\widetilde{O}(m)$ as well. This shows that the algorithm runs in $\widetilde{O}(m)$-time, as required.

By the Decomposition Lemma, we know that $|F| = \sum_{i=1}^{k} |E(S_i)| \geq \frac{|E|}{2}$, which satisfies the requirement of the Lemma. Moreover, by the spectral sparsification result, we know that $|F'| = \sum_{i=1}^{k} |E'_i| \leq \sum_{i=1}^{k} \widetilde{O}(|S_i|) \leq \widetilde{O}(n)$, as required. We also saw that by construction the weights $w_{F'}$ are bounded:

$$\forall e \in F' \quad , \quad \frac{1}{\text{poly}(n)} \leq w_{F'}(e) \leq n.$$

To obtain the cut-approximation guarantee, we use the fact that for every i, by spectral sparsification,

$$\forall S \subseteq S_i \quad , \quad (1-\varepsilon) \cdot |\partial_{G_i}(S)| \leq w'_i(\partial_{G'_i}(S)) \leq (1+\varepsilon) \cdot |\partial_{G_i}(S)|.$$

We have $H' = (V, F', w_{F'})$ and $H = (V, F)$. Consider now $T \subseteq V$ and apply the previous bound to $T \cap S_i$ for all i. Because $F' \subseteq F = \cup_{i=1}^{k} E(S_i)$, we have that summing over the k bounds yields

$$\forall T \subseteq V \quad , \quad (1-\varepsilon)|\partial_H(T)| \leq w_{F'}(\partial_{H'}(T)) \leq (1+\varepsilon)|\partial_H(T)|,$$

which is the desired cut-approximaton guarantee.

Finally, we are left to prove that the embedding $\boldsymbol{H}$ from $H' = (V, F', w_{F'})$ to $G = (V, E)$ has low congestion and can be applied efficiently. By definition of congestion,

$$\text{cong}(\boldsymbol{H}) = \max_{\boldsymbol{x} \in \mathbb{R}^{F'}} \frac{\|\boldsymbol{H}\boldsymbol{x}\|_\infty}{\|\boldsymbol{U}^{-1}_{F'}\boldsymbol{x}\|_\infty} = \||\boldsymbol{H}|\boldsymbol{U}_{F'}\boldsymbol{1}_{F'}\|_\infty = \left\| \left| \bigoplus_{i=1}^{k} \boldsymbol{B}_{E(V_i)} \mathcal{L}^{+}_{G(V_i)} \boldsymbol{B}^{T}_{E(V_i)} \boldsymbol{I}_{(E(V_i),E'_i)} \right| \boldsymbol{U}_{F'}\boldsymbol{1}_{F'} \right\|_\infty .$$

Decomposing $\mathbb{R}^E$ into the subspaces $\{\mathbb{R}^{E(V_i)}\}$ and $\mathbb{R}^{F'}$ into the subspaces $\{\mathbb{R}^{E'_i}\}$ we have:

$$\operatorname{cong}(\boldsymbol{H}) \leq \max_{i\in[k]} \left\| \left| \boldsymbol{B}_{E(V_i)} \mathcal{L}^{+}_{G(V_i)} \boldsymbol{B}^{T}_{E(V_i)} \boldsymbol{I}_{(E(V_i),E'_i)} \right| \boldsymbol{U}_{E'_i} \mathbf{1}_{E'_i} \right\|_{\infty}.$$

For each $i \in [k]$, consider now the set of demands D_i over V_i, $D_i \stackrel{\text{def}}{=} \{\boldsymbol{\chi}_e\}_{e\in E'_i}$, given by the edges of E'_i with their capacities w'_i. That is, $\boldsymbol{\chi}_e \in \mathbb{R}^{V_i}$ is the demand corresponding to edge $e \in E'_i$ with weight $w'_i(e)$. Consider also the electrical routing $\boldsymbol{A}_{\mathcal{E},i} = \boldsymbol{B}_{E(V_i)}\mathcal{L}^{+}_{G(V_i)}$ over $G(V_i)$. Then:

$$\operatorname{cong}(\boldsymbol{H}) \leq \max_{i\in[k]} \operatorname{cong}(\boldsymbol{A}_{\mathcal{E},i} D_i)$$

Notice that, by construction, D_i is routable in $G'_i = (S_i, E'_i, w'_i)$ and $\operatorname{OPT}_{G'_i}(D_i) = 1$. But, by our use of spectral sparsifiers in the construction, G'_i is an ε-cut approximation of G_i. Hence, by the flow-cut gap of Aumann and Rabani [21], we have:

$$\operatorname{OPT}_{G_i}(D_i) \leq O(\log(|D_i|)) \cdot \operatorname{OPT}_{G'_i}(D_i) \leq \widetilde{O}(1).$$

When we route D_i oblivious in $G(V_i)$, we can consider the $E(S_i)$-competitive ratio $\rho^{E(S_i)}(\boldsymbol{A}_{\mathcal{E},i})$ of the electrical routing $\boldsymbol{A}_{\mathcal{E},i} = \boldsymbol{B}_{E(V_i)}\mathcal{L}^{+}_{G(V_i)}$, as D_i is routable in $E(S_i)$, because $E'_i \subseteq E(S_i)$. We have

$$\operatorname{cong}(\boldsymbol{H}) \leq \max_{i\in[k]} \rho^{E(S_i)}_{G(V_i)}(\boldsymbol{A}_{\mathcal{E},i}) \cdot \operatorname{OPT}^{E(S_i)}_{G(V_i)}(D_i) = \max_{i\in[k]} \rho^{E(S_i)}_{G(V_i)}(\boldsymbol{A}_{\mathcal{E},i}) \cdot \operatorname{OPT}_{G_i}(D_i),$$

Finally, putting these bounds together, we have:

$$\operatorname{cong}(\boldsymbol{H}) \leq \max_{i\in[k]} \rho^{E(S_i)}_{G(V_i)}(\boldsymbol{A}_{\mathcal{E},i}) \cdot \operatorname{OPT}_{G_i}(D_i) \leq \widetilde{O}(1) \cdot \max_{i\in[k]} \rho^{E(S_i)}_{G(V_i)}(\boldsymbol{A}_{\mathcal{E},i}).$$

But, by the Decomposition Lemma, there exists T_i with $S_i \subseteq T_i \subseteq V_i$ such that

$$\Phi(G(T_i)) \geq \Omega\left(\frac{1}{\log^2 n}\right).$$

Then, by Lemma 12.5.9, we have that:

$$\rho^{E(S_i)}_{G(V_i)}(\boldsymbol{A}_{\mathcal{E},i}) \leq O\left(\frac{\log \operatorname{vol}(G(T_i))}{\Phi(G(T_i))^2}\right) \leq \widetilde{O}(1).$$

This concludes the proof that $\operatorname{cong}(\boldsymbol{H}) \leq \widetilde{O}(1)$. To complete the proof of the Lemma, we just notice that $\boldsymbol{H}$ can be invoked in time $\widetilde{O}(m)$. A call of $\boldsymbol{H}$ involves solving k-electrical-problems, one for each $G(V_i)$. This can be done in time $\sum_{i=1}^{k} \widetilde{O}(|E(V_i)|) \leq \widetilde{O}(m)$, using any of the nearly-linear Laplacian system solvers available, such as [141]. □

12.6 Removing Vertices in Oblivious Routing Construction

In this section we show how to reduce computing an efficient oblivious routing on a graph $G = (V, E)$ to computing an oblivious routing for t graphs with $\widetilde{O}(\frac{|V|}{t})$ vertices and at most $|E|$ edges. Formally we show

Theorem 12.6.1. *Let $G = (V, E, \boldsymbol{\mu})$ be an undirected capacitated graph with capacity ratio U. For all $t > 0$ in $\widetilde{O}(t \cdot |E|)$ time we can compute graphs $G_1, \ldots, G_t$ each with at most $\widetilde{O}(\frac{|E|\log(U)}{t})$ vertices, at most $|E|$ edges, and capacity ratio at most $|V| \cdot U$, such that given oblivious routings $\boldsymbol{A}_i$ for each*

G_i, in $\widetilde{O}(t \cdot |E|)$ time we can compute an oblivious routing $\boldsymbol{A} \in \mathbb{R}^{E \times V}$ on G such that

$$\mathcal{T}(\boldsymbol{A}) = \widetilde{O}\left(t \cdot |E| + \sum_{i=1}^{t} \mathcal{T}(\boldsymbol{A}_i)\right) \quad \text{and} \quad \rho(\boldsymbol{A}) = \widetilde{O}\left(\max_i \rho(\boldsymbol{A}_i)\right)$$

We break this proof into several parts. First we show how to embed G into a collection of t graphs consisting of trees minus some edges which we call *patrial tree embeddings* (Section 12.6.1). Then we show how to embed a partial tree embedding in an "almost j-tree" [182], that is a graph consisting of a tree and a subgraph on at most j vertices, for $j = 2t$ (Section 12.6.2). Finally, we show how to reduce oblivious routing on an almost j-tree to oblivious routing on a graph with at most $O(j)$ vertices by removing degree-1 and degree-2 vertices (Section 12.6.3). Finally, in Section 12.6.4 we put this all together to prove Theorem 12.4.10.

We remark that much of the ideas in the section were either highly influenced from [182] or are direct restatements of theorems from [182] adapted to our setting. We encourage the reader to look over that paper for further details regarding the techniques used in this section.

12.6.1 From Graphs to Partial Tree Embeddings

To prove Theorem 12.4.10, we make heavy use of spanning trees and various properties of them. In particular, we use the facts that for every pair of vertices there is a unique *tree path* connecting them, that every edge in the tree *induces a cut* in the graph, and that we can embed a graph in a tree by simply routing ever edge over its tree path and that the congestion of this embedding will be determined by the *load* the edges place on tree edges. We define these quantities formally below.

Definition 12.6.2 (Tree Path)**.** For undirected graph $G = (V, E)$, spanning tree T, and all $a, b \in V$ we let $P_{a,b} \subseteq E$ denote the unique path from a to b using only edges in T and we let $\boldsymbol{p}_{a,b} \in \mathbb{R}^E$ denote the vector representation of this path corresponding to the unique vector sending one one unit from a to b that is nonzero only on T (i.e. $\boldsymbol{B}^T \boldsymbol{p}_{a,b} = \boldsymbol{\chi}_{a,b}$ and $\forall e \in E \setminus T$ we have $\boldsymbol{p}_{a,b}(e) = 0$).

Definition 12.6.3 (Tree Cuts)**.** For undirected $G = (V, E)$ and spanning tree $T \subseteq E$ the *edges cut by* e, $\partial_T(F)$, and the edges cut by F, $\partial_T(e)$, are given by

$$\partial_T(e) \stackrel{\text{def}}{=} \{e' \in E \mid e' \in P_e\} \quad \text{and} \quad \partial_T(F) \stackrel{\text{def}}{=} \cup_{e \in F} \partial(e)$$

Definition 12.6.4 (Tree Load)**.** For undirected capacitated $G = (V, E, \boldsymbol{\mu})$ and spanning tree $T \subseteq E$ the load on edge $e \in E$ by T, $\text{cong}_T(e)$ is given by $\text{load}_T(e) = \sum_{e' \in E | e \in P_{e'}} \boldsymbol{\mu}_{e'}$

While these properties do highlight the fact that we could just embed our graph into a collection of trees to simplify the structure of our graph, this approach suffers from a high computational cost [225]. Instead we show that we can embed parts of the graph onto collections of trees at a lower computational cost but higher complexity. In particular we will consider what we call partial tree embeddings.

Definition 12.6.5 ([Partial Tree Embedding [7]])**.** For undirected capacititated graph $G = (V, E, \boldsymbol{\mu})$, spanning tree T and spanning tree subset $F \subseteq T$ we define the *partial tree embedding graph* $H = H(G, T, F) = (V, E', \boldsymbol{\mu}')$ to a be a graph on the same vertex set where $E' = T \cup \partial_T(F)$ and

$$\forall e \in E' \quad : \quad \boldsymbol{\mu}'(e) = \begin{cases} \text{load}_T(e) & \text{if } e \in T \setminus F. \\ \boldsymbol{\mu}(e) & \text{otherwise} \end{cases}$$

[7] This is a restatement of the $H(T, F)$ graphs in [182].

Furthermore, we let $\boldsymbol{M}_H \in \mathbb{R}^{E' \times E}$ denote the embedding from G to $H(G, T, F)$ where edges not cut by F are routed over the tree and other edges are mapped to themselves.

$$\forall e \in E \quad : \quad \boldsymbol{M}_H(e) = \begin{cases} \boldsymbol{p}_e & e \notin \partial_T(F) \\ \mathbf{1}_e & \text{otherwise} \end{cases}$$

and we let $\boldsymbol{M}'_H \in \mathbb{R}^{E \times E'}$ denote the embeding from H to G that simply maps edges in H to their corresponding edges in G, i.e. $\forall e \in E'$, $\boldsymbol{M}'_H(e) = \mathbf{1}_e$.

Note that by definition $\text{cong}(\boldsymbol{M}_H) \leq 1$, i.e. a graph embeds into its partial tree embedding with no congestion. However, to get embedding guarantees in the other direction more work is required. For this purpose we use a lemma from Madry [182] saying that we can construct a convex combination or a distribution of partial tree embeddings we can get such a guarantee.

Lemma 12.6.6 ([Probabilistic Partial Tree Embedding [8]]). *For any undirected capacitated graph $G = (V, E, \boldsymbol{\mu})$ and any $t > 0$ in $\widetilde{O}(t \cdot m)$ time we can find a collection of partial tree embeddings $H_1 = H(G, T_1, F_1), \ldots, H_t = H(G, T_t, F_t)$ and coefficients $\lambda_i \geq 0$ with $\sum_i \lambda_i = 1$ such that $\forall i \in [t]$ we have $|F_i| = \widetilde{O}(\frac{m \log U}{t})$ and such that $\sum_i \lambda_i \boldsymbol{M}'_{H_i}$ embeds $G' = \sum_i \lambda_i G_i$ into G with congestion $\widetilde{O}(1)$.*

Using this lemma, we can prove that we can reduce constructing an oblivious routing for a graph to constructing oblivious routings on several partial tree embeddings.

Lemma 12.6.7. *Let the H_i be graphs produced by Lemma 12.6.6 and for all i let $\boldsymbol{A}_i$ be an oblivious routing algorithm for H_i. It follows that $\boldsymbol{A} = \sum_i \lambda_i \boldsymbol{M}'_{H_i} \boldsymbol{A}_i$ is an oblivious routing on G with $\rho(\boldsymbol{A}) \leq \widetilde{O}(\max_i \rho(\boldsymbol{A}_i) \log n)$ and $\mathcal{T}(\boldsymbol{A}) = O(\sum_i \mathcal{T}(\boldsymbol{A}_i))$.*

Proof. The proof is similar to the proof of Lemma 12.4.8. For all i let $\boldsymbol{U}_i$ denote the capacity matrix of graph G_i. Then using Lemma 12.4.3 we get

$$\rho(\boldsymbol{A}) = \|\boldsymbol{U}^{-1}\boldsymbol{A}\boldsymbol{B}^T\boldsymbol{U}\|_\infty = \left\| \sum_{i=1}^{t} \lambda_i \boldsymbol{U}^{-1} \boldsymbol{M}'_{H_i} \boldsymbol{A}_i \boldsymbol{B}^T \boldsymbol{U} \right\|_\infty$$

Using that $\boldsymbol{M}_{H_i}$ is an embedding and therefore $\boldsymbol{B}_{H_i}^T \boldsymbol{M}_{H_i} = \boldsymbol{B}^T$ we get

$$\rho(\boldsymbol{A}) = \left\| \sum_{i=1}^{t} \lambda_i \boldsymbol{U}^{-1} \boldsymbol{M}'_{H_i} \boldsymbol{A}_i \boldsymbol{B}_{H_i}^T \boldsymbol{M}_{H_i} \boldsymbol{U} \right\|_\infty \leq \max_{j,k} \left\| \sum_{i=1}^{t} \lambda_i \boldsymbol{U}^{-1} \boldsymbol{M}'_i \boldsymbol{U}_i \right\|_\infty \cdot \rho(\boldsymbol{A}_j) \cdot \text{cong}(\boldsymbol{M}_{H_k})$$

Since $\sum_i \lambda_i \boldsymbol{M}'_{H_i}$ is an embedding of congestion of at most $\widetilde{O}(1)$ and $\text{cong}(\boldsymbol{M}_{H_k}) \leq 1$ we have the desired result. □

12.6.2 From Partial Tree Embeddings To Almost-j-trees

Here we show how to reduce constructing an oblivious routing for a partial tree embedding to constructing an oblivious routing for what Madry [182] calls an "almost j-tree," the union of a tree plus a subgraph on at most j vertices. First we define such objects and then we prove the reduction.

Definition 12.6.8 (Almost -tree). We call a graph $G = (V, E)$ an *almost j-tree* if there is a spanning tree $T \subseteq E$ such that the endpoints of $E \setminus T$ include at most j vertices.

[8] This in an adaptation of Corollary 5.6 in [182]

Lemma 12.6.9. *For undirected capacitated $G = (V, E, \boldsymbol{\mu})$ and partial tree embedding $H = H(G, T, F)$ in $\widetilde{O}(|E|)$ time we can construct an almost $2\cdot|F|$-tree $G' = (V, E', \boldsymbol{\mu}')$ with $|E'| \leq |E|$ and an embedding $\boldsymbol{M}'$ from G' to H such that H is embeddable into G' with congestion 2, $\text{cong}(\boldsymbol{M}') = 2$, and $\mathcal{T}(\boldsymbol{M}') = \widetilde{O}(|E|)$.*

Proof. For every $e = (a, b) \in E$, we let $v^1(e) \in V$ denote the first vertex on tree path $P_{(a,b)}$ incident to F and we let $v^2(e) \in V$ denote the last vertex incident to F on tree path $P_{(a,b)}$. Note that for every $e = (a, b) \in T$ we have that $(v^1(e), v^2(e)) = e$.

We define $G' = (V, E', \boldsymbol{\mu}')$ to simply be the graph that consists of all these $(v^1(e), v^2(e))$ pairs

$$E' = \{(a, b) \mid \exists e \in E \text{ such that } (a, b) = (v^1(e), v^2(e))\}$$

and we define the weights to simply be the sums

$$\forall e' \in E' \quad : \quad \boldsymbol{\mu}'(e') \stackrel{\text{def}}{=} \sum_{e \in E \mid e = (v^1(e'), v^2(e'))} \boldsymbol{\mu}(e)$$

Now to embed H in G' we define $\boldsymbol{M}$ by

$$\forall e = (a, b) \in E \quad : \quad \boldsymbol{M}\mathbf{1}_e = \boldsymbol{p}_{a,v^1(e)} + \mathbf{1}_{(v^1(e),v^2(e))} + \boldsymbol{p}_{v^2(e),b}$$

and to embed G' in H we define $\boldsymbol{M}'$ by

$$\forall e' \in E \quad : \quad \boldsymbol{M}'\mathbf{1}_{e'} = \sum_{e=(a,b)\in E \mid e' = (v^1(e),v^2(e))} \frac{\boldsymbol{\mu}(e)}{\boldsymbol{\mu}'(e')} \left[\boldsymbol{p}_{v^1(e),a} + \mathbf{1}_{(a,b)} + \boldsymbol{p}_{b,v^2(e)}\right]$$

In other words we route edges in H along the tree until we encounter nodes in F and then we route them along added edges and we simply route the other way for the reverse embedding. By construction clearly the congestion of the embedding in either direction is 2.

To bound the running time, we note that by having every edge e in H maintain its $v^1(e)$ and $v^2(e)$ information, having every edge e' in E' maintain the set $\{e \in E | e' = (v^1(e), v^2(e))\}$ in a list, and using link cut trees [240] or the static tree structure in [141] to update information along tree paths we can obtain the desired value of $\mathcal{T}(\boldsymbol{M}')$. □

12.6.3 From Almost-J Trees to Less Vertices

Here we show that by "greedy elimination" [243] [147] [146], i.e. removing all degree 1 and degree 2 vertices in $O(m)$ time we can reduce oblivious routing in almost-j-trees to oblivious routing in graphs with $O(j)$ vertices while only losing $O(1)$ in the competitive ratio. Again, we remark that the lemmas in this section are derived heavily from [182] but repeated for completeness and to prove additional properties that we will need for our purposes.

We start by showing that an almost-j-tree with no degree 1 or degree 2 vertices has at most $O(j)$ vertices.

Lemma 12.6.10. *For any almost j-tree $G = (V, E)$ with no degree 1 or degree 2 vertices, we have $|V| \leq 3j - 2$.*

Proof. Since G is an almost j-tree, there is some $J \subseteq V$ with $|J| \leq j$ such that the removal of all edges with both endpoints in J creates a forest. Now, since $K = V - J$ is incident only to forest edges clearly the sum of the degrees of the vertices in K is at most $2(|V| - 1)$ (otherwise there would be a cycle). However, since the minimum degree in G is 3, clearly this sum is at least $3(|V| - j)$. Combining yields that $3|V| - 3j \leq 2|V| - 2$. □

Next, we show how to remove degree one vertices efficiently.

Lemma 12.6.11 (Removing Degree One Vertices). *Let $G = (V, E, \boldsymbol{\mu})$ be an unweighted capacitated graph, let $a \in V$ be a degree 1 vertex, let $e = (a, b) \in E$ be the single edge incident to a, and let $G' = (V', E', \boldsymbol{\mu}')$ be the graph that results from simply removing e and a, i.e. $V' = V \setminus \{a\}$ and $E' = E \setminus \{e\}$. Given $a \in V$ and an oblivious routing algorithm $\boldsymbol{A}'$ in G' in $O(1)$ time we can construct an oblivious routing algorithm $\boldsymbol{A}$ in G such that*

$$\mathcal{T}(\boldsymbol{A}) = O(\mathcal{T}(\boldsymbol{A}') + 1) \quad , \text{ and } \quad \rho(\boldsymbol{A}) = \rho(\boldsymbol{A}')$$

Proof. For any demand vector $\boldsymbol{\chi}$, the only way to route demand at a in G is over e. Therefore, if $\boldsymbol{B}\boldsymbol{f} = \boldsymbol{\chi}$ then $\boldsymbol{f}(e) = \boldsymbol{\chi}$. Therefore, to get an oblivious routing algorithm on G, we can simply send demand at a over edge e, modify the demand at b accordingly, and then run the oblivious routing algorithm on G' on the remaining vertices. The routing algorithm we get is the following

$$\boldsymbol{A} \stackrel{\text{def}}{=} \boldsymbol{I}_{E' \to E} \boldsymbol{A}' (\boldsymbol{I} + \mathbf{1}_b \mathbf{1}_a^T) + \mathbf{1}_e \mathbf{1}_a^T$$

Since all routing algorithms send this flow on e we get that $\rho(\boldsymbol{A}) = \rho(\boldsymbol{A}')$ and since the above operators not counting $\boldsymbol{A}$ have only $O(1)$ entries that are not the identity we can clearly implement the operations in the desired running time. □

Using the above lemma we show how to remove all degree 1 and 2 vertices in $O(m)$ time while only increasing the congestion by $O(1)$.

Lemma 12.6.12 (Greedy Elimination). *Let $G = (V, E, \boldsymbol{\mu})$ be an unweighted capacitated graph and let $G' = (V', E', \boldsymbol{\mu}')$ be the graph the results from iteratively removing vertices of degree 1 and replacing degree 2 vertices with an edge connecting its neighbors of the minimum capacity of its adjacent edges. We can construct G' in $O(m)$ time and given an oblivious routing algorithm $\boldsymbol{A}'$ in G' in $O(1)$ time we can construct an oblivious routing algorithm $\boldsymbol{A}$ in G such that* [9]

$$\mathcal{T}(\boldsymbol{A}) = O(\mathcal{T}(\boldsymbol{A}') + |E|) \quad , \text{ and } \quad \rho(\boldsymbol{A}) \leq 4 \cdot \rho(\boldsymbol{A}')$$

Proof. First we repeatedly apply Lemma 12.6.11 repeatedly to in reduce to the case that there are no degree 1 vertices. By simply array of the degrees of every vertex and a list of degree 1 vertices this can be done in $O(m)$ time. We denote the result of these operations by graph K.

Next, we repeatedly find degree two vertices that have not been explored and explore this vertices neighbors to get a path of vertices, $a_1, a_2, \ldots, a_k \in V$ for $k > 3$ such that each vertex $a_2, \ldots, a_{k-1}$ is of degree two. We then compute $j = \arg\min_{i \in [k-1]} \boldsymbol{\mu}(a_i, a_{i+1})$, remove edge (a_j, a_{j+1}) and add an edge (a_1, a_k) of capacity $\boldsymbol{\mu}(a_j, a_{j+1})$. We denote the result of doing this for all degree two vertices by K' and note that again by careful implementation this can be performed in $O(m)$ time.

Note that clearly K is embeddable in K' with congestion 2 just by routing every edge over itself except the removed edges which we route by the path plus the added edges. Furthermore, K' is embeddable in K with congestion 2 again by routing every edge on itself except for the edges which we added which we route back over the paths they came from. Furthermore, we note that clearly this embedding and the transpose of this operator is computable in $O(m)$ time.

Finally, by again repeatedly applying Lemma 12.6.11 to K' until there are no degree 1 vertices we get a graph G' that has no degree one or degree two vertices (since nothing decreased the degree of vertices with degree more than two). Furthermore, by Lemma 12.6.11 and by Lemma 12.4.8 we see that we can compose these operators to compute $\boldsymbol{A}$ with the desired properties. □

[9] Note that the constant of 4 below is improved to 3 in [182].

12.6.4 Putting It All Together

Here we put together the previous components to prove the main theorem of this section.

Node Reduction Theorem 12.4.10. Using Lemma 12.6.6, we can construct $G' = \sum_{i=1}^{t} \lambda_i G_i$ and embeddings $\boldsymbol{M}_1, \ldots, \boldsymbol{M}_t$ from G_i to G. Next we can apply Lemma 12.6.9 to each G_i to get almost-j-trees $G'_1, \ldots, G'_t$ and embeddings $\boldsymbol{M}'_1, \ldots, \boldsymbol{M}'_t$ from G'_i to G_i. Furthermore, using Lemma 12.6.12 we can construction graphs $G''_1, \ldots, G''_t$ with the desired properties (the congestion ratio property follows from the fact that we only add capacities during these reductions)

Now given oblivious routing algorithms $\boldsymbol{A}''_1, \ldots, \boldsymbol{A}''_t$ on the G''_i and again by Lemma 12.6.12 we could get oblivious routing algorithms $\boldsymbol{A}'_1, \ldots, \boldsymbol{A}'_t$ on the G'_i with constant times more congestion. Finally, by the guarantees of Lemma 12.4.8 we have that $\boldsymbol{A} \stackrel{\text{def}}{=} \sum_{i=1}^{t} \lambda \boldsymbol{M}_i \boldsymbol{M}'_i \boldsymbol{A}'_i$ is an oblivious routing algorithm that satisfies the requirements. □

12.7 Nonlinear Projection and Maximum Concurrent Flow

12.7.1 Gradient Descent Method for Nonlinear Projection Problem

In this section, we strengthen and generalize the **MaxFlow** algorithm to a more general setting. We believe this algorithm may be of independent interest as it includes maximum concurrent flow problem, the compressive sensing problem, etc. For some norms, e.g. $\|\cdot\|_1$ as typically of interest compressive sensing, the Nesterov algorithm [208] can be used to replace gradient descent. However, this kind of accelerated method is not known in the general norm settings as good proxy function may not exist at all. Even worse, in the non-smooth regime, the minimization problem on the $\|\cdot\|_p$ space with $p > 2$ is difficult under some oracle assumption [201]. For these reasons we focus here on the gradient descent method which is always applicable.

Given a norm $\|\cdot\|$, we wish to solve the what we call the *non-linear projection* problem

$$\min_{\boldsymbol{x} \in L} \|\boldsymbol{x} - \boldsymbol{y}\|$$

where $\boldsymbol{y}$ is an given point and L is a linear subspace. We assume the following:

Assumption 12.7.1.

1. *There are a family of convex differentiable functions f_t such that for all $\boldsymbol{x} \in L$, we have*

$$\|\boldsymbol{x}\| \leq f_t(\boldsymbol{x}) \leq \|\boldsymbol{x}\| + Kt$$

   and the Lipschitz constant of ∇f_t is $\frac{1}{t}$.

2. *There is a projection matrix $\boldsymbol{P}$ onto the subspace L.*

In other words we assume that there is a family of regularized objective functions f_t and a projection matrix $\boldsymbol{P}$, which we can think of as an approximation algorithm of this projection problem.

Now, let $\boldsymbol{x}^*$ be a minimizer of $\min_{\boldsymbol{x} \in L} \|\boldsymbol{x} - \boldsymbol{y}\|$. Since $\boldsymbol{x}^* \in L$, we have $\boldsymbol{P}\boldsymbol{x}^* = \boldsymbol{x}^*$ and hence

$$\begin{aligned} \|\boldsymbol{P}\boldsymbol{y} - \boldsymbol{y}\| &\leq \|\boldsymbol{y} - \boldsymbol{x}^*\| + \|\boldsymbol{x}^* - \boldsymbol{P}\boldsymbol{y}\| \\ &\leq \|\boldsymbol{y} - \boldsymbol{x}^*\| + \|\boldsymbol{P}\boldsymbol{x}^* - \boldsymbol{P}\boldsymbol{y}\| \\ &\leq (1 + \|\boldsymbol{P}\|) \min_{\boldsymbol{x} \in L} \|\boldsymbol{x} - \boldsymbol{y}\|. \end{aligned} \tag{12.4}$$

Therefore, the approximation ratio of $\boldsymbol{P}$ is $1 + \|\boldsymbol{P}\|$ and we see that our problem is to show that we can solve nonlinear projection using a decent linear projection matrix. Our algorithm for solving this problem is shown in **NonlinearProjection**.

Algorithm 34: NonlinearProjection

Input: a point $\boldsymbol{y}$ and $\mathrm{OPT} = \min_{\boldsymbol{x} \in L} \|\boldsymbol{x} - \boldsymbol{y}\|$.
Let $\boldsymbol{y}_0 = (\boldsymbol{I} - \boldsymbol{P})\,\vec{y}$ and $\boldsymbol{x}_0 = 0$.
for $j = 0, \cdots,$ *until* $2^{-j}\|\boldsymbol{P}\| \le \frac{1}{2}$ **do**
  If $2^{-j}\|\boldsymbol{P}\| > 1$, then let $t_j = \frac{2^{-(j+2)}\|\boldsymbol{P}\|\mathrm{OPT}}{K}$ and $k_j = 3200\|\boldsymbol{P}\|^2 K$.
  If $2^{-j}\|\boldsymbol{P}\| \le 1$, then let $t_j = \frac{\varepsilon \mathrm{OPT}}{2K}$ and $k_j = \frac{800\|\boldsymbol{P}\|^2 K}{\varepsilon^2}$.
  Let $g_j(\boldsymbol{x}) = f_{t_j}(\boldsymbol{P}\boldsymbol{x} - \boldsymbol{y}_j)$ and $\boldsymbol{x}_0 = 0$.
  for $i = 0, \cdots, k_j - 1$ **do**
    $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i - \frac{t}{\|\boldsymbol{P}\|^2}(\nabla g_j(\boldsymbol{x}_i))^{\#}$.
  end
  Let $\boldsymbol{y}_{j+1} = \boldsymbol{y}_j - \boldsymbol{P}\boldsymbol{x}_{k_j}$.
end
Output: $\boldsymbol{y} - \boldsymbol{y}_{\text{last}}$.

Note that this algorithm and its proof are quite similar to Theorem 12.3.4 but modified to scale parameters over an outer loop. By changing the parameter t we can decrease the dependence of the initial error.[10]

Theorem 12.7.2. *Assume the conditions in Assumption 12.7.1 are satisfied. Let $\mathcal{T}$ be the time needed to compute $\boldsymbol{P}x$ and $\boldsymbol{P}^T x$ and $x^{\#}$. Then,* ***NonlinearProjection*** *outputs a vector $\boldsymbol{x}$ with $\|\boldsymbol{x}\| \le (1+\varepsilon)\min_{\boldsymbol{x} \in L}\|\boldsymbol{x} - \boldsymbol{y}\|$ and the algorithm takes time*

$$O\left(\|\boldsymbol{P}\|^2 K\,(\mathcal{T} + m)\left(\frac{1}{\varepsilon^2} + \log\|\boldsymbol{P}\|\right)\right).$$

Proof. We prove by induction on j that when $2^{-(j-1)}\|\boldsymbol{P}\| \ge 1$ we have $\|\boldsymbol{y}_j\| \le \left(1 + 2^{-j}\|\boldsymbol{P}\|\right)\mathrm{OPT}$.

For the base case ($j = 0$), (12.7.2) shows that $\|\boldsymbol{y}_0\| \le (1 + \|\boldsymbol{P}\|)\,\mathrm{OPT}$.

For the inductive case we assume that the assertion holds for some j. We start by bounding the corresponding R in Theorem 12.3.1 for g_j, which we denote R_j. Note that

$$g_j(\boldsymbol{x}_0) = f_{t_j}(-\boldsymbol{y}_j) \le \|\boldsymbol{y}_j\| + Kt_j \le \left(1 + 2^{-j}\|\boldsymbol{P}\|\right)\mathrm{OPT} + Kt_j.$$

Hence, the condition that $g_j(\boldsymbol{x}) \le g_j(\boldsymbol{x}_0)$ implies that

$$\|\boldsymbol{P}\boldsymbol{x} - \boldsymbol{y}_j\| \le \left(1 + 2^{-j}\|\boldsymbol{P}\|\right)\mathrm{OPT} + Kt_j.$$

Take any $\boldsymbol{y} \in X^*$, let $\boldsymbol{c} = \boldsymbol{x} - \boldsymbol{P}\boldsymbol{x} + \boldsymbol{y}$, and note that $\boldsymbol{P}\boldsymbol{c} = \boldsymbol{P}\boldsymbol{y}$ and therefore $\boldsymbol{c} \in X^*$. Using these facts, we can bound R_j as follows

$$\begin{aligned}
R_j &= \max_{\boldsymbol{x} \in \mathbb{R}^E \,:\, g_j(\boldsymbol{x}) \le g_j(\boldsymbol{x}_0)} \|\boldsymbol{x} - \boldsymbol{c}\| \\
&\le \max_{\boldsymbol{x} \in \mathbb{R}^E \,:\, g_j(\boldsymbol{x}) \le g_j(\boldsymbol{x}_0)} \|\boldsymbol{P}\boldsymbol{x}\| + \|\boldsymbol{P}\boldsymbol{y}\| \\
&\le 2\|\boldsymbol{y}_0\| + \|\boldsymbol{P}\boldsymbol{x} - \boldsymbol{y}_j\| + \|\boldsymbol{P}\boldsymbol{y} - \boldsymbol{y}_j\| \\
&\le 4\left(1 + 2^{-j}\|\boldsymbol{P}\|\right)\mathrm{OPT} + 2Kt_j.
\end{aligned}$$

[10] This is an idea that has been applied previously to solve linear programming problems [207].

Similar to Lemma 12.3.3, the Lipschitz constant L_j of g_j is $\|\boldsymbol{P}\|^2/t_j$. Hence, Theorem 12.3.1 shows that

$$\begin{aligned} g_j(\boldsymbol{x}_{k_j}) &\leq \min_{\boldsymbol{x}} g_j(\boldsymbol{x}) + \frac{2 \cdot L_j \cdot R_j^2}{k_j + 4} \\ &\leq \min_{\boldsymbol{x}} \|\boldsymbol{P}\vec{x} - \vec{y_j}\| + \frac{2 \cdot L_j \cdot R_j^2}{k_j + 4} + Kt_j \end{aligned}$$

So, we have

$$\begin{aligned} \|\boldsymbol{P}\boldsymbol{x}_{k_j} - \boldsymbol{y}_j\| &\leq f_{t_j}(\boldsymbol{P}\boldsymbol{x}_{k_j} - \boldsymbol{y}_j) \\ &\leq \mathrm{OPT} + Kt_j + \frac{2\|\boldsymbol{P}\|^2}{t_j(k_j+4)} \left(4\left(1 + 2^{-j}\|\boldsymbol{P}\|\right)\mathrm{OPT} + 2Kt_j\right)^2. \end{aligned}$$

When $2^{-j}\|\boldsymbol{P}\| > 1$, we have $t_j = \frac{2^{-(j+2)}\|\boldsymbol{P}\|\mathrm{OPT}}{K}$ and $k_j = 3200\|\boldsymbol{P}\|^2 K$ and hence

$$\|\boldsymbol{y}_{j+1}\| = \|\boldsymbol{P}\boldsymbol{x}_{k_j} - \boldsymbol{y}_j\| \leq \left(1 + 2^{-j-1}\|\boldsymbol{P}\|\right)\mathrm{OPT}.$$

When $2^{-j}\|\boldsymbol{P}\| \leq 1$, we have $t_j = \frac{\varepsilon \mathrm{OPT}}{2K}$ and $k_j = \frac{800\|\boldsymbol{P}\|^2 K}{\varepsilon^2}$ and hence

$$\|\boldsymbol{y}_{j+1}\| = \|\boldsymbol{P}\boldsymbol{x}_{k_j} - \boldsymbol{y}_j\| \leq (1+\varepsilon)\,\mathrm{OPT}.$$

Since $\boldsymbol{y}_{\text{last}}$ is $\boldsymbol{y}$ plus some vectors in L, $\boldsymbol{y} - \boldsymbol{y}_{\text{last}} \in L$ and $\|\boldsymbol{t} - \boldsymbol{y}_{\text{last}} - \boldsymbol{y}\| = \|\boldsymbol{y}_{\text{last}}\| \leq (1+\varepsilon)\,\mathrm{OPT}$. □

12.7.2 Maximum Concurrent Flow

For an arbitrary set of demands $\boldsymbol{\chi}_i \in \mathbb{R}^V$ with $\sum_{v \in V} \boldsymbol{\chi}_i(v) = 0$ for $i = 1, \cdots, k$, we wish to solve the following *maximum concurrent flow* problem

$$\max_{\alpha \in R, \boldsymbol{f} \in \mathbb{R}^E} \alpha \quad \text{subject to} \quad \boldsymbol{B}^T \boldsymbol{f}_i = \alpha \boldsymbol{\chi}_i \quad \text{and} \quad \|\boldsymbol{U}^{-1} \sum_{i=1}^{k} |\boldsymbol{f}_i|\|_\infty \leq 1.$$

Similar to Section 12.3.2, it is equivalent to the problem

$$\min_{\boldsymbol{c} \in \mathbb{R}^{E \times [k]}} \| \sum_{i=1}^{k} |\boldsymbol{\alpha_i} + (\boldsymbol{Q}\boldsymbol{x})_i| \,\|_\infty$$

where $\boldsymbol{Q}$ is a projection matrix onto the subspace $\{\boldsymbol{B}^T\boldsymbol{U}\boldsymbol{x}_i = 0\}$, the output maximum concurrent flow is

$$\boldsymbol{f}_i(\boldsymbol{x}) = \boldsymbol{U}(\boldsymbol{\alpha}_i + (\boldsymbol{Q}\boldsymbol{x})_i) / \| \sum_{i=1}^{k} |\boldsymbol{\alpha}_i + (\boldsymbol{Q}\boldsymbol{x})_i| \,\|_\infty \quad ,$$

and $\boldsymbol{U}\boldsymbol{\alpha}_i$ is any flow such that $\boldsymbol{B}^T\boldsymbol{U}\vec{\alpha_i} = \boldsymbol{\chi}_i$. In order to apply **NonlinearProjection**, we need to find a regularized norm and a good projection matrix. Let us define the norm

$$\|\boldsymbol{x}\|_{1;\infty} = \max_{e \in E} \sum_{i=1}^{k} |x_i(e)|.$$

The problem is simply $\|\boldsymbol{\alpha} + \boldsymbol{Q}\boldsymbol{x}\|_{1;\infty}$ where $\boldsymbol{Q}$ is a projection matrix from $\mathbb{R}^{E \times [k]}$ to $\mathbb{R}^{E \times [k]}$ onto some subspace. Since each copy $\mathbb{R}^E$ is same, there is no reason that there is coupling in $\boldsymbol{Q}$ between different copies of $\mathbb{R}^E$. In the next lemma, we formalize this by the fact that any good projection

matrix$\boldsymbol{P}$ onto the subspace $\{\boldsymbol{B}^T\boldsymbol{U}\vec{x} = 0\} \subset \mathbb{R}^E$ extends to a good projection $\boldsymbol{Q}$ onto the subspace $\{\boldsymbol{B}^T\boldsymbol{U}\boldsymbol{x}_i = 0\} \subset \mathbb{R}^{E\times[k]}$. Therefore, we can simply extends the good circulation projection $\boldsymbol{P}$ by formula $(\boldsymbol{Q}\boldsymbol{x})_i = \boldsymbol{P}\boldsymbol{x}_i$. Thus, the only last piece needed is a regularized $\|\cdot\|_{1;\infty}$. However, it turns out that smoothing via conjugate does not work well in this case because the dual space of $\|\cdot\|_{1;\infty}$ involves with $\|\cdot\|_\infty$, which is unfavorable for this kind of smoothing procedure. It can be shown that there is no such good regularized $\|\cdot\|_{1;\infty}$. Therefore, we could not do $O(m^{1+o(1)}k/\varepsilon^2)$ using this approach, however, $O(m^{1+o(1)}k^2/\varepsilon^2)$ is possible by using a bad regularized $\|\cdot\|_{1;\infty}$. We believe the dependence of k can be improved to $3/2$ using this approach by suitable using Nesterov algorithm because the $\|\cdot\|_1$ space caused by the multicommodity is a favorable geometry for accelerated methods.

Lemma 12.7.3. *Let* $smaxL1_t(\boldsymbol{x}) = \mathrm{smax}_t\left(\sum_{i=1}^k \sqrt{(x_i(e))^2 + t^2}\right)$. *It is a convex continuously differentiable function. The Lipschitz constant of* $\nabla smaxL1_t$ *is* $\frac{2}{t}$ *and*

$$\|\boldsymbol{x}\|_{1;\infty} - t\ln(2m) \le smaxL1_t(\boldsymbol{x}) \le \|\boldsymbol{x}\|_{1;\infty} + kt.$$

Proof. 1) It is clear that $\mathrm{smax}L1_t$ is smooth.

2) $\mathrm{smax}L1_t$ is convex.

Since smax_t is increasing for positive values and $\sqrt{x^2+t^2}$ is convex, for any $\boldsymbol{x},\boldsymbol{y} \in \mathbb{R}^{E\times[k]}$ and $0 \le t \le 1$, we have

$$\begin{aligned}
\mathrm{smax}L1_t(t\boldsymbol{x} + (1-t)\boldsymbol{y}) &= \mathrm{smax}_t\left(\sum_{i=1}^k \sqrt{((tx_i + (1-t)y_i)(e))^2 + t^2}\right)\\
&\le \mathrm{smax}_t\left(\sum_{i=1}^k \left(t\sqrt{(x_i(e))^2+t^2} + (1-t)\sqrt{(y_i(e))^2+t^2}\right)\right)\\
&\le t\mathrm{smax}L1_t(\boldsymbol{x}) + (1-t)\mathrm{smax}L1_t(\boldsymbol{y}).
\end{aligned}$$

3) The Lipschitz constant of $\nabla\mathrm{smax}L1_t$ is $\frac{2}{t}$.

Note that smax_t (not its gradient) has Lipschitz constant 1 because for any $\boldsymbol{x},\boldsymbol{y} \in \mathbb{R}^E$,

$$\begin{aligned}
&|\mathrm{smax}_t(\boldsymbol{x}) - \mathrm{smax}_t(\boldsymbol{y})|\\
=\ &\left|t\ln\left(\frac{\sum_{e\in E}\left(\exp(-\frac{x(e)}{t}) + \exp(\frac{x(e)}{t})\right)}{2m}\right) - t\ln\left(\frac{\sum_{e\in E}\left(\exp(-\frac{y(e)}{t}) + \exp(\frac{y(e)}{t})\right)}{2m}\right)\right|\\
=\ &t\left|\ln\left(\frac{\sum_{e\in E}\left(\exp(-\frac{x(e)}{t}) + \exp(\frac{x(e)}{t})\right)}{\sum_{e\in E}\left(\exp(-\frac{y(e)}{t}) + \exp(\frac{y(e)}{t})\right)}\right)\right|\\
\le\ &t\left|\ln\left(\max_{e\in E}\exp(\frac{|x-y|(e)}{t})\right)\right|\\
=\ &\|\boldsymbol{x}-\boldsymbol{y}\|_\infty.
\end{aligned}$$

Also, by the definition of derivative, for any $\boldsymbol{x},\boldsymbol{y} \in \mathbb{R}^n$ and $t \in \mathbb{R}$, we have

$$\mathrm{smax}_t(\boldsymbol{x} + t\boldsymbol{y}) - \mathrm{smax}_t(\boldsymbol{x}) = t\langle\nabla\mathrm{smax}_t(\boldsymbol{x}), \boldsymbol{y}\rangle + o(t).$$

and it implies $|\langle\nabla\mathrm{smax}_t(\boldsymbol{x}), \boldsymbol{y}\rangle| \le \|\boldsymbol{y}\|_\infty$ for arbitrary $\boldsymbol{y}$ and hence

$$\|\nabla\mathrm{smax}_t(\boldsymbol{x})\|_1 \le 1. \tag{12.5}$$

For notational simplicity, let $s_1 = \text{smax}L1_t$, $s_2 = \text{smax}_t$ and $s_3(x) = \sqrt{x^2+t^2}$. Thus, we have

$$s_1(\vec{x}) = s_2\left(\sum_{i=1}^{k} s_3(x_i(e))\right).$$

Now, we want to prove

$$\|\nabla s_1(\boldsymbol{x}) - \nabla s_1(\boldsymbol{y})\|_{\infty;1} \le \frac{2}{t}\|\boldsymbol{x}-\boldsymbol{y}\|_{1;\infty}.$$

Note that

$$\frac{\partial s_1(\boldsymbol{x})}{\partial x_i(e)} = \frac{\partial s_2}{\partial e}\left(\sum_j s_3(x_j(e))\right)\frac{ds_3}{dx}(x_i(e)).$$

Hence, we have

$$\begin{aligned}
&\|\nabla s_1(\boldsymbol{x}) - \nabla s_1(\boldsymbol{y})\|_{\infty;1}\\
=\ & \sum_e \max_i \left|\frac{\partial s_2}{\partial e}\left(\sum_j s_3(x_j(e))\right)\frac{ds_3}{dx}(x_i(e)) - \frac{\partial s_2}{\partial e}\left(\sum_j s_3(y_j(e))\right)\frac{ds_3}{dx}(y_i(e))\right|\\
\le\ & \sum_e \left|\frac{\partial s_2}{\partial e}\left(\sum_j s_3(x_j(e))\right)\right| \max_i \left|\frac{ds_3}{dx}(x_i(e)) - \frac{ds_3}{dx}(y_i(e))\right|\\
&+ \sum_e \max_i \frac{ds_3}{dx}(y_i(e))\left|\frac{\partial s_2}{\partial e}\left(\sum_j s_3(x_j(e))\right) - \frac{\partial s_2}{\partial e}\left(\sum_j s_3(y_j(e))\right)\right|.
\end{aligned}$$

Since s_3 has $\frac{1}{t}$-Lipschitz gradient, we have

$$\left|\frac{ds_3}{dx}(\boldsymbol{x}) - \frac{ds_3}{dx}(\boldsymbol{y})\right| \le \frac{1}{t}|\boldsymbol{x}-\boldsymbol{y}|.$$

By (12.5), we have

$$\sum_e \left|\frac{\partial s_2}{\partial e}(\boldsymbol{x}(e))\right| \le 1.$$

Hence, we have

$$\begin{aligned}
&\sum_e \max_i \left|\frac{ds_3}{dx}(x_i(e)) - \frac{ds_3}{dx}(y_i(e))\right| \left|\frac{\partial s_2}{\partial e}\left(\sum_i s_3(x_i(e))\right)\right|\\
\le\ & \frac{1}{t}\max_{i,e}|x_i(e)-y_i(e)| \sum_e \left|\frac{\partial s_3}{\partial e}\left(\sum_i s_3(x_i(e))\right)\right|\\
=\ & \frac{1}{t}\|\vec{x}-\vec{y}\|_{1;\infty}.
\end{aligned}$$

Since s_3 is 1-Lipschitz, we have

$$\left|\frac{ds_3}{dx}\right| \le 1.$$

Since s_2 has $\frac{1}{t}$-Lipschitz gradient in $\|\cdot\|_\infty$, we have

$$\sum_e \left| \frac{\partial s_2}{\partial e}(x) - \frac{\partial s_2}{\partial e}(y) \right| \le \frac{1}{t}\|\boldsymbol{x}-\boldsymbol{y}\|_\infty.$$

Hence, we have

$$\begin{aligned}
& \sum_e \max_i \frac{ds_3}{dx}(y_i(e)) \left| \frac{\partial s_2}{\partial e}\left(\sum_j s_3(x_j(e))\right) - \frac{\partial s_2}{\partial e}\left(\sum_j s_3(y_j(e))\right) \right| \\
\le & \sum_e \left| \frac{\partial s_2}{\partial e}\left(\sum_j s_3(x_j(e))\right) - \frac{\partial s_2}{\partial e}\left(\sum_j s_3(y_j(e))\right) \right| \\
\le & \frac{1}{t}\| \sum_i s_3(x_i(e)) - \sum s_3(y_i(e))\|_\infty \\
\le & \frac{1}{t}\| \sum_i |x_i(e) - y_i(e)| \,\| \\
= & \frac{1}{t}\|\boldsymbol{x}-\boldsymbol{y}\|_{1;\infty}
\end{aligned}$$

Therefore, we have

$$\|\nabla s_1(\boldsymbol{x}) - \nabla s_1(\boldsymbol{y})\|_{\infty;1} \le \frac{2}{t}\|\boldsymbol{x}-\boldsymbol{y}\|_{1;\infty}.$$

4) Using the fact that

$$\|x(e)\| \le \sum_{i=1}^{k} \sqrt{(x_i(e))^2 + t^2} \le \|x(e)\|_1 + kt$$

and smax is 1-Lipschitz, we have

$$\|\boldsymbol{x}\|_{1;\infty} - t\ln(2m) \le \mathrm{smax} L1_t(\boldsymbol{x}) \le \|\boldsymbol{x}\|_{1;\infty} + kt.$$

□

The last thing needed is to check is that the # operator is easy to compute.

Lemma 12.7.4. *In $\|\cdot\|_{1;\infty}$, the # operator is given by an explicit formula*

$$\left(\boldsymbol{x}^\#\right)_i(e) = \begin{cases} ||\boldsymbol{x}||_{\infty;1}\mathrm{sign}(x_i(e)) & \text{if } i \text{ is the smallest index such that } \max_j x_j(e) = x_i(e) \\ 0 & \text{otherwises} \end{cases}.$$

Proof. It can be proved by direct computation. □

Now, all the conditions in the Assumption 12.7.1 are satisfied. Therefore, Theorem 12.7.2 and Theorem 12.4.12 gives us the following theorem:

Theorem 12.7.5. *Given an undirected capacitated graph $G = (V, E, \boldsymbol{\mu})$ with capacity ratio U. Assume $U = \mathrm{poly}(|V|)$. There is an algorithm finds an $(1-\varepsilon)$ approximate Maximum Concurrent Flow in time*

$$O\left(\frac{k^2}{\varepsilon^2}|E|2^{O\left(\sqrt{\log|V|\log\log|V|}\right)}\right).$$

Proof. Let $\boldsymbol{A}$ be the oblivious routing algorithm given by Theorem 12.4.12. And we have $\rho(\boldsymbol{A}) \leq 2^{O\left(\sqrt{\log|V|\log\log|V|}\right)}$. Let us define the scaled circulation projection matrix $\boldsymbol{P} = \boldsymbol{I} - \boldsymbol{U}\boldsymbol{A}\boldsymbol{B}^T\boldsymbol{U}^{-1}$. Lemma 12.4.5 shows that $\|\boldsymbol{P}\|_\infty \leq 1 + 2^{O\left(\sqrt{\log|V|\log\log|V|}\right)}$.

Let the multi-commodity circulation projection matrix $\boldsymbol{Q} : \mathbb{R}^{E\times[k]} \to \mathbb{R}^{E\times[k]}$ defined by $(\boldsymbol{Q}\boldsymbol{x})_i = \boldsymbol{P}\boldsymbol{x}_i$. Note that the definition of $\|\boldsymbol{Q}\|_{1;\infty}$ is similar to $\rho(\boldsymbol{Q})$. By similar proof as Lemma 12.4.3, we have $\|\boldsymbol{Q}\|_{1;\infty} = \|\boldsymbol{P}\|_\infty$. Hence, we have $\|\boldsymbol{Q}\|_{1;\infty} \leq 1 + 2^{O\left(\sqrt{\log|V|\log\log|V|}\right)}$. Also, since $\boldsymbol{P}$ is a projection matrix on the subspace $\{\boldsymbol{x} \in \mathbb{R}^E : \boldsymbol{B}^T\boldsymbol{U}\boldsymbol{x} = 0\}$, $\boldsymbol{Q}$ is a projection matrix on the subspace $\{\boldsymbol{x} \in \mathbb{R}^{E\times[k]} : \boldsymbol{B}^T\boldsymbol{U}\boldsymbol{x}_i = 0\}$.

By Lemma 12.7.3, the function $\mathrm{smax}L1_t(\boldsymbol{x})$ is a convex continuously differentiable function such that the Lipschitz constant of $\nabla\mathrm{smax}L1_t$ is $\frac{2}{t}$ and

$$\|\boldsymbol{x}\|_{1;\infty} - t\ln(2m) \leq \mathrm{smax}L1_t(\boldsymbol{x}) \leq \|\boldsymbol{x}\|_{1;\infty} + kt.$$

Given an arbitrary set of demands $\boldsymbol{\chi}_i \in \mathbb{R}^V$, we find a vector $\vec{y}$ such that

$$\boldsymbol{B}^T\boldsymbol{U}\boldsymbol{y} = -\boldsymbol{\chi}_i.$$

Then, we use the **NonlinearProjection** to solve

$$\min_{\boldsymbol{B}^T\boldsymbol{U}\boldsymbol{x}=0} \|\boldsymbol{x} - \boldsymbol{y}\|_{1;\infty}$$

using a family of functions $\mathrm{smax}L1_t(\boldsymbol{x}) + t\ln(2n)$ and the projection matrix $\boldsymbol{Q}$. Since each iteration involves calculation of gradients and # operator, it takes $O(mk)$ each iteration. And it takes $\tilde{O}\left(\|\boldsymbol{Q}\|^2_{1;\infty}K/\varepsilon^2\right)$ iterations in total where $K = k+\ln(2m)$. In total, it **NonlinearProjection** outputs a $(1+\varepsilon)$ approximate minimizer $\boldsymbol{x}$ in time

$$O\left(\frac{k^2}{\varepsilon^2}|E|2^{O\left(\sqrt{\log|V|\log\log|V|}\right)}\right).$$

And it gives a $(1-\varepsilon)$ approximate maximum concurrent flow $\vec{f}_i$ by a direct formula. □

12.8 Appendix

In this section, we present some basic fact used in this chapter about norm and dual norm. Also, we presented some lemmas about convex functions with Lipschitz gradient. See [35, 206] for comprehensive discussion.

12.8.1 Norms

Fact 12.8.1. $\boldsymbol{x} = 0 \quad \Leftrightarrow \boldsymbol{x}^\# = 0.$

Proof. If $\boldsymbol{x} = 0$ then $\forall \boldsymbol{s} \neq 0$ we have $\langle \boldsymbol{x}, \boldsymbol{s}\rangle - \frac{1}{2}\|\boldsymbol{s}\|^2 < 0$ but $\langle \boldsymbol{x}, \boldsymbol{x}\rangle - \frac{1}{2}\|\boldsymbol{x}\|^2 = 0$. So we have $\boldsymbol{x}^\# = 0$. If $\boldsymbol{x} \neq 0$ then let $\boldsymbol{s} = \frac{\langle \boldsymbol{x},\boldsymbol{x}\rangle}{\|\boldsymbol{x}\|^2}\boldsymbol{x}$ with this choice we have $\langle \boldsymbol{x}, \boldsymbol{s}\rangle - \frac{1}{2}\|\boldsymbol{s}\|^2 = \frac{1}{2}\frac{\langle \boldsymbol{x},\boldsymbol{x}\rangle^2}{\|\boldsymbol{x}\|^2} > 0$. However, for $\boldsymbol{s} = 0$ we have that $\langle \boldsymbol{x}, \boldsymbol{s}\rangle - \frac{1}{2}\|\boldsymbol{s}\|^2 = 0$ therefore we have $\boldsymbol{x}^\# \neq 0$. □

Fact 12.8.2. $\forall \boldsymbol{x} \in \mathbb{R}^n \quad : \quad \langle x, x^\#\rangle = \|x^\#\|^2.$

Proof. If $\boldsymbol{x} = 0$ then $\boldsymbol{x}^\# = 0$ by Fact 12.8.1 and we have the result. Otherwise, again by Fact 12.8.1

we know that $\boldsymbol{x}^{\#} \neq 0$ and therefore by the definition of $\boldsymbol{x}^{\#}$ we have

$$1 = \arg\max_{c\in\mathbb{R}} \langle x, c\cdot x^{\#}\rangle - \frac{1}{2}\|c\cdot x^{\#}\|^2 = \arg\max_{c\in\mathbb{R}} c\cdot\langle x, x^{\#}\rangle - \frac{c^2}{2}\|x^{\#}\|^2$$

Setting the derivative of with respect to c to 0 we get that $1 = c = \frac{\langle \boldsymbol{x},\boldsymbol{x}^{\#}\rangle}{\|\boldsymbol{x}^{\#}\|^2}$. □

Fact 12.8.3. $\forall \boldsymbol{x}\in\mathbb{R}^n \quad : \quad \|\boldsymbol{x}\|^* = \|\boldsymbol{x}^{\#}\|$.

Proof. Note that if $\boldsymbol{x} = 0$ then the fact follows from Fact (12.8.1) otherwise we have

$$\|\boldsymbol{x}\|^* = \max_{\|y\|\leq 1}\langle x, y\rangle = \max_{\|y\|=1}\langle x, y\rangle \leq \max_{y\in\mathbb{R}^n}\frac{\langle x, y\rangle}{\|y\|}$$

From this it is clear that $\|\boldsymbol{x}\|^* \geq \|\boldsymbol{x}^{\#}\|$. To see the other direction consider a $\boldsymbol{y}$ that maximizes the above and let $\boldsymbol{z} = \frac{\langle \boldsymbol{x},\boldsymbol{y}\rangle}{\|\boldsymbol{y}\|^2}\boldsymbol{y}$

$$\langle \boldsymbol{x}, \boldsymbol{z}\rangle - \frac{1}{2}\|\boldsymbol{z}\|^2 \leq \langle \boldsymbol{x}, \boldsymbol{x}^{\#}\rangle - \frac{1}{2}\|\boldsymbol{x}^{\#}\|^2$$

and therefore

$$\|\boldsymbol{x}\|^{*2} - \frac{1}{2}\|\boldsymbol{x}\|^{*2} \leq \frac{1}{2}\|\boldsymbol{x}^{\#}\|^2$$

□

Fact 12.8.4 (Cauchy Schwarz). $\forall \boldsymbol{x}, \boldsymbol{y}\in\mathbb{R}^n \quad : \quad \langle \boldsymbol{y}, \boldsymbol{x}\rangle \leq \|\boldsymbol{y}\|^*\|\boldsymbol{x}\|$.

Proof. By the definition of dual norm, for all $\|\boldsymbol{x}\| = 1$, we have $\langle \boldsymbol{y}, \boldsymbol{x}\rangle \leq \|\boldsymbol{y}\|^*$. Hence, it follows by linearity of both side. □

■ 12.8.2 Functions with Lipschitz Gradient

Lemma 12.8.5. *Let f be a continuously differentiable convex function. Then, the following are equivalence:*

$$\forall \boldsymbol{x}, \boldsymbol{y}\in\mathbb{R}^n \quad : \quad \|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\|^* \leq L\cdot\|\boldsymbol{x} - \boldsymbol{y}\|$$

and

$$\forall \boldsymbol{x}, \boldsymbol{y}\in\mathbb{R}^n \quad : \quad f(\boldsymbol{x}) \leq f(\boldsymbol{y}) + \langle \nabla f(\boldsymbol{y}), \boldsymbol{x} - \boldsymbol{y}\rangle + \frac{L}{2}\|\boldsymbol{x} - \boldsymbol{y}\|^2.$$

For any such f and any $\boldsymbol{x}\in\mathbb{R}^n$, we have

$$f\left(\boldsymbol{x} - \frac{1}{L}\nabla f(\boldsymbol{x})^{\#}\right) \leq f(\boldsymbol{x}) - \frac{1}{2L}\|\nabla f(\boldsymbol{x})\|^{*2}.$$

Proof. From the first condition, we have

$$\begin{aligned}
f(\boldsymbol{y}) &= f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x}\rangle + \int_0^1 \langle \nabla f(\boldsymbol{x} + t(\boldsymbol{y} - \boldsymbol{x})) - \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x}\rangle dt \\
&\leq f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x}\rangle + \int_0^1 \|\nabla f(\boldsymbol{x} + t(\boldsymbol{y} - \boldsymbol{x})) - \nabla f(\boldsymbol{x})\|^*\|\boldsymbol{y} - \boldsymbol{x}\| dt \\
&\leq f(x) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x}\rangle + \int_0^1 Lt\|\boldsymbol{y} - \boldsymbol{x}\|^2 dt \\
&= f(x) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x}\rangle + \frac{L}{2}\|\boldsymbol{y} - \boldsymbol{x}\|^2.
\end{aligned}$$

Given the second condition. For any $\boldsymbol{x} \in \mathbb{R}^n$. let $\phi_{\boldsymbol{x}}(\boldsymbol{y}) = f(\boldsymbol{y}) - \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} \rangle$. From the convexity of f, for any $\boldsymbol{y} \in \mathbb{R}^n$

$$f(\boldsymbol{y}) - f(\boldsymbol{x}) \geq \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle.$$

Hence, $\boldsymbol{x}$ is a minimizer of $\phi_{\boldsymbol{x}}$. Hence, we have

$$\begin{aligned}
\phi_{\boldsymbol{x}}(\boldsymbol{x}) &\leq \phi_{\boldsymbol{x}}(\boldsymbol{y} - \frac{1}{L}\nabla\phi_{\boldsymbol{x}}(\boldsymbol{y})^{\#}) \\
&\leq \phi_{\boldsymbol{x}}(\boldsymbol{y}) - \langle \nabla\phi_{\boldsymbol{x}}(\boldsymbol{y}), \frac{1}{L}\nabla\phi_{\boldsymbol{x}}(\boldsymbol{y})^{\#} \rangle + \frac{L}{2}\|\frac{1}{L}\nabla\phi_{\boldsymbol{x}}(\boldsymbol{y})^{\#}\|^2 \qquad \text{(First part of this lemma)} \\
&= \phi_{\boldsymbol{x}}(\boldsymbol{y}) - \frac{1}{2L}\|\nabla\phi_{\boldsymbol{x}}(\boldsymbol{y})^{\#}\|^2 \\
&= \phi_{\boldsymbol{x}}(\boldsymbol{y}) - \frac{1}{2L}\left(\|\nabla\phi_{\boldsymbol{x}}(\boldsymbol{y})\|^*\right)^2.
\end{aligned}$$

Hence,

$$f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle + \frac{1}{2L}\left(\|\nabla f(\boldsymbol{y}) - \nabla f(\boldsymbol{x})\|^*\right)^2.$$

Adding up this inequality with $\boldsymbol{x}$ and $\boldsymbol{y}$ interchanged, we have

$$\begin{aligned}
\frac{1}{L}\left(\|\nabla f(\boldsymbol{y}) - \nabla f(\boldsymbol{x})\|^*\right)^2 &\leq \langle \nabla f(\boldsymbol{y}) - \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle \\
&\leq \|\nabla f(\boldsymbol{y}) - \nabla f(\boldsymbol{x})\|^* \|\boldsymbol{y} - \boldsymbol{x}\|.
\end{aligned}$$

The last inequality follows from similar proof in above for $\phi_{\boldsymbol{x}}$. □

The next lemma relate the Hessian of function with the Lipschitz parameter L and this lemma gives us a easy way to compute L.

Lemma 12.8.6. *Let f be a twice differentiable function such that for any $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$*

$$0 \leq \boldsymbol{y}^T \left(\nabla^2 f(\boldsymbol{x})\right) \boldsymbol{y} \leq L||\boldsymbol{y}||^2.$$

Then, f is convex and the gradient of f is Lipschitz continuous with Lipschitz parameter L.

Proof. Similarly to Lemma 12.8.5, we have

$$\begin{aligned}
f(\boldsymbol{y}) &= f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle + \int_0^1 \langle \nabla f(\boldsymbol{x} + t(\boldsymbol{y} - \boldsymbol{x})) - \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle dt \\
&= f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle + \int_0^1 t(\boldsymbol{y} - \boldsymbol{x})^T \nabla^2 f(\boldsymbol{x} + \theta_t(\boldsymbol{y} - \boldsymbol{x}))(\boldsymbol{y} - \boldsymbol{x}) dt
\end{aligned}$$

where the $0 \leq \theta_t \leq t$ comes from mean value theorem. By the assumption, we have

$$\begin{aligned}
f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle &\leq f(\boldsymbol{y}) \\
&\leq f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle + \int_0^1 tL\|\boldsymbol{y} - \boldsymbol{x}\|^2 dt \\
&\leq f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle + \frac{L}{2}\|\boldsymbol{y} - \boldsymbol{x}\|^2.
\end{aligned}$$

And the conclusion follows from Lemma 12.8.5. □

Chapter 13

Connection Laplacian In Nearly-Linear Time

13.1 Introduction

In this chapter, we introduce the *sparsified Cholesky* and *sparsified multigrid* algorithms for solving systems of linear equations. Two advantages of these algorithms over other nearly-linear time algorithms for solving systems of equations in Laplacian matrices [256, 243, 147, 146, 141, 57] are:

1. They give nearly-linear time algorithms for solving systems of equations in a much broader class of matrices—the connection Laplacians and Hermitian block diagonally dominant matrices. Connection Laplacians [238, 29] are a generalization of graph Laplacians that arise in many applications, including celebrated work on cryo-electron microscopy [237, 235, 275], phase retrieval [7, 186], and many image processing problems (e.g. [221, 16]). Previous algorithms for solving systems of equations in graph Laplacians cannot be extended to solve equations in connection Laplacians because the previous algorithms relied on some form of low stretch spanning trees, a concept that has no analog for the more general connection Laplacians.

2. They provide linear-sized approximate inverses of connection Laplacian matrices. That is, for every n-dimensional connection Laplacian $\boldsymbol{M}$, the sparsified Cholesky factorization algorithm produces an block-upper-triangular matrix $\boldsymbol{U}$ and block diagonal $\boldsymbol{D}$ with $O(n)$ nonzero entries so that $\boldsymbol{U}^T \boldsymbol{D}\, \boldsymbol{U}$ is a constant-factor approximation of $\boldsymbol{M}$. Such matrices $\boldsymbol{U}$ and $\boldsymbol{D}$ allow one to solve systems of equations in $\boldsymbol{M}$ to accuracy ε in time $O(m \log \varepsilon^{-1})$, where m is the number of nonzero entries of $\boldsymbol{M}$. Even for ordinary Laplacian matrices, the existence of such approximate inverses is entirely new. The one caveat of this result is that we do not yet know how to compute those approximate inverses in nearly linear time.

The sparsified Cholesky and sparsified multigrid algorithms work by sparsifying the matrices produced during Gaussian elimination. Recall that Cholesky factorization is the version of Gaussian elimination for symmetric matrices, and that the high running time of Gaussian elimination comes from *fill*—new nonzero matrix entries that are created by row operations. If Gaussian elimination never produced rows with a super-constant number of entries, then it would run in linear time. The sparsified Cholesky algorithm accelerates Gaussian elimination by sparsifying the rows that are produced by elimination, thereby guaranteeing that the elimination will be fast. Sparsified Cholesky is inspired by one of the major advances in algorithms for solving linear equations in Laplacian matrices—the Incomplete Cholesky factorization (ICC) [191]. However, ICC merely drops entries produced by elimination, whereas sparsification also carefully increases ones that remain. The difference is crucial, and is why ICC does not provide a nearly-linear time solver.

To control the error introduced by sparsification, we have to be careful not to do it too often. This means that our algorithm actually chooses a large set of rows and columns to eliminate at once, and then sparsifies the result. This is the basis of our first algorithms, which establish the existence of linear time solvers and linear sized approximate inverses, after precomputation. This precomputation

is analogous to the procedure of computing a matrix inverse: the approximate inverse takes much less time to apply than to compute.

To produce entire algorithms that run in nearly linear time (both to compute and apply the approximate inverse) requires a little more work. To avoid the work of computing the matrix obtained by eliminating the large set of rows and columns, we design a fast algorithm for approximating it quickly. This resulting algorithm produces a solver routine that does a little more work at each level, and so resembles a multigrid V-cycle [252]. We call the resulting algorithm the *sparsified multigrid*. We note that Krishnan, Fattal, and Szeliski [150] present experimental results from the use of a sparsification heuristic in a multigrid algorithm for solving problems in computer vision.

Our new algorithms are most closely related to the Laplacian solver recently introduced by Peng and Spielman [223]: unlike the other Laplacian solvers, they rely only on sparsification and do not directly rely on "support theory" preconditioners or any form of low stretch spanning trees. This is why our algorithms can solve a much broader family of linear systems. To sparsify without using graph theoretic algorithms, we employ the algorithm in Chapter 3 that allows us to sparsify a matrix by solving systems of equations in a subsampled matrix.

13.1.1 Connection Laplacians and Block DD Matrices

In this section, we define block diagonally dominant (bDD) matrices—the most general family of matrices such that the associated systems of linear equations can be solved by our algorithms. We begin by defining our motivating case: the connection Laplacians

Connection Laplacians may be thought of as a generalization of graph Laplacians where every vertex is associated with a vector, instead of a real number, and every edge is associated with a unitary matrix. Like graph Laplacians, they describe a natural quadratic form. Let $\boldsymbol{M}_{[i,j]} \in \mathbb{C}^{r\times r}$ be the unitary matrix associated with edge (i,j), and let $w_{i,j}$ be the (nonnegative) weight of edge (i,j). We require that $\boldsymbol{M}_{[i,j]} = \boldsymbol{M}^*_{[j,i]}$, where $*$ denotes the conjugate transpose. The quadratic form associated with this connection Laplacian is a function of vectors $\boldsymbol{v}^{(i)} \in \mathbb{C}^r$, one for each vertex i, that equals

$$\sum_{(i,j)\in E} \boldsymbol{w}_{i,j} \|\boldsymbol{v}^{(i)} - \boldsymbol{M}_{[i,j]}\boldsymbol{v}^{(j)}\|^2.$$

The matrix corresponding to this quadratic form is a block matrix with blocks $\boldsymbol{M}_{[i,j]}$. Most applications of the connection Laplacian require one to either solve systems of linear equations in this matrix, or to compute approximations of its smallest eigenvalues and eigenvectors. By applying the inverse power method (or inverse Lanczos), we can perform these eigenvector calculations by solving a logarithmic number of linear systems in the matrix (see [243, Section 7]).

The matrices obtained from the connection Laplacian are a special case of block diagonally dominant (bDD) matrices, which we now define. Throughout this chapter, we consider block matrices having entries in $\mathbb{C}^{r\times r}$, where $r > 0$ is a fixed integer. We say that $\boldsymbol{M} \in (\mathbb{C}^{r\times r})^{m\times n}$ has m block-rows, and n block-columns. For $i \in [m], j \in [n]$, we let $\boldsymbol{M}_{[i,j]} \in \mathbb{C}^{r\times r}$ denote the i,j-block in $\boldsymbol{M}$, and $\boldsymbol{M}_{[j]}$ denote the j^{th} block-column, i.e., $\boldsymbol{M}_{[j]} = [(\boldsymbol{M}_{[1,j]})^*, (\boldsymbol{M}_{[2,j]})^*, \ldots, (\boldsymbol{M}_{[m,j]})^*]^*$. For sets $F \subseteq [m], C \subseteq [n]$, we let $\boldsymbol{M}_{[F,C]} \in (\mathbb{C}^{r\times r})^{|F|\times|C|}$ denote the block-submatrix with blocks $\boldsymbol{M}_{[i,j]}$ for $i \in F, j \in C$. $\boldsymbol{M}$ is block-diagonal, if $\boldsymbol{M}_{[i,j]} = 0$ for $i \neq j$. *We emphasize that all computations are done over $\mathbb{C}$, not over a matrix group.*

Definition 13.1.1. A Hermitian block-matrix $\boldsymbol{M} \in (\mathbb{C}^{r\times r})^{n\times n}$ is block diagonally dominant (or bDD) if

$$\text{for all } i \in [n], \qquad \boldsymbol{M}_{[i,i]} \succeq \boldsymbol{I}_r \cdot \sum_{j:j\neq i} \|\boldsymbol{M}_{[i,j]}\|_2,$$

where $\boldsymbol{I}_r \in \mathbb{C}^{r\times r}$ denotes the identity matrix.

For the rest of this chapter, we use $\|\cdot\|$ to denote $\|\cdot\|_2$. Equivalently, a Hermitian $\boldsymbol{M}$ is a bDD matrix iff it can be written as $\boldsymbol{D} - \boldsymbol{A}$ where $\boldsymbol{D}$ is block-diagonal, and $\boldsymbol{D}_{[i,i]} \succeq \boldsymbol{I}_r \sum_j \|\boldsymbol{A}_{[i,j]}\|$ (see Lemma 13.6.1).
Throughout this chapter we treat r as a constant. The hidden dependence on r, of the running times of the algorithms we present, is polynomial. The major results of this chapter are the following.

Theorem 13.1.2 (Sparsified Multigrid). *There is an algorithm that, when given a bDD matrix $\boldsymbol{M}$ with n block rows and m nonzero blocks, produces a solver for $\boldsymbol{M}$ in $O(m \log n + n \log^{2+o(1)} n)$ work and $O(n^{o(1)})$ depth so that the solver finds ε-approximate solutions to systems of equations in $\boldsymbol{M}$ in $O((m + n \log^{1+o(1)} n) \log(1/\varepsilon))$ work and $O(\log^2 n \log\log n \log(1/\varepsilon))$ depth.*

Previously, the existence of nearly-linear time solvers was even unknown for the 1-dimensional case ($r = 1$) when the off-diagonal entries were allowed to be complex numbers.

We can use the above algorithm to find approximations of the smallest eigenvalues and eigenvectors of such matrices at an additional logarithmic cost.

Theorem 13.1.3 (Sparsified Cholesky). *For every bDD matrix $\boldsymbol{M}$ with n block-rows there exists a diagonal matrix $\boldsymbol{D}$ and an upper triangular matrix $\boldsymbol{U}$ with $O(n)$ nonzero blocks so that*

$$\boldsymbol{U}^T \boldsymbol{D}\, \boldsymbol{U} \approx_{3/4} \boldsymbol{M}.$$

Moreover, linear equations in $\boldsymbol{U}$, $\boldsymbol{U}^T$, and $\boldsymbol{D}$ can be solved with linear work in depth $O(\log^2 n)$, and these matrices can be computed in polynomial time.

These matrices allow one to solve systems of linear equations in $\boldsymbol{M}$ to ε-accuracy in parallel time $O(\log^2 n \log^{-1} \varepsilon)$ and work $O(m \log^{-1} \varepsilon)$. Results of this form were previously unknown even for graph Laplacians.

In the next two sections we explain the ideas used to prove these theorems. Proofs may be found in the appendices that follow.

13.2 Preliminaries

We say that $\boldsymbol{A}$ is an ε-approximation of $\boldsymbol{B}$, written $\boldsymbol{A} \approx_\varepsilon \boldsymbol{B}$, if $e^{\varepsilon}\boldsymbol{B} \succcurlyeq \boldsymbol{A} \succcurlyeq e^{-\varepsilon}\boldsymbol{B}$. This relation is symmetric. We say that $\tilde{\boldsymbol{x}}$ is an ε-approximate solution to the system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ if $\|\tilde{\boldsymbol{x}} - \boldsymbol{A}^{-1}\boldsymbol{b}\|_{\boldsymbol{A}} \leq \varepsilon \|\boldsymbol{x}\|_{\boldsymbol{A}}$, where $\|\boldsymbol{x}\|_{\boldsymbol{A}} = (\boldsymbol{x}^T \boldsymbol{A}\boldsymbol{x})^{1/2}$. If $\varepsilon < 1/2$, $\boldsymbol{A} \approx_\varepsilon \boldsymbol{B}$ and $\boldsymbol{B}\tilde{\boldsymbol{x}} = \boldsymbol{b}$, then $\tilde{\boldsymbol{x}}$ is a 2ε approximate solution to $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$.

Fact 13.2.1. *If $\boldsymbol{A} \approx_\varepsilon \boldsymbol{B}$ and $\boldsymbol{B} \approx_\delta \boldsymbol{C}$, then $\boldsymbol{A} \approx_{\varepsilon+\delta} \boldsymbol{C}$.*

We now address one technicality of dealing with *bDD* matrices: it is not immediate whether or not a *bDD* matrix is singular. Moreover, if it is singular, the structure of its null space is not immediately clear either. Throughout the rest of this chapter, we will consider *bDD* matrices to which a small multiple of the identity have been added. These matrices will be nonsingular. To reduce the problem of solving equations in a general *bDD* matrix $\boldsymbol{M}$ to that of solving equations in a nonsingular matrix, we require an estimate of the smallest nonzero eigenvalue of $\boldsymbol{M}$.

Fact 13.2.2. *Suppose that all nonzero eigenvalues of $\boldsymbol{M}$ are at least μ and $\boldsymbol{Z} \approx_\varepsilon (\boldsymbol{M} + \varepsilon\mu\boldsymbol{I})^{-1}$ for some $0 < \varepsilon < 1/2$. Given any $\boldsymbol{b}$ such that $\boldsymbol{M}\boldsymbol{x} = \boldsymbol{b}$ for some $\boldsymbol{x}$, we have*

$$\|\boldsymbol{x} - \boldsymbol{Z}\boldsymbol{b}\|_{\boldsymbol{M}}^2 \leq 6\varepsilon \|x\|_{\boldsymbol{M}}^2.$$

Hence, we can solve systems in $\boldsymbol{M}$ by approximately solving systems in $\boldsymbol{M} + \varepsilon\mu\boldsymbol{I}$. Any lower bound μ on the smallest nonzero eigenvalue of $\boldsymbol{M}$ will suffice. It only impacts the running times of our algorithms in the numerical precision with which one must carry out the computations.

The above fact allows us to solve systems in $\boldsymbol{M}$ that have a solution. If $\boldsymbol{M}$ is singular and we want to apply its pseudoinverse (that is, to find the closest solution in the range of $\boldsymbol{M}$), we can do so by pre and post multiplying by $\boldsymbol{M}$ to project onto its range. The resulting algorithm, which is implicit in the following fact, requires applying a solver for $\boldsymbol{M}$ three times. It also takes $O(\log \kappa)$ times as long to run, where κ is the finite condition number[1] of $\boldsymbol{M}$.

Fact 13.2.3. *Let κ be an upper bound on the finite condition number of $\boldsymbol{M}$. Given an error parameter $0 < \varepsilon < 1$, let $\delta = \varepsilon/56\kappa^3$. If all nonzero eigenvalues of $\boldsymbol{M}$ are at least μ, and if $\boldsymbol{Z} \approx_\delta (\boldsymbol{M} + \varepsilon\mu\boldsymbol{I})^{-1}$, then*

$$\boldsymbol{M}\boldsymbol{Z}^3\boldsymbol{M} \approx_{4\varepsilon} \boldsymbol{M}^+.$$

13.3 Overview of the Algorithms

We index the block rows and columns of a block matrix by a set of vertices (or indices) V. When we perform an elimination, we eliminate a set $F \subset V$, and let $C = V - F$. Here, F stands for "fine" and C stands for "coarse". In contrast with standard multigrid methods, we will have $|F| \leq |C|$.

To describe block-Cholesky factorization, we write the matrix $\boldsymbol{M}$ with the rows and columns in F first:

$$\boldsymbol{M} = \begin{bmatrix} \boldsymbol{M}_{[F,F]} & \boldsymbol{M}_{[F,C]} \\ \boldsymbol{M}_{[C,F]} & \boldsymbol{M}_{[C,C]} \end{bmatrix}.$$

Cholesky factorization writes the inverse of this matrix as

$$\boldsymbol{M}^{-1} = \begin{bmatrix} \boldsymbol{I} & -\boldsymbol{M}_{[F,F]}^{-1}\boldsymbol{M}_{[F,C]} \\ 0 & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{M}_{[F,F]}^{-1} & 0 \\ 0 & Sc(\boldsymbol{M}, F)^{-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{I} & 0 \\ -\boldsymbol{M}_{[C,F]}\boldsymbol{M}_{[F,F]}^{-1} & \boldsymbol{I} \end{bmatrix}, \tag{13.1}$$

where

$$Sc(\boldsymbol{M}, F) \stackrel{\text{def}}{=} \boldsymbol{M}_{[C,C]} - \boldsymbol{M}_{[C,F]}\boldsymbol{M}_{[F,F]}^{-1}\boldsymbol{M}_{[F,C]}$$

is the Schur complement of $\boldsymbol{M}$ with respect to F.

Our algorithms rely on two fundamental facts about *bDD* matrices: that the Schur complement of a *bDD* matrix is a *bDD* matrix (Lemma 13.6.3) and that one can sparsify *bDD* matrices. The following theorem is implicit in [46].

Theorem 13.3.1. *For every $\varepsilon \leq 1$, every bDD matrix $\boldsymbol{M}$ with n r-dimensional block rows and columns can be ε-approximated by a bDD matrix having at most $10nr/\varepsilon^2$ nonzero blocks.*

We use the identity (13.1) to reduce the problem of solving a system of equations in $\boldsymbol{M}$ to that of solving equations in its Schur complement. The easiest part of this is multiplication by $\boldsymbol{M}_{[C,F]}$: the time is proportional to the number of nonzero entries in the submatrix, and we can sparsify $\boldsymbol{M}$ to guarantee that this is small.

The costlier part of the reduction is the application of the inverse of $\boldsymbol{M}_{[F,F]}$ three times. This would be fast if $\boldsymbol{M}_{[F,F]}$ were block-diagonal, which corresponds to F being an independent set. We cannot find a sufficiently large independent set F, but we can find a large set that is almost independent. This results in a matrix $\boldsymbol{M}_{[F,F]}$ that is well approximated by its diagonal, and thus linear equations

[1] The finite condition number is the ratio of the largest singular value to the smallest *nonzero* singular value.

in this matrix can be quickly solved to high accuracy by a few Jacobi iterations (see Theorem 13.3.1 and Section 13.9).

We can prove the existence of linear sized approximate inverses by explicitly computing $Sc(\boldsymbol{M}, F)$, sparsifying it, and then recursively applying the algorithm just described. To make this algorithm efficient, we must compute a sparse approximation to $Sc(\boldsymbol{M}, F)$ without constructing $Sc(\boldsymbol{M}, F)$. This is the problem of *spectral vertex sparsification*, and we provide a fast algorithm for this task in Sections 13.3.4 and 13.11.

13.3.1 Schur Complement Chains

We encode this recursive algorithm by a *Schur complement chain* (SCC). An SCC defines a linear operator that can be used to approximately solve equations in an initial matrix $\boldsymbol{M}^{(0)}$. If the *bDD* matrix $\boldsymbol{M}$ is sparse, then it is the same as $\boldsymbol{M}^{(0)}$; if not, then $\boldsymbol{M}^{(0)}$ is a sparse approximation to $\boldsymbol{M}$. Let F_1 be the first set of vertices eliminated, $\boldsymbol{M}^{(1)}$ a sparse approximation to the Schur complement of $\boldsymbol{M}^{(0)}$ with respect to F_1, and $\boldsymbol{Z}^{(1)}$ an operators that approximates the inverse of $\boldsymbol{M}^{(0)}_{[F_1,F_1]}$.

Definition 13.3.2 (Schur Complement Chain)**.** An ε-Schur complement chain (ε-SCC) for a matrix $\boldsymbol{M}^0$ indexed by vertex set V is a sequence of operators and subsets,

$$\left((\boldsymbol{M}^{(1)}, \boldsymbol{Z}^{(1)}), \ldots, (\boldsymbol{M}^{(d)}, \boldsymbol{Z}^{(d)}); F_1, \ldots, F_d\right),$$

so that for $C_0 = V$ and $C_{i+1} = C_i \setminus F_{i+1}$, $|C_d| \leq 1000$ and for $1 \leq i \leq d$,

$$\boldsymbol{M}^{(i)} \approx_{\varepsilon_i} Sc\left(\boldsymbol{M}^{(i-1)}, F_i\right) \quad \text{and} \quad 0 \preceq (\boldsymbol{Z}^{(i)})^{-1} - \boldsymbol{M}^{(i-1)}_{[F_i,F_i]} \preceq \varepsilon_i \cdot Sc\left(\boldsymbol{M}^{(i-1)}, C_i\right).$$

The algorithm ApplyChain, described in Section 13.7, applies an SCC to solve equations in $\boldsymbol{M}^{(0)}$ in the natural way and satisfies the following guarantee.

Lemma 13.3.1. Consider an ε-SCC for $\boldsymbol{M}^{(0)}$ where $\boldsymbol{M}^{(i)}$ and $\boldsymbol{Z}^{(i)}$ can be applied to a vector in work $W_{\boldsymbol{M}^{(i)}}, W_{\boldsymbol{D}^{(i)}}$ and depth $D_{\boldsymbol{M}^{(i)}}, D_{\boldsymbol{Z}^{(i)}}$ respectively.

The algorithm ApplyChain $\left((\boldsymbol{M}^{(1)}, \boldsymbol{Z}^{(1)}), \ldots, (\boldsymbol{M}^{(d)}, \boldsymbol{Z}^{(d)}); F_1, \ldots, F_d; \boldsymbol{b}\right)$ corresponds to a linear operator $\boldsymbol{W}$ acting on $\boldsymbol{b}$ such that

1. $\boldsymbol{W}^{-1} \approx_{\sum_{i=1}^{d} 2\varepsilon_i} \boldsymbol{M}^{(0)}$, and
2. for any vector $\boldsymbol{b}$, ApplyChain $\left((\boldsymbol{M}^{(1)}, \boldsymbol{Z}^{(1)}), \ldots, (\boldsymbol{M}^{(d)}, \boldsymbol{Z}^{(d)}); F_1, \ldots, F_d; \boldsymbol{b}\right)$ runs in $O\left(\sum_{i=1}^{d} (D_{\boldsymbol{M}^{(i)}} + D_{\boldsymbol{Z}^{(i)}})\right)$ depth and $O\left(\sum_{i=1}^{d} (W_{\boldsymbol{M}^{(i)}} + W_{\boldsymbol{Z}^{(i)}})\right)$ work.

For an ε-SCC chain for $\boldsymbol{M}^{(0)}$, we define the depth and work of the chain to be the work and depth required by ApplyChain to apply the SCC.

13.3.2 Choosing F_i: α-bDD Matrices

We must choose the set of vertices F_i so that we can approximate the inverse of $\boldsymbol{M}^{(i)}_{[F_i,F_i]}$ by an operator $\boldsymbol{Z}^{(i)}$ that is efficiently computable. We do this by requiring that the matrix $\boldsymbol{M}^{(i)}_{[F_i,F_i]}$ be α-block diagonally dominant (α-bDD), a term that we now define.

Definition 13.3.3. A Hermitian block-matrix $\boldsymbol{M}$ is α-bDD if

$$\forall i, \quad \boldsymbol{M}_{[i,i]} \succcurlyeq (1+\alpha)\boldsymbol{I}_r \cdot \textstyle\sum_{j:j\neq i} \|\boldsymbol{M}_{[i,j]}\|. \tag{13.2}$$

We remark that a 0-bDD matrix is simply a bDD matrix. In particular, for $r = 1$, Laplacian matrices are 0-bDD.

By picking a subset of rows at random and discarding those that violate condition (13.2), the algorithm bDDSubset (described in Section 13.8) finds a linear sized subset F of the block-rows of a bDD matrix $\boldsymbol{M}$ so that $\boldsymbol{M}_{[F,F]}$ is α-bDD.

Lemma 13.3.2. Given a bDD matrix $\boldsymbol{M}$ with n block-rows, and an $\alpha \geq 0$, bDDSubset computes a subset F of size least $n/(8(1+\alpha))$ such that $\boldsymbol{M}_{[F,F]}$ is α-bDD. It runs in runs in $O(m)$ expected work and $O(\log n)$ expected depth, where m is the number of nonzero blocks in $\boldsymbol{M}$.

We can express an α-bDD matrix as a sum of a block diagonal matrix and a bDD matrix so that it is well-approximated by the diagonal.

Lemma 13.3.3. Every α-bDD matrix $\boldsymbol{M}$ can be written in the form $\boldsymbol{X} + \boldsymbol{L}$ where $\boldsymbol{X}$ is block-diagonal, $\boldsymbol{L}$ is bDD, and $\boldsymbol{X} \succcurlyeq \frac{\alpha}{2}\boldsymbol{L}$.

As $\boldsymbol{L}$ is positive semidefinite, $(1 + 2/\alpha)\,\boldsymbol{X} \succcurlyeq \boldsymbol{M} \succcurlyeq \boldsymbol{X}$, which means that $\boldsymbol{X}$ is a good approximation of $\boldsymbol{M}$ when α is reasonably big. As block-diagonal matrices like $\boldsymbol{X}$ are easy to invert, systems in these well-conditioned matrices can be solved rapidly using preconditioned iterative methods. In Section 13.9, we show that a variant of Jacobi iteration provides an operator that satisfies the requirements of Definition 13.3.2.

Theorem 13.3.1. *Let $\boldsymbol{M}$ be a bDD matrix with index set V, and let $F \subseteq V$ such that $\boldsymbol{M}_{[F,F]}$ is α-bDD for some $\alpha \geq 4$, and has m_{FF} nonzero blocks. The algorithm* Jacobi$(\varepsilon, \boldsymbol{M}_{[F,F]}, \boldsymbol{b})$ *acts as a linear operator $\boldsymbol{Z}$ on $\boldsymbol{b}$ that satisfies*

$$0 \preceq (\boldsymbol{Z})^{-1} - \boldsymbol{M}_{[F,F]} \preceq \varepsilon \cdot Sc(\boldsymbol{M}, V \setminus F).$$

The algorithm takes $O(m_{FF} \log(\frac{1}{\varepsilon}))$ work and $O(\log n \log(\frac{1}{\varepsilon}))$ depth.

We now explain how Theorems 13.3.1 and 13.3.1 allow us to construct Schur complement chains that can be applied in nearly linear time. We optimize the construction in the next section.

Theorem 13.3.1 tells us that there is a *bDD* matrix $\boldsymbol{M}^{(0)}$ with $O(n/\varepsilon^2)$ nonzero blocks that that ε-approximates $\boldsymbol{M}$, and that for every i there is a *bDD* matrix $\boldsymbol{M}^{(i+1)}$ with $O(|C_i|/\varepsilon^2)$ nonzero blocks that is an ε-approximation of $Sc\left(\boldsymbol{M}^{(i)}, F_i\right)$. We will pick ε later. Lemma 13.3.2 provides a set F_i containing a constant fraction of the block-rows of $\boldsymbol{M}^{(i)}$ so that $\boldsymbol{M}^{(i)}_{[F_i,F_i]}$ is 4-bDD. Theorem 13.3.1 then provides an operator that solves systems in $\boldsymbol{M}^{(i)}_{[F_i,F_i]}$ to ε accuracy in time $O(\log \varepsilon^{-1})$ times the number of nonzero entries in $\boldsymbol{M}^{(i)}_{[F_i,F_i]}$. This is at most the number of nonzero entries in $\boldsymbol{M}^{(i)}$, and thus at most $O(|C_i|/\varepsilon^2)$. As each F_i contains at least a constant fraction of the rows of $\boldsymbol{M}^{(i)}$, the depth of the recursion, d, is $O(\log n)$. Thus, we can obtain constant accuracy by setting $\varepsilon = \Theta(1/\log n)$. The time required to apply the resulting SCC would thus be $O(n \log^2 n \log\log n)$.

We can reduce the running time by setting ε_1 to a constant and allowing it to shrink as i increases. For example, setting $\varepsilon_i = 1/2(i+1)^2$ results in a linear time algorithm that produces a constant-factor approximation of the inverse of $\boldsymbol{M}$. We refine this idea in the next section.

13.3.3 Linear Sized Approximate Inverses

In this section we sketch the proof of Theorem 13.1.3, which tells us that every *bDD* matrix has a linear-sized approximate inverse. The rest of the details appear in Section 13.10.

The linear-sized approximate inverse of a *bDD* matrix $\boldsymbol{M}$ with n block rows and columns is provided by a 3/4-approximate of the form $\boldsymbol{U}^T \boldsymbol{D}\, \boldsymbol{U}$ where $\boldsymbol{D}$ is block diagonal and $\boldsymbol{U}$ is block upper-triangular

and has $O(n)$ nonzero blocks. As systems of linear equations in block-triangular matrices like $\boldsymbol{U}$ and $\boldsymbol{U}^T$ can be solved in time proportional to their number of nonzero blocks, this provides a linear time algorithm for computing a 3/4 approximation of the inverse of $\boldsymbol{M}$. By iteratively refining the solutions provided by the approximate inverse, this allows one to find ε-accurate solutions to systems of equations in $\boldsymbol{M}$ in time $O(m \log \varepsilon^{-1})$.

The matrix $\boldsymbol{U}$ that we construct has meta-block structure that allows solves in $\boldsymbol{U}$ and $\boldsymbol{U}^T$ to be performed with linear work and depth $O(\log^2 n)$. This results in a parallel algorithm for solving equations in $\boldsymbol{M}$ in work $O(m \log \varepsilon^{-1})$ and depth $O(\log^2 n \log \varepsilon^{-1})$. We remark that with some additional work (analogous to Section 7 of [160]), one could reduce this depth to $O(\log n \log\log n \log \varepsilon^{-1})$.

The key to constructing $\boldsymbol{U}$ is realizing that the algorithm Jacobi corresponds to multiplying by the matrix $\boldsymbol{Z}^{(k)}$ defined in equation (13.7). Moreover, this matrix is a polynomial in $\boldsymbol{X}$ and $\boldsymbol{L}$ of degree $\log(3/\varepsilon)$, where $\boldsymbol{M}^{(i)}_{[F_i,F_i]} = \boldsymbol{X} + \boldsymbol{L}$ where $\boldsymbol{L}$ is bDD, and $\boldsymbol{X}$ block diagonal such that $\boldsymbol{X} \succcurlyeq 2\boldsymbol{L}$. To force $\boldsymbol{Z}^k$ to be a sparse matrix, we require $\boldsymbol{L}$ be sparse.

If we use the algorithm bSDDSubset to choose F_i, then $\boldsymbol{L}$ need not be sparse. However, this problem is easily remedied by forbidding algorithm bSDDSubset from choosing any vertex of more than twice average degree. Thus, we can ensure that all $\boldsymbol{Z}^{(i)}$ are sparse.

Lemma 13.3.4. *For every bDD matrix $\boldsymbol{M}$ and every $\alpha \geq 0$, there is a subset F of size at least $\frac{n}{16(1+\alpha)}$ such that $M_{[F,F]}$ is α-bDD and the number of nonzeros blocks in each block-row of F at most twice the average number of nonzero blocks in a block-row of $\boldsymbol{M}$.*

Proof. Discard every block-row of $\boldsymbol{M}$ that has more than twice the average number of nonzeros blocks per row-block. Then remove the corresponding row blocks. The remaining matrix has dimension at least $n/2$. We can now use Lemma 13.3.2 to find an α-bDD submatrix. □

We obtain the $\boldsymbol{U}^T\boldsymbol{D}\,\boldsymbol{U}$ factorization by applying the inverse of the factorization (13.1):

$$\boldsymbol{M} = \begin{bmatrix} \boldsymbol{I} & 0 \\ \boldsymbol{M}^{-1}_{[F,F]}\boldsymbol{M}_{[F,C]} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{M}_{[F,F]} & 0 \\ 0 & Sc(\boldsymbol{M}, F) \end{bmatrix} \begin{bmatrix} \boldsymbol{I} & \boldsymbol{M}_{[C,F]}\boldsymbol{M}^{-1}_{[F,F]} \\ 0 & \boldsymbol{I} \end{bmatrix}.$$

In the left and right triangular matrices we replace $\boldsymbol{M}^{-1}_{[F,F]}$ with the polynomial we obtain from Jacobi. In the middle matrix, it suffices to approximate $\boldsymbol{M}_{[F,F]}$ by a block diagonal matrix, and $Sc(\boldsymbol{M}, F)$ by a factorization of its sparse approximation given by Theorem 13.3.1. The details, along with a careful setting of ε_i, are carried out in Section 13.10.

13.3.4 Spectral Vertex Sparsification Algorithm

In this section, we outline a procedure ApproxSchur that efficiently approximates the Schur complement of a bDD matrix $\boldsymbol{M}$ w.r.t. a set of indices F s.t. $\boldsymbol{M}_{[F,F]}$ is α-bDD. The following lemma summarizes the guarantees of ApproxSchur.

Lemma 13.3.4. Let $\boldsymbol{M}$ be a bDD matrix with index set V, and m nonzero blocks. Let $F \subseteq V$ be such that $\boldsymbol{M}_{[F,F]}$ is α-bDD for some $\alpha \geq 4$. The algorithm ApproxSchur$(\boldsymbol{M}, F, \varepsilon)$, returns a matrix $\widetilde{\boldsymbol{M}}_{SC}$ s.t.

1. $\widetilde{\boldsymbol{M}}_{SC}$ has $O(m(\varepsilon^{-1}\log\log\varepsilon^{-1})^{O(\log\log\varepsilon^{-1})})$ nonzero blocks, and
2. $\widetilde{\boldsymbol{M}}_{SC} \approx_\varepsilon Sc(\boldsymbol{M}, F)$,

in $O(m(\varepsilon^{-1}\log\log\varepsilon^{-1})^{O(\log\log\varepsilon^{-1})})$ work and $O(\log n(\log\log\varepsilon^{-1}))$ depth.

We sketch a proof of the above lemma in this section. A complete proof and pseudocode for ApproxSchur are given in Section 13.11.

First consider a very simple special case: where F is a singleton, $F = \{i\}$. Let $C = V \setminus F$ be the remaining indices.

$$Sc(\boldsymbol{M}, i) = \boldsymbol{M}_{[C,C]} - \boldsymbol{M}_{[C,i]}\boldsymbol{M}_{[i,i]}^{-1}\boldsymbol{M}_{[i,C]}$$

Thus, if $\boldsymbol{M}_{[C,i]}$ has k nonzero blocks, $Sc(\boldsymbol{M}, i)$ could have k^2 additional nonzero blocks compared to $\boldsymbol{M}_{[C,C]}$, potentially making it dense. If $\boldsymbol{M}$ were a graph Laplacian, then $\boldsymbol{M}_{[C,i]}\boldsymbol{M}_{[i,i]}^{-1}\boldsymbol{M}_{[i,C]}$ would represent the adjacency matrix of a weighted clique. In Section 13.14, we construct weighted expanders that allow us to ε-approximate $Sc(\boldsymbol{M}, i)$ in this case using $m + O(k\varepsilon^{-4})$ edges. In Section 13.11.4, we show how to use such weighted expanders to sparsify $Sc(\boldsymbol{M}, i)$ when $\boldsymbol{M}$ is a bDD matrix.

This reduction can also be performed in parallel. If F is such that $\boldsymbol{M}_{[F,F]}$ is block diagonal, we can approximate $Sc(\boldsymbol{M}, F)$ by expressing $\boldsymbol{M}_{[C,i]}\boldsymbol{M}_{[i,i]}^{-1}\boldsymbol{M}_{[i,C]}$ as $\sum_{i\in F}\boldsymbol{M}_{[C,i]}\boldsymbol{M}_{[i,i]}^{-1}\boldsymbol{M}_{[i,C]}$, and using $|F|$ weighted expanders. However, $\boldsymbol{M}_{[F,F]}$ may not be diagonal. Instead, we give a procedure SchurSquare that generates $\boldsymbol{M}'$ that is better approximated by its diagonal.

Invoking SchurSquare a few times leads a sequence of matrices $\boldsymbol{M}^{(0)}_{[F,F]}, \boldsymbol{M}^{(1)}_{[F,F]}, \ldots, \boldsymbol{M}^{(i)}_{[F,F]}$. We will show that $\boldsymbol{M}^{(i)}_{[F,F]}$ is ε-approximated by its diagonal and we call the procedure LastStep to approximate $Sc\left(\boldsymbol{M}^{(i)}, F\right)$. An additional caveat is that replacing $\boldsymbol{M}^{(i)}_{[F,F]}$ by its diagonal at this step gives errors that are difficult to bound. We discuss the correct approximation below

SchurSquare is based on a *squaring* identity for matrix inverse developed in [223]. Given a splitting of $\boldsymbol{M}_{[F,F]}$ into $\boldsymbol{D} - \boldsymbol{A}$, where $\boldsymbol{D}$ is block-diagonal, and $\boldsymbol{A}$ has its diagonal blocks as zero, it relies on the fact that the matrix

$$\boldsymbol{M}_2 = \frac{1}{2}\begin{bmatrix} \boldsymbol{D} - \boldsymbol{A}\boldsymbol{D}^{-1}\boldsymbol{A} & \boldsymbol{M}_{[F,C]} + \boldsymbol{A}\boldsymbol{D}^{-1}\boldsymbol{M}_{[F,C]} \\ \boldsymbol{M}_{[C,F]} + \boldsymbol{M}_{[C,F]}\boldsymbol{D}^{-1}\boldsymbol{A} & 2\boldsymbol{M}_{[C,C]} - \boldsymbol{M}_{[C,F]}\boldsymbol{D}^{-1}\boldsymbol{M}_{[F,C]} \end{bmatrix} \tag{13.3}$$

satisfies $Sc(\boldsymbol{M}, F) = Sc(\boldsymbol{M}_2, F)$. Furthermore, we can show that if $\boldsymbol{M}$ is α-bDD, $\boldsymbol{D} - \boldsymbol{A}\boldsymbol{D}^{-1}\boldsymbol{A}$ is α^2-bDD, which indicates that the block on $[F, F]$ rapidly approaches being diagonal.

As $\boldsymbol{M}_2$ may be dense, we construct a sparse approximation to it. Since $\boldsymbol{D}$ is diagonal, we can construct sparse approximations to the blocks $(\boldsymbol{M}_2)_{[F,F]}$ and $(\boldsymbol{M}_2)_{[C,C]}$ in a manner analogous to the case of diagonal $\boldsymbol{M}_{[F,F]}$. Similarly, we use bipartite expanders to construct sparse approximations to $(\boldsymbol{M}_2)_{[C,F]}$ and $(\boldsymbol{M}_2)_{[F,C]}$.

Our sequence of calls to SchurSquare terminates with $\boldsymbol{M}^{(i)}_{[F,F]}$ being roughly ε^{-1}-bDD. We then return LastStep$(\boldsymbol{M}^{(i)}, F, \varepsilon^{-1}, \varepsilon)$. As mentioned above, we cannot just replace $\boldsymbol{M}^{(i)}_{[F,F]}$ by its diagonal. Instead, LastStep performs one step of *squaring* similar to SchurSquare with a key difference: Rather than expressing $\boldsymbol{M}^{(i)}_{[F,F]}$ as $\boldsymbol{D} - \boldsymbol{A}$, it expresses it as $\boldsymbol{X} + \boldsymbol{L}$, where $\boldsymbol{X}$ is block-diagonal, and $\boldsymbol{L}$ is just barely bDD. With this splitting, it constructs $\boldsymbol{M}^{(last)}$ after performing one iteration similar to Eq. (13.3). After this step, it replaces the $\boldsymbol{M}^{(last)}_{[F,F]}$ block with the block-diagonal matrix $\boldsymbol{X}$. Again, we directly produce sparse approximations to $\boldsymbol{M}^{(last)}$ and its Schur complement via weighted (bipartite) expanders. A precise description and proofs are given in Section 13.11.2.

13.3.5 Sparsifying bDD Matrices

The main technical hurdle left to address is how we sparsify bDD matrices. We to do this both to approximate $\boldsymbol{M}$ by $\boldsymbol{M}^{(0)}$, if $\boldsymbol{M}$ is not already sparse, and to ensure that all the matrices $\boldsymbol{M}^{(i)}$ remain sparse. While the spectral vertex sparsification algorithm described in the previous section allows us

to compute an approximation to a Schur complement $Sc\left(\boldsymbol{M}^{(i)}, F_i\right)$, it is sparse only when $\boldsymbol{M}^{(i)}$ is already sparse. As we iteratively apply this procedure, the density of the matrices produced will grow unacceptably. We overcome this problem by occasionally applying another sparsification routine that substantially decreases the number of nonzero blocks. The cost of this sparsification routine is that it requires solving systems of equations in sparse bDD matrices. We, of course, do this recursively.

Our sparsification procedure begins by generalizing the observation that graph Laplacians can be sparsified by sampling edges with probabilities determined by their effective resistances [242]. There is a block analog of leverage scores (13.36) that provides probabilities of sampling blocks so that the resulting sampled matrix approximates the original and has $O(n \log n)$ nonzero blocks with high probability. To compute this block analog of leverage scores we employ the procedure developed in Chapter 3.

Once we generalize their results to block matrices, they show that we can obtain sufficiently good estimates of the block leverage scores by computing leverage scores in a bDD matrix obtained by randomly subsampling blocks of the original. The block leverage scores in this matrix are obtained by solving a logarithmic number of linear equations in this subsampled matrix. Thus, our sparsification procedure requires constructing a solver for a subsampled matrix and then applying that solver a logarithmic number of times. We compute this solver recursively.

There is a tradeoff between the number of nonzero blocks in the subsampled system and in the resulting approximation of the original matrix. If the original matrix has m nonzero blocks and we subsample to a system of m/K nonzero blocks, then we obtain an ε-approximation of the original matrix with $O(K\varepsilon^{-2} n \log n)$ nonzero blocks.

The details of the analysis of the undersampling procedure appear in Section 13.12.

13.3.6 Our Algorithm

We now explain how we prove Theorem 13.1.2. The details supporting this exposition appear in Section 13.13. Our main goal is to control the density of the Schur complement chain as we repeatedly invoke Lemma 13.3.4.

Starting from some $\boldsymbol{M}^{(0)}$, we compute sets F_i (via calls to bDDSubset), approximate solvers $\boldsymbol{Z}^{(i)}$ (via Jacobi), and approximations of Schur complements $\boldsymbol{M}^{(i)}$ (via ApproxSchur), until we obtain a matrix $\boldsymbol{M}^{(i)}$ such that its dimension is a smaller than that of $\boldsymbol{M}^{(0)}$ by a large constant factor (like 4). While the dimension of $\boldsymbol{M}^{(i)}$ is much smaller, its number of nonzero blocks is potentially larger by an even larger factor. We use the procedure described in the previous section to sparsify it. This sparsification procedure produces a sparse approximation of the matrix at the cost of solving systems of equations in a subsampled version of that matrix. Some care is required to balance the cost of the resulting recursion.

We now sketch an analysis of a nearly linear time algorithm that results from a simple choice of parameters. We optimize the parameter choice and analysis in Section 13.13. Let n be the dimension of $\boldsymbol{M}^{(0)}$ and let m be its number of nonzero blocks. To begin, assume that $m \leq n\Delta \log^3 n$, for a Δ to be specified later. We call this the *sparse* case, and address the case of dense $\boldsymbol{M}$ later.

We consider fixing $\varepsilon_i = c/\log n$ for all i, for some constant c. As the depth of the Schur complement chain is $O(\log n)$, this results in a solver with constant accuracy. A constant number of iterations of the procedure described above are required to produce an $\boldsymbol{M}^{(i)}$ whose dimension is a factor of 4 smaller than $\boldsymbol{M}^{(0)}$. Lemma 13.3.4 tells us that the edge density of this $\boldsymbol{M}^{(i)}$ is potentially higher than that of $\boldsymbol{M}^{(0)}$ by a factor of

$$O\left(\left(\varepsilon^{-1} \log\log \varepsilon^{-1}\right)^{O\left(\log\log \varepsilon^{-1}\right)}\right) = \exp\left(O(\log\log^2 n)\right) = n^{o(1)}.$$

Set Δ to be this factor. We use the sparsification procedure from the previous section to guarantee that no matrix in the chain has density higher than that of this matrix, which is upper bounded by $(m/n)\Delta = \Delta^2 \log^3 n$.

Setting $K = 2\Delta$, the subsampling produces a matrix of density half that of $\boldsymbol{M}^{(0)}$, and it produces a sparse approximation of $\boldsymbol{M}^{(i)}$ of density $O(K\varepsilon_i^{-2} \log n) \leq O(\Delta \log^3 n)$, which, by setting constants appropriately, we can also force to be half that of $\boldsymbol{M}^{(0)}$. In order to perform the sparsification procedure, we need to construct a Schur complement chain for the subsampled matrix, and then use it to solve $O(\log n)$ systems of linear equations. The cost of using this chain to solve equations in the subsampled system is at most $O(n\Delta^2 \log^4 n)$, and the cost of using the solutions to these equations to sparsify $\boldsymbol{M}^{(i)}$ is $O(m \log n)$. The cost of the calls to bDDSubset and ApproxSchur are proportional to the number of edges in the matrices, which is $O(n\Delta \log^4 n)$.

We repeat this procedure all the way down the chain, only using sparsification when the dimension of $\boldsymbol{M}^{(i)}$ shrinks by a factor of 4. Since none of the matrices that we generate have density higher than $\Delta \log^3 n$, we remain in the sparse case. Let $T_{sparse}(n)$ be the time required to construct a solver chain on systems of size n with $m \leq n\Delta \log^3 n$. We obtain the following recurrence

$$T_{sparse}(n) \leq 2T_{sparse}(n/4) + n\Delta^2 \log^4 n + m \log n + m\Delta \leq 2T_{sparse}(n/4) + n\Delta^2 \log^4 n + n\Delta \log n + n\Delta^2,$$

which gives

$$T(n)_{sparse} \leq O(n\Delta^2 + n\Delta^2 \log^4 n) \leq n^{1+o(1)}.$$

To handle the case of dense $\boldsymbol{M}$, we repeatedly sparsify while keeping n fixed until we obtain a matrix with fewer than $n\Delta \log^3 n$ edges, at which point we switch to the algorithm described above. The running time of this algorithm on a graph with m edges, $T_{dense}(m)$, satisfies the recurrence

$$T_{dense}(m) \leq \begin{cases} T_{sparse}(n) & \text{if } m \leq n\Delta \log^3 n, \text{ and} \\ 2T_{dense}\left(m/2\right) + n\Delta^2 \log^4 n + m \log n + m\Delta & \text{otherwise.} \end{cases}$$

Thus $T_{dense}(m)$ is upper bounded bounded by $O(mn^{o(1)} + n^{1+o(1)})$.

We tighten this bound in Section 13.13 to prove Theorem 13.1.2 by carefully choosing the parameters to accompany a sequence $\boldsymbol{\varepsilon}$ that starts constant and decreases slowly.

13.4 Summary

We introduce a new approach to solving systems of linear equations that gives the first nearly linear time algorithms for solving systems proof that connection Laplacians have linear-sized approximate inverses. This was unknown even for graph Laplacians.

Our algorithms build on ideas introduced in [223] and are a break from those used in the previous work on solving systems of equations in graph Laplacians [256, 243, 147, 146, 141, 57]. Those algorithms all rest on *support theory* [34], originally introduced by Vaidya [256], and rely on the fact that the Laplacian of one edge is approximated by the Laplacian of a path between its endpoints. No analogous fact is true for connection Laplacians, even those with complex entries for $r = 1$.

Instead, our algorithms rely on many new ideas, the first being that of sparsifying the matrices that appear during elimination. Other critical ideas are finding α-bDD subsets of vertices to eliminate in bulk, approximating Schur complements without computing them explicitly, and the use of sub-sampling to sparsify in a recursive fashion. To efficiently compute approximations of the Schur complements, we introduce a new operation that transforms a matrix into one with the same Schur complement but a much better conditioned upper block (13.3). To obtain the sharp bounds in Theorem 13.1.2, we exploit a new linear-time algorithm for constructing linear-sized sparse approximations to

implicitly represented weighted cliques whose edge weights are products of weights at vertices (Section 13.14), and extend this to the analog for bDD matrices (Section 13.11.4).

13.5 Background

Proof. (of Fact 13.2.2) Note that

$$\|\boldsymbol{x} - \boldsymbol{Z}b\|_{\boldsymbol{M}}^2 = \boldsymbol{x}^*\boldsymbol{M}x - 2\boldsymbol{x}^*\boldsymbol{M}\boldsymbol{Z}\boldsymbol{M}\boldsymbol{x} + \boldsymbol{x}^*\boldsymbol{M}\boldsymbol{Z}\boldsymbol{M}\boldsymbol{Z}\boldsymbol{M}\boldsymbol{x}$$

Since all nonzero eigenvalues of $\boldsymbol{M}$ are at least μ, the eigenvalues of $\boldsymbol{M}^{1/2}(\boldsymbol{M}+\varepsilon\mu\boldsymbol{I})^{-1}\boldsymbol{M}^{1/2}$ lie between $1/(1+\varepsilon)$ and 1. Using $\boldsymbol{Z} \approx_\varepsilon (\boldsymbol{M}+\varepsilon\mu\boldsymbol{I})^{-1}$, we see that the eigenvalues of $\boldsymbol{M}^{1/2}\boldsymbol{Z}\boldsymbol{M}^{1/2}$ lie between $e^{-2\varepsilon}$ and e^{ε}. Using $0 < \varepsilon < 1/2$, we have

$$\|\boldsymbol{x} - \boldsymbol{Z}\boldsymbol{b}\|_{\boldsymbol{M}}^2 \leq (1 - 2e^{-2\varepsilon} + e^{2\varepsilon})\boldsymbol{x}^*\boldsymbol{M}\boldsymbol{x} \leq 6\varepsilon\|\boldsymbol{x}\|_{\boldsymbol{M}}^2.$$

□

Fact 13.5.1. *Let $\boldsymbol{A}$ be a matrix of condition number κ and let $\boldsymbol{A} \approx_\varepsilon \boldsymbol{B}$ for $\varepsilon \leq (56\kappa^3)^{-1}$. Then, $\boldsymbol{A}^3 \approx_{28\kappa^3\varepsilon} \boldsymbol{B}^3$.*

Proof. First, observe that $\boldsymbol{A} \approx_\varepsilon \boldsymbol{B}$ implies that $\|\boldsymbol{B}\| \leq (1+e^{\varepsilon})\|\boldsymbol{A}\| \leq 2\|\boldsymbol{A}\|$. It also implies that $\|\boldsymbol{A}-\boldsymbol{B}\| \leq 2\varepsilon\|\boldsymbol{A}\|$. As

$$\boldsymbol{A}^2 - \boldsymbol{B}^2 = \frac{1}{2}(\boldsymbol{A}-\boldsymbol{B})(\boldsymbol{A}+\boldsymbol{B}) + \frac{1}{2}(\boldsymbol{A}+\boldsymbol{B})(\boldsymbol{A}-\boldsymbol{B}),$$

$\|\boldsymbol{A}^2 - \boldsymbol{B}^2\| \leq 6\varepsilon\|\boldsymbol{A}\|^2$. Similarly, as

$$\boldsymbol{A}^3 - \boldsymbol{B}^3 = \frac{1}{2}(\boldsymbol{A}-\boldsymbol{B})(\boldsymbol{A}^2+\boldsymbol{B}^2) + \frac{1}{2}(\boldsymbol{A}+\boldsymbol{B})(\boldsymbol{A}^2-\boldsymbol{B}^2),$$

$\|\boldsymbol{A}^3 - \boldsymbol{B}^3\| \leq 28\varepsilon\|\boldsymbol{A}\|^3$.

Let $\kappa = \|\boldsymbol{A}\|/\lambda_{min}(\boldsymbol{A})$. The above relation implies that

$$\|\boldsymbol{A}^3 - \boldsymbol{B}^3\| \leq 28\varepsilon\kappa^3\lambda_{min}(\boldsymbol{A})^3 = 28\varepsilon\kappa^3\lambda_{min}(\boldsymbol{A}^3).$$

Thus, as $28\varepsilon\kappa^3 \leq 1/2$,

$$\boldsymbol{A}^3 \approx_{56\varepsilon\kappa^3} \boldsymbol{B}^3.$$

□

Proof. (of Fact 13.2.3)

By Fact 13.5.1, using $\delta = \frac{\varepsilon}{56\kappa^3}$,

$$\boldsymbol{M}\boldsymbol{Z}^3\boldsymbol{M} \approx_\varepsilon \boldsymbol{M}(\boldsymbol{M}+\varepsilon\mu I)^{-3}\boldsymbol{M}.$$

$\boldsymbol{M}$ has an eigendecomposition in the same basis as $(\boldsymbol{M}+\varepsilon\mu I)^{-3}$, and so it follows that $\boldsymbol{M}(\boldsymbol{M}+\varepsilon\mu I)^{-3}\boldsymbol{M}$ has the same eigenbasis, and the same null space as $\boldsymbol{M}$.

When λ^{-1} is the eigenvalue of $\boldsymbol{M}^+$ of an eigenvector $\boldsymbol{v}$, the corresponding eigenvalue of $\boldsymbol{M}(\boldsymbol{M}+\varepsilon\mu I)^{-3}\boldsymbol{M}$ is $\beta \stackrel{\text{def}}{=} \frac{\lambda^2}{(\lambda+\varepsilon\mu)^3}$ and

$$\lambda^{-1} > \beta \geq \frac{\lambda^2}{(1+\varepsilon)^3\lambda^3} = e^{-3\varepsilon}\lambda^{-1}.$$

So $\boldsymbol{M}(\boldsymbol{M}+\varepsilon\mu I)^{-3}\boldsymbol{M} \approx_{3\varepsilon} \boldsymbol{M}^+$, and $\boldsymbol{M}\boldsymbol{Z}^3\boldsymbol{M} \approx_{4\varepsilon} \boldsymbol{M}^+$. □

Fact 13.5.2. *For every $\boldsymbol{d} \in \mathbb{C}^{r\times r}$, we can find $\boldsymbol{Q}^{(1)}, \boldsymbol{Q}^{(2)} \in \mathbb{C}^{r\times r}$ where*

$$(\boldsymbol{Q}^{(1)})(\boldsymbol{Q}^{(1)})^* = (\boldsymbol{Q}^{(1)})^*(\boldsymbol{Q}^{(1)}) = (\boldsymbol{Q}^{(2)})(\boldsymbol{Q}^{(2)})^* = (\boldsymbol{Q}^{(2)})^*(\boldsymbol{Q}^{(2)}) = \boldsymbol{I}_r,$$

such that

$$\boldsymbol{d} = \frac{1}{2}\|\boldsymbol{d}\|(\boldsymbol{Q}^{(1)} + \boldsymbol{Q}^{(2)}).$$

Proof. Let $\hat{\boldsymbol{d}} = \frac{1}{\|\boldsymbol{d}\|}\boldsymbol{d}$. Thus, $\|\hat{\boldsymbol{d}}\| = 1$. Write $\hat{\boldsymbol{d}}$ using its singular value decomposition as $\boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^*$, where $\boldsymbol{D}, \boldsymbol{U}, \boldsymbol{V} \in \mathbb{C}^{r\times r}$, $\boldsymbol{D}$ is a real diagonal matrix with the singular values of $\boldsymbol{d}$ on the diagonal, and $\boldsymbol{U}\boldsymbol{U}^* = \boldsymbol{U}^*\boldsymbol{U} = \boldsymbol{V}\boldsymbol{V}^* = \boldsymbol{V}^*\boldsymbol{V} = \boldsymbol{I}_r$. Since $\|\hat{\boldsymbol{d}}\| = 1$, we have $\boldsymbol{D}_{j,j} \in [0,1]$ for all $j \in [r]$. Thus, there exists a real θ_j such that $\cos\theta_j = \boldsymbol{D}_{j,j}$. If we let $\boldsymbol{D}^{(1)}, \boldsymbol{D}^{(2)}$ be diagonal matrices usch that $\boldsymbol{D}^{(1)}_{j,j} = \exp(i\theta_j), \boldsymbol{D}^{(2)}_{j,j} = \exp(-i\theta_j)$, we have $\boldsymbol{D} = \frac{1}{2}(\boldsymbol{D}^{(1)} + \boldsymbol{D}^{(2)})$. Moreover,

$$(\boldsymbol{D}^{(1)})(\boldsymbol{D}^{(1)})^* = (\boldsymbol{D}^{(1)})^*(\boldsymbol{D}^{(1)}) = (\boldsymbol{D}^{(2)})(\boldsymbol{D}^{(2)})^* = (\boldsymbol{D}^{(2)})^*(\boldsymbol{D}^{(2)}) = \boldsymbol{I}_r.$$

Letting $\boldsymbol{Q}^{(k)} = \boldsymbol{U}\boldsymbol{D}^{(k)}\boldsymbol{V}^*$ for $k = 1, 2$, we get $\hat{\boldsymbol{d}} = \frac{1}{2}(\boldsymbol{Q}^{(1)} + \boldsymbol{Q}^{(2)})$ and hence $\boldsymbol{d} = \frac{1}{2}\|\boldsymbol{d}\|(\boldsymbol{Q}^{(1)} + \boldsymbol{Q}^{(2)})$. Moreover,

$$(\boldsymbol{Q}^{(1)})(\boldsymbol{Q}^{(1)})^* = (\boldsymbol{Q}^{(1)})^*(\boldsymbol{Q}^{(1)}) = (\boldsymbol{Q}^{(2)})(\boldsymbol{Q}^{(2)})^* = (\boldsymbol{Q}^{(2)})^*(\boldsymbol{Q}^{(2)}) = \boldsymbol{I}_r.$$

□

Fact 13.5.3 (Lemma B.1. from [194])**.** *If $\boldsymbol{M}$ and $\widetilde{\boldsymbol{M}}$ are positive semidefinite matrices satisfying $\boldsymbol{M} \preceq \widetilde{\boldsymbol{M}}$, then*

$$Sc(\boldsymbol{M}, F) \preceq Sc\left(\widetilde{\boldsymbol{M}}, F\right).$$

This fact can be proven via an energy minimization definition of Schur complement. More details on this formulation can be found in [194].

■ 13.6 Block Diagonally Dominant Matrices

In this section, we prove a few basic facts about bDD matrices. The following lemma gives an equivalent definition of bDD matrices.

Lemma 13.6.1. *A Hermitian block-matrix $\boldsymbol{M} \in (\mathbb{C}^{r\times r})^{n\times n}$ is bDD iff it can be written as $\boldsymbol{D} - \boldsymbol{A}$ where $\boldsymbol{D}$ is block-diagonal, $\boldsymbol{A}$ is Hermitian, and $\boldsymbol{D}_{[i,i]} \succeq \boldsymbol{I}_r \sum_j \|\boldsymbol{A}_{[i,j]}\|$, for all $i \in V$.*

Proof. The only if direction is easy. For bDD matrix $\boldsymbol{M}$, if we let $\boldsymbol{D}$ be the block diagonal matrix such that $\boldsymbol{D}_{[i,i]} = \boldsymbol{M}_{[i,i]}$, and $\boldsymbol{A}$ be the matrix such that

$$\boldsymbol{A}_{[i,j]} = \begin{cases} 0 & i = j \\ \boldsymbol{M}_{[i,j]} & i \neq j \end{cases}$$

It is immediate that $\boldsymbol{A}$ is Hermitian. Moreover, $\boldsymbol{M}$ is bDD implies that $\boldsymbol{D}_{[i,i]} \succeq \boldsymbol{I}_r \sum_j \|\boldsymbol{A}_{[i,j]}\|$.

For the if direction. Suppose we have $\boldsymbol{D}, \boldsymbol{A}$ such that $\boldsymbol{D}$ is block-diagonal, and for all i $\boldsymbol{D}_{[i,i]} \succeq \boldsymbol{I}_r \sum_j \|\boldsymbol{A}_{[i,j]}\|$. Thus, letting $\boldsymbol{M} = \boldsymbol{D} - \boldsymbol{A}$, we have, for all $i \in V$,

$$\boldsymbol{I}_r \cdot \sum_{j\neq i} \|\boldsymbol{M}_{[i,j]}\| = \boldsymbol{I}_r \cdot \sum_{j\neq i} \|\boldsymbol{A}_{[i,j]}\| \preccurlyeq \boldsymbol{D}_{[i,i]} - \boldsymbol{I}_r \cdot \|\boldsymbol{A}_{[i,i]}\| \preccurlyeq \boldsymbol{D}_{[i,i]} - \boldsymbol{A}_{[i,i]} = \boldsymbol{M}_{[i,i]},$$

where we used that since $\boldsymbol{A}, \boldsymbol{M}$ are Hermitian, $\boldsymbol{D}$ is also Hermitian, and thus $\boldsymbol{D}_{[i,i]}, \boldsymbol{A}_{[i,i]}$ are Hermitian.

□

This immediately implies the corollary that flipping the sign of off-diagonal blocks preserves bDD-ness.

Corollary 13.6.2. *Given a bDD matrix $\boldsymbol{M}$, write it as $\boldsymbol{D} - \boldsymbol{A}$, where $\boldsymbol{D}$ is a block-diagonal, and $\boldsymbol{A}$ has its diagonal blocks as zero. Then, $\boldsymbol{D} + \boldsymbol{A}$ is also PSD.*

Proof. First observe that for all i, $(\boldsymbol{D} + \boldsymbol{A})_{[i,i]} = (\boldsymbol{D} - \boldsymbol{A})_{[i,i]} = \boldsymbol{M}_{[i,i]}$, i.e., their diagonal blocks are identical. Moreover, for all $i \neq j$, we have $\|(\boldsymbol{D} + \boldsymbol{A})_{[i,j]}\| = \|(\boldsymbol{D} - \boldsymbol{A})_{[i,j]}\| = \|\boldsymbol{M}_{[i,j]}\|$. Thus $\boldsymbol{D} + \boldsymbol{A}$ is also bDD, and hence PSD. □

Next, we show that the class of bDD matrices is closed under Schur complement.

Lemma 13.6.3. *The class of bDD matrices is closed under Schur complement.*

Proof. Since Schur complementation does not depend on the order of indices eliminated, it suffices to prove that for any bDD matrix $\boldsymbol{M} \in (\mathbb{C}^{r\times r})^{n\times n}$, $Sc(\boldsymbol{M}, 1)$ is a bDD matrix. Let $C = V \setminus \{1\}$.

We have $Sc(\boldsymbol{M}, 1) = \boldsymbol{M}_{[C,C]} - \boldsymbol{M}_{[C,1]}\boldsymbol{M}_{[1,1]}^{-1}\boldsymbol{M}_{[1,C]}$. Let $\boldsymbol{D} \in (\mathbb{C}^{r\times r})^{C\times C}$ be the block diagonal matrix such that $\boldsymbol{D}_{[i,i]} = \boldsymbol{M}_{[i,i]}$ for $i \in C$. Expressing $Sc(\boldsymbol{M}, 1)$ as $\boldsymbol{D} + (\boldsymbol{M}_{[C,C]} - \boldsymbol{D} + \boldsymbol{M}_{[C,1]}\boldsymbol{M}_{[1,1]}^{-1}\boldsymbol{M}_{[1,C]})$, we have for any $i \in C$,

$$\begin{aligned}
\sum_{j\in C} \|(\boldsymbol{M}_{[C,C]} - \boldsymbol{D} + \boldsymbol{M}_{[C,1]}\boldsymbol{M}_{[1,1]}^{-1}\boldsymbol{M}_{[1,C]})_{[i,j]}\| &\leq \left(\sum_{j\in C: j\neq i} \|\boldsymbol{M}_{[i,j]}\|\right) + \sum_{j\in C} \|\boldsymbol{M}_{[i,1]}\boldsymbol{M}_{[1,1]}^{-1}\boldsymbol{M}_{[1,j]}\| \\
&\leq \left(\sum_{j\in C: j\neq i} \|\boldsymbol{M}_{[i,j]}\|\right) + \|\boldsymbol{M}_{[i,1]}\|\|\boldsymbol{M}_{[1,1]}^{-1}\| \sum_{j\in C} \|\boldsymbol{M}_{[1,j]}\| \\
&\leq \left(\sum_{j\in C: j\neq i} \|\boldsymbol{M}_{[i,j]}\|\right) + \|\boldsymbol{M}_{[i,1]}\| = \sum_{j\in V: j\neq i} \|\boldsymbol{M}_{[i,j]}\|,
\end{aligned}$$

where the last inequality uses $\|\boldsymbol{M}_{[1,1]}^{-1}\| \left(\sum_{j\neq 1} \|\boldsymbol{M}_{[1,j]}\|\right) \leq 1$, since $\boldsymbol{M}_{[1,1]} \succcurlyeq \boldsymbol{I}_r \cdot \sum_{j\neq 1} \|\boldsymbol{M}_{[1,j]}\|$. Thus using Lemma 13.6.1, we have $Sc(\boldsymbol{M}, 1) = \boldsymbol{D} - (-(\boldsymbol{M}_{[C,C]} - \boldsymbol{D} + \boldsymbol{M}_{[C,1]}\boldsymbol{M}_{[1,1]}^{-1}\boldsymbol{M}_{[1,C]}))$ is bDD. □

The next definition describes a special form that we can express any bDD matrix in, which will occasionally be useful.

Definition 13.6.4. A matrix $\boldsymbol{B} \in (\mathbb{C}^{r\times r})^{n\times m}$ is called a *unitary edge-vertex transfer matrix*, when each block column of $\boldsymbol{B}_{[e]}$ has exactly two nonzero blocks $\boldsymbol{U}_e, \boldsymbol{V}_e \in \mathbb{C}^{r\times r}$ s.t. $\boldsymbol{U}_e\boldsymbol{U}_e^* = \boldsymbol{V}_e\boldsymbol{V}_e^* = w_e\boldsymbol{I}_r$ where $w_e \geq 0$.

Lemma 13.6.5. *Every bDD matrix $\boldsymbol{M} \in (\mathbb{C}^{r\times r})^{n\times n}$ with m nonzero off-diagonal blocks can be written as $\boldsymbol{X} + \boldsymbol{B}\boldsymbol{B}^*$ where $\boldsymbol{B} \in (\mathbb{C}^{r\times r})^{n\times 2m}$ is a unitary edge-vertex transfer matrix and $\boldsymbol{X}$ is a block diagonal PSD matrix. This implies that every bDD matrix is PSD. Furthermore, for every block diagonal matrix $\boldsymbol{Y}$ s.t. $\boldsymbol{M} - \boldsymbol{Y}$ is bDD, we have $\boldsymbol{X} \succcurlyeq \boldsymbol{Y}$. This decomposition can be found in $O(m)$ time and $O(\log n)$ depth.*

Proof. Consider a pair $\{i, j\} \in V \times V$ such that $i \neq j$, and $\boldsymbol{M}_{[i,j]} \neq 0$. Using Fact 13.5.2, we can write such a $\boldsymbol{M}_{[i,j]}$ as $\frac{1}{2}\|\boldsymbol{M}_{[i,j]}\|(\boldsymbol{Q}_{\{i,j\}}^{(1)} + \boldsymbol{Q}_{\{i,j\}}^{(2)})$, where $\boldsymbol{Q}_{\{i,j\}}^{(1)}(\boldsymbol{Q}_{\{i,j\}}^{(1)})^* = \boldsymbol{Q}_{\{i,j\}}^{(2)}(\boldsymbol{Q}_{\{i,j\}}^{(2)})^* = \boldsymbol{I}_r$. we construct two vectors vector $\boldsymbol{B}_{\{i,j\}}^{(1)}, \boldsymbol{B}_{\{i,j\}}^{(2)} \in (\mathbb{C}^{r\times r})^n$ such that for $k = 1, 2$, $\left(\boldsymbol{B}_{\{i,j\}}^{(k)}\right)_{[i]} = \frac{1}{\sqrt{2}}\|\boldsymbol{M}_{[i,j]}\|^{1/2}\boldsymbol{I}_r$,

$\left(\boldsymbol{B}^{(k)}_{\{i,j\}}\right)_{[j]} = \frac{1}{\sqrt{2}}\|\boldsymbol{M}_{[i,j]}\|^{1/2}\left(\boldsymbol{Q}^{(k)}_{\{i,j\}}\right)^*$, and all other blocks are zero. We can verify that for all $k, \ell \in V$,

$$\left(\boldsymbol{B}^{(1)}_{\{i,j\}}(\boldsymbol{B}^{(1)}_{\{i,j\}})^* + \boldsymbol{B}^{(2)}_{\{i,j\}}(\boldsymbol{B}^{(2)}_{\{i,j\}})^*\right)_{[k,\ell]} = \begin{cases} \|\boldsymbol{M}_{[i,j]}\|\boldsymbol{I}_r & k = \ell = i, \\ \|\boldsymbol{M}_{[i,j]}\|\boldsymbol{I}_r & k = \ell = j, \\ \boldsymbol{M}_{[i,j]} & k = i, \ell = j, \\ \boldsymbol{M}^*_{[i,j]} = \boldsymbol{M}_{[j,i]} & k = j, \ell = i, \\ 0 & \text{otherwise.} \end{cases}$$

We let $\boldsymbol{X} = \boldsymbol{M} - \sum_{\{i,j\}:i\neq j}\left(\boldsymbol{B}^{(1)}_{\{i,j\}}(\boldsymbol{B}^{(1)}_{\{i,j\}})^* + \boldsymbol{B}^{(2)}_{\{i,j\}}(\boldsymbol{B}^{(2)}_{\{i,j\}})^*\right)$, which must be block-diagonal. We now show that for all $i \in V$, the block $\boldsymbol{X}_{[i,i]}$ is PSD. We have for all $i \in V$,

$$\boldsymbol{X}_{[i,i]} = \boldsymbol{M}_{[i,i]} - \boldsymbol{I}_r \cdot \sum_{j:j\neq i}\|\boldsymbol{M}_{[i,j]}\| \succcurlyeq 0,$$

where the last inequality holds since $\boldsymbol{M}$ is bDD.

Thus, if we define $\boldsymbol{B} \in (\mathbb{C}^{r\times r})^{n\times 2m}$ such that its columns are all the vectors $\boldsymbol{B}^{(1)}_{\{i,j\}}, \boldsymbol{B}^{(2)}_{\{i,j\}}$ defined above, we have $\boldsymbol{M} = \boldsymbol{X} + \boldsymbol{B}\boldsymbol{B}^*$, and every column of $\boldsymbol{B}$ has exactly 2 nonzero blocks.

To show that for every block diagonal $\boldsymbol{Y}$ s.t. $\boldsymbol{M} - \boldsymbol{Y}$ is bDD, $\boldsymbol{X} \succcurlyeq \boldsymbol{Y}$, first consider applying the decomposition described above to $\boldsymbol{M} - \boldsymbol{Y}$ instead of $\boldsymbol{M}$. Since the construction of $\boldsymbol{B}$ only depends on the off-diagonal blocks, we get $\boldsymbol{M} - \boldsymbol{Y} = \boldsymbol{\Lambda} + \boldsymbol{B}\boldsymbol{B}^*$, where $\boldsymbol{\Lambda}$ is block diagonal and PSD. So, $\boldsymbol{X} - \boldsymbol{Y} = \boldsymbol{M} - \boldsymbol{B}\boldsymbol{B}^* - \boldsymbol{Y} = \boldsymbol{\Lambda} \succcurlyeq 0$.

It is immediate that the decomposition can be found in $O(m)$ time and $O(\log n)$ depth. □

13.7 Schur Complement Chains

In this section, we give a proof of Lemma 13.3.1. We restate the lemma here for convenience.

Lemma 13.3.1. Consider an ε-SCC for $\boldsymbol{M}^{(0)}$ where $\boldsymbol{M}^{(i)}$ and $\boldsymbol{Z}^{(i)}$ can be applied to a vector in work $W_{\boldsymbol{M}^{(i)}}, W_{\boldsymbol{D}^{(i)}}$ and depth $D_{\boldsymbol{M}^{(i)}}, D_{\boldsymbol{Z}^{(i)}}$ respectively.

The algorithm ApplyChain $\left((\boldsymbol{M}^{(1)}, \boldsymbol{Z}^{(1)}), \ldots, (\boldsymbol{M}^{(d)}, \boldsymbol{Z}^{(d)}); F_1, \ldots, F_d; \boldsymbol{b}\right)$ corresponds to a linear operator $\boldsymbol{W}$ acting on $\boldsymbol{b}$ such that

1. $\boldsymbol{W}^{-1} \approx_{\sum_{i=1}^d 2\varepsilon_i} \boldsymbol{M}^{(0)}$, and
2. for any vector $\boldsymbol{b}$, ApplyChain $\left((\boldsymbol{M}^{(1)}, \boldsymbol{Z}^{(1)}), \ldots, (\boldsymbol{M}^{(d)}, \boldsymbol{Z}^{(d)}); F_1, \ldots, F_d; \boldsymbol{b}\right)$ runs in $O\left(\sum_{i=1}^d (D_{\boldsymbol{M}^{(i)}} + D_{\boldsymbol{Z}^{(i)}})\right)$ depth and $O\left(\sum_{i=1}^d (W_{\boldsymbol{M}^{(i)}} + W_{\boldsymbol{Z}^{(i)}})\right)$ work.

The pseudocode for procedure ApplyChain that uses an ε-vertex sparsifier chain to approximately solve a system of equations in $\boldsymbol{M}^{(0)}$ is given in Figure 13-1.

Proof. We begin by observing that the output vector $\boldsymbol{x}^{(0)}$ is a linear transformation of the input vector $\boldsymbol{b}^{(0)}$. Let $\boldsymbol{W}^{(0)}$ be the matrix that realizes this transformation. Similarly, for $1 \le i \le d$, define $\boldsymbol{W}^{(i)}$ to be the matrix so that

$$\boldsymbol{x}^{(i)} = \boldsymbol{W}^{(i)}\boldsymbol{b}^{(i)}.$$

An examination of the algorithm reveals that

$$\boldsymbol{W}^{(d)} = \left(\boldsymbol{M}^{(d)}\right)^{-1}, \tag{13.4}$$

Algorithm 35: $\boldsymbol{x}^{(0)} = \text{ApplyChain}\left((\boldsymbol{M}^{(1)}, \boldsymbol{Z}^{(1)}), \ldots, (\boldsymbol{M}^{(d)}, \boldsymbol{Z}^{(d)}); F_1, \ldots, F_d\right)$

Initialize $\boldsymbol{b}^{(0)} \leftarrow \boldsymbol{b}$.
for $i = 1, \ldots, d$ **do**
  $\boldsymbol{x}^{(i-1)}_{[F_i]} \leftarrow \boldsymbol{Z}^{(i)} \boldsymbol{b}^{(i-1)}_{[F_i]}$,
  $\boldsymbol{b}^{(i)} \leftarrow \boldsymbol{b}^{(i-1)}_{[C_i]} - \boldsymbol{M}^{(i-1)}_{[C_i,F_i]} \boldsymbol{x}^{(i-1)}_{[F_i]}$.
end
$\boldsymbol{x}^{(d)} \leftarrow \left(\boldsymbol{M}^{(d)}\right)^{-1} \boldsymbol{b}^{(d)}$.
for $i = d, \ldots, 1$ **do**
  $\boldsymbol{x}^{(i-1)}_{[C_i]} \leftarrow \boldsymbol{x}^{(i)}$.
  $\boldsymbol{x}^{(i-1)}_{[F_i]} \leftarrow \boldsymbol{x}^{(i-1)}_{[F_i]} - \boldsymbol{Z}^{(i)} \boldsymbol{M}^{(i-1)}_{[F_i,C_i]} \boldsymbol{x}^{(i)}$.
end

Figure 13-1: Solver Algorithm using Vertex Sparsifier Chain

and for $1 \leq i \leq d$,

$$\boldsymbol{W}^{(i-1)} = \begin{bmatrix} \boldsymbol{I} & -\boldsymbol{Z}^{(i)} \boldsymbol{M}^{(i-1)}_{[F_i,C_i]} \\ 0 & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{Z}^{(i)} & 0 \\ 0 & \boldsymbol{W}^{(i)} \end{bmatrix} \begin{bmatrix} \boldsymbol{I} & 0 \\ -\boldsymbol{M}^{(i-1)}_{[C_i,F_i]} \boldsymbol{Z}^{(i)} & \boldsymbol{I} \end{bmatrix}. \tag{13.5}$$

We will now prove by backwards induction on i that

$$\left(\boldsymbol{W}^{(i)}\right)^{-1} \approx_{\sum_{j=i+1}^{d} 2\varepsilon_j} \boldsymbol{M}^{(i)}.$$

The base case of $i = d$ follows from (13.4). Using the definition of an $\boldsymbol{\varepsilon}$-SCC, we know that $0 \preceq (\boldsymbol{Z}^{(i)})^{-1} - \boldsymbol{M}^{(i-1)}_{[F_i,F_i]} \preceq \varepsilon_i \cdot Sc\left(\boldsymbol{M}^{(i-1)}, C_i\right)$. We show in Lemma 13.7.1 that this implies

$$\begin{bmatrix} \boldsymbol{I} & -\boldsymbol{Z}^{(i)} \boldsymbol{M}^{(i-1)}_{[F_i,C_i]} \\ 0 & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{Z}^{(i)} & 0 \\ 0 & Sc\left(\boldsymbol{M}^{(i-1)}, F_i\right)^{-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{I} & 0 \\ -\boldsymbol{M}^{(i-1)}_{[C_i,F_i]} \boldsymbol{Z}^{(i)} & \boldsymbol{I} \end{bmatrix} \approx_{\varepsilon_i} \left(\boldsymbol{M}^{(i-1)}\right)^{-1}.$$

As $\boldsymbol{M}^{(i)} \approx_{\varepsilon_i} Sc\left(\boldsymbol{M}^{(i-1)}, F_i\right)$,

$$\begin{bmatrix} \boldsymbol{I} & -\boldsymbol{Z}^{(i)} \boldsymbol{M}^{(i-1)}_{[F_i,C_i]} \\ 0 & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{Z}^{(i)} & 0 \\ 0 & \left(\boldsymbol{M}^{(i)}\right)^{-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{I} & 0 \\ -\boldsymbol{M}^{(i-1)}_{[C_i,F_i]} \boldsymbol{Z}^{(i)} & \boldsymbol{I} \end{bmatrix} \approx_{2\varepsilon_i} \left(\boldsymbol{M}^{(i-1)}\right)^{-1}.$$

By combining this identity with (13.5) and our inductive hypothesis, we obtain

$$\boldsymbol{W}^{(i-1)} \approx_{\sum_{j=i}^{d} 2\varepsilon_j} \left(\boldsymbol{M}^{(i-1)}\right)^{-1}.$$

Thus, by induction, we obtain

$$\boldsymbol{W}^{(0)} \approx_{\sum_{j=1}^{d} 2\varepsilon_j} \left(\boldsymbol{M}^{(0)}\right)^{-1}.$$

The whole algorithm involves a constant number of applications of $\boldsymbol{Z}^{(i)}, \boldsymbol{M}^{(i-1)}_{[F_i,C_i]}$, and $\boldsymbol{M}^{(i-1)}_{[C_i,F_i]}$. We observe that in order to compute $\boldsymbol{M}^{(i-1)}_{[F_i,C_i]} x_{C_i}$, using a multiplication procedure for $\boldsymbol{M}^{(i-1)}$, we can

pad x_{C_i} with zeros, multiply by $\boldsymbol{M}^{(i-1)}$, and read off the answer on the indices in C_i. Similarly, we can multiply vectors with $\boldsymbol{M}^{(i-1)}_{[C_i,F_i]}$. This immediately gives the claimed bounds on work and depth. □

We now prove the deferred claims from the above proof.

Lemma 13.7.1. *Let $\boldsymbol{M} \in (\mathbb{C}^{r\times r})^{n\times n}$ be a bDD matrix, $F \subseteq V$ be a subset of the indices, and $\boldsymbol{Z} \in (\mathbb{C}^{r\times r})^{|F|\times|F|}$ be an hermitian operator satisfying $0 \preceq \boldsymbol{Z}^{-1} - \boldsymbol{M}_{[F,F]} \preceq \varepsilon \cdot Sc(\boldsymbol{M}, C)$. Then,*

$$\begin{bmatrix} \boldsymbol{I} & -\boldsymbol{Z}\boldsymbol{M}_{[F,C]} \\ 0 & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{Z} & 0 \\ 0 & Sc(\boldsymbol{M}, F)^{-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{I} & 0 \\ -\boldsymbol{M}_{[C,F]}\boldsymbol{Z} & \boldsymbol{I} \end{bmatrix} \approx_\varepsilon \boldsymbol{M}^{-1}.$$

Proof. Define

$$\widehat{\boldsymbol{M}} = \begin{bmatrix} (\boldsymbol{Z})^{-1} & \boldsymbol{M}_{[F,C]} \\ \boldsymbol{M}_{[C,F]} & \boldsymbol{M}_{[C,C]} \end{bmatrix}.$$

Using Lemma 13.7.2, we know that the assumption on $\boldsymbol{Z}$ is equivalent to

$$\boldsymbol{M} \preccurlyeq \widehat{\boldsymbol{M}} \preccurlyeq (1+\varepsilon)\,\boldsymbol{M}.$$

By Eq. (13.1), this implies

$$\boldsymbol{M}^{-1} \succcurlyeq \begin{bmatrix} \boldsymbol{I} & -\boldsymbol{Z}\boldsymbol{M}_{[F,C]} \\ 0 & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{Z} & 0 \\ 0 & Sc\left(\widehat{\boldsymbol{M}}, F\right)^{-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{I} & 0 \\ -\boldsymbol{M}_{[C,F]}\boldsymbol{Z} & \boldsymbol{I} \end{bmatrix} \succcurlyeq (1+\varepsilon)^{-1}\boldsymbol{M}^{-1}.$$

From Facts 13.5.3, we know that

$$Sc(\boldsymbol{M}, F)^{-1} \succcurlyeq Sc\left(\widehat{\boldsymbol{M}}, F\right)^{-1} \succcurlyeq (1+\varepsilon)^{-1} Sc(\boldsymbol{M}, F)^{-1}.$$

Now, we substitute this inequality into the one above and obtain

$$(1+\varepsilon)\boldsymbol{M}^{-1} \succcurlyeq \begin{bmatrix} \boldsymbol{I} & -\boldsymbol{Z}\boldsymbol{M}_{[F,C]} \\ 0 & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{Z} & 0 \\ 0 & Sc(\boldsymbol{M}, F)^{-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{I} & 0 \\ -\boldsymbol{M}_{[C,F]}\boldsymbol{Z} & \boldsymbol{I} \end{bmatrix} \succcurlyeq (1+\varepsilon)^{-1}\boldsymbol{M}^{-1},$$

which implies the lemma. □

Lemma 13.7.2. *Given a bDD matrix $\boldsymbol{M}$, a partition of its indices (F, C) such that $\boldsymbol{M}_{[F,F]}, \boldsymbol{M}_{[C,C]}$ are invertible, and an invertible hermitian operator $\boldsymbol{Z} \in (\mathbb{C}^{r\times r})^{|F|\times|F|}$, the following two conditions are equivalent:*

1. $0 \preceq (\boldsymbol{Z})^{-1} - \boldsymbol{M}_{[F,F]} \preceq \varepsilon \cdot Sc(\boldsymbol{M}, C)$.
2. $$\boldsymbol{M} \preccurlyeq \begin{bmatrix} (\boldsymbol{Z})^{-1} & \boldsymbol{M}_{[F,C]} \\ \boldsymbol{M}_{[C,F]} & \boldsymbol{M}_{[C,C]} \end{bmatrix} \preccurlyeq (1+\varepsilon)\boldsymbol{M}.$$

Proof. Writing

$$M = \begin{bmatrix} \boldsymbol{M}_{[F,F]} & \boldsymbol{M}_{[F,C]} \\ \boldsymbol{M}_{[C,F]} & \boldsymbol{M}_{[C,C]} \end{bmatrix},$$

condition 2 is equivalent to

$$0 \preccurlyeq \begin{bmatrix} (\boldsymbol{Z})^{-1} - \boldsymbol{M}_{[F,F]} & 0 \\ 0 & 0 \end{bmatrix} \preccurlyeq \varepsilon\boldsymbol{M}.$$

The left inequality in this statement is equivalent to the left inequality in condition 1. Thus, it suffices to prove the right sides are equivalent.

To this end, using $\boldsymbol{M}_{[F,C]} = \boldsymbol{M}^*_{[C,F]}$, it suffices to prove that $\forall x \in (\mathbb{C}^r)^{|F|}, y \in (\mathbb{C}^r)^{|C|}$,

$$x^*((\boldsymbol{Z})^{-1} - \boldsymbol{M}_{[F,F]})x \leq \varepsilon(x^*\boldsymbol{M}_{[F,F]}x + 2x^*\boldsymbol{M}_{[F,C]}y + y^*\boldsymbol{M}_{[C,C]}y).$$

This is equivalent to proving $\forall x \in (\mathbb{C}^r)^{|F|}$,

$$x^*((\boldsymbol{Z})^{-1} - \boldsymbol{M}_{[F,F]})x \leq \varepsilon \inf_{y \in (\mathbb{C}^r)^{|C|}} (x^*\boldsymbol{M}_{[F,F]}x + 2x^*\boldsymbol{M}_{[F,C]}y + y^*\boldsymbol{M}_{[C,C]}y).$$

Since $\boldsymbol{M}_{[C,C]} \succcurlyeq 0$, the rhs is a convex function of y. The minimum is achieved at $y = -\boldsymbol{M}^{-1}_{[C,C]}\boldsymbol{M}^*_{[F,C]}x$, and we obtain the equivalent condition

$$x^*((\boldsymbol{Z})^{-1} - \boldsymbol{M}_{[F,F]})x \leq \varepsilon x^*(\boldsymbol{M}_{[F,F]} - \boldsymbol{M}_{[F,C]}\boldsymbol{M}^{-1}_{[C,C]}\boldsymbol{M}^*_{[F,C]})x.$$

Since $Sc(\boldsymbol{M}, C) = \boldsymbol{M}_{[F,F]} - \boldsymbol{M}_{[F,C]}\boldsymbol{M}^{-1}_{[C,C]}\boldsymbol{M}_{[C,F]} = \boldsymbol{M}_{[F,F]} - \boldsymbol{M}_{[F,C]}\boldsymbol{M}^{-1}_{[C,C]}\boldsymbol{M}^*_{[F,C]}$, we obtain our claim. □

13.8 Finding α-bDD Subsets

In this section we check that a simple randomized sampling procedure leads to α-bDD subsets. Specifically we will prove Lemma 13.3.2:

Lemma 13.3.2. Given a bDD matrix $\boldsymbol{M}$ with n block-rows, and an $\alpha \geq 0$, bDDSubset computes a subset F of size least $n/(8(1+\alpha))$ such that $\boldsymbol{M}_{[F,F]}$ is α-bDD. It runs in runs in $O(m)$ expected work and $O(\log n)$ expected depth, where m is the number of nonzero blocks in $\boldsymbol{M}$.

Pseudocode for this routine is given in Figure 13-2.

Algorithm 36: $F =$ bDDSubset$(\boldsymbol{M}, \alpha)$, where $\boldsymbol{M}$ is a bDD matrix with n rows.

Let F' be a uniform random subset of $\{1, \ldots, n\}$ of size $\frac{n}{4(1+\alpha)}$.
Set

$$F = \left\{ i \in F' \text{ such that } \sum_{j:j\neq i} \|\boldsymbol{M}_{[i,j]}\| \geq (1+\alpha) \sum_{j \in F': j \neq i} \|\boldsymbol{M}_{[i,j]}\| \right\}.$$

If $|F| < \frac{n}{8(1+\alpha)}$, goto Step 36.
Output F.

Figure 13-2: Routine for finding an α-strongly block diagonally dominant submatrix

We first show that the set returned is guaranteed to be α-strongly block diagonally dominant.

Lemma 13.8.1. *If* bSDDSubset *terminates, it returns F such that $\boldsymbol{M}_{[F,F]}$ is α-bDD.*

Proof. Consider some $i \in F$, the criteria for including i in F in Step 36 gives:

$$\sum_{j:j\neq i} \|\boldsymbol{M}_{[i,j]}\| \geq (1+\alpha) \sum_{j \in F', j \neq i} \|\boldsymbol{M}_{[i,j]}\| \geq (1+\alpha) \sum_{j \in F, j \neq i} \|\boldsymbol{M}_{[i,j]}\|,$$

where the last inequality follows since F is a subset of F'.

Incorporating this into the definition of $\boldsymbol{M}$ being bDD gives

$$\boldsymbol{M}_{[i,i]} \succcurlyeq \boldsymbol{I}_r \cdot \sum_{j:j\neq i} \|\boldsymbol{M}_{[i,j]}\| \geq (1+\alpha)\boldsymbol{I}_r \cdot \sum_{j \in F, j \neq i} \|\boldsymbol{M}_{[i,j]}\|.$$

which means $\boldsymbol{M}_{[F,F]}$ is α-bDD. □

It remains to show that the algorithm finds a big F quickly. This can be done by upper bounding the expected size of F, or the probability of a single index being in F.

Lemma 13.8.2. *For any i, we have*

$$Pr\left[i \in F' \text{ and } i \notin F\right] \leq \frac{1}{16\,(1+\alpha)}.$$

Proof. This event only happens if $i \in F'$ and

$$\sum_{j \in F', j \neq i} \|\boldsymbol{M}_{[i,j]}\| > \frac{1}{1+\alpha} \sum_{j : j \neq i} \|\boldsymbol{M}_{[i,j]}\|. \tag{13.6}$$

Conditioning on i being selected initially, or $i \in F'$, the probability that each other $j \neq i$ is in F' is

$$\frac{1}{n-1}\left(\frac{n}{4(1+\alpha)} - 1\right) < \frac{1}{4(\alpha+1)},$$

which gives:

$$\mathbb{E}\left[\sum_{j \in F', j \neq i} \|\boldsymbol{M}_{[i,j]}\| \middle| i \in F'\right] < \frac{1}{4(1+\alpha)} \sum_{j : j \neq i} \|\boldsymbol{M}_{[i,j]}\|.$$

Markov's inequality then gives:

$$\Pr\left[\sum_{j \in F', j \neq i} \|\boldsymbol{M}_{[i,j]}\| > \frac{1}{1+\alpha} \sum_{j : j \neq i} \|\boldsymbol{M}_{[i,j]}\| \middle| i \in F'\right] < 1/4,$$

and thus

$$\Pr\left[i \in F' \text{ and } i \notin F\right] = \Pr\left[i \notin F | i \in F'\right] \Pr\left[i \in F'\right] < \frac{1}{4}\frac{1}{4(1+\alpha)} = \frac{1}{16(1+\alpha)}.$$

□

Combining these two bounds gives Lemma 13.3.2.

Proof. (of Lemma 13.3.2) Applying Linearity of Expectation to Lemma 13.8.2 gives

$$\mathbb{E}\left[\left|F' \setminus F\right|\right] \leq \frac{n}{16\,(1+\alpha)}.$$

Markov's inequality then gives

$$\Pr\left[\left|F' \setminus F\right| \geq \frac{n}{8(1+\alpha)}\right] < 1/2.$$

So, with probability at least $1/2$, $|F| \geq n/(8(1+\alpha))$, and the algorithm will pass the test in line 3. Thus, the expected number of iterations made by the algorithm is at most 2. The claimed bounds on the expected work and depth of the algorithm follow. □

13.9 Jacobi Iteration on α-bDD Matrices

From an α-bDD set F, we will construct an operator $\boldsymbol{Z}$ that approximates $\boldsymbol{M}_{[F,F]}^{-1}$ and that can be applied quickly. Specifically, we will show:

Theorem 13.3.1. *Let $\boldsymbol{M}$ be a bDD matrix with index set V, and let $F \subseteq V$ such that $\boldsymbol{M}_{[F,F]}$ is α-bDD for some $\alpha \geq 4$, and has m_{FF} nonzero blocks. The algorithm* Jacobi$(\varepsilon, \boldsymbol{M}_{[F,F]}, \boldsymbol{b})$ *acts as a linear operator $\boldsymbol{Z}$ on $\boldsymbol{b}$ that satisfies*

$$0 \preceq (\boldsymbol{Z})^{-1} - \boldsymbol{M}_{[F,F]} \preceq \varepsilon \cdot Sc(\boldsymbol{M}, V \setminus F).$$

The algorithm takes $O(m_{FF} \log(\frac{1}{\varepsilon}))$ work and $O(\log n \log(\frac{1}{\varepsilon}))$ depth.

Pseudocode of this routine is given in Figure 13-3.

Algorithm 37: $\boldsymbol{x} =$ Jacobi$(\varepsilon, \boldsymbol{M}, \boldsymbol{b})$

Create the matrix $\boldsymbol{L}$ where

$$\boldsymbol{L}_{[i,j]} = \begin{cases} \boldsymbol{I}_r \cdot \sum_{j:j \neq i} \|\boldsymbol{M}_{[i,j]}\| & \text{if } i = j, \\ \boldsymbol{M}_{[i,j]} & \text{otherwise.} \end{cases}$$

Set $\boldsymbol{X} = \boldsymbol{M} - \boldsymbol{L}$.
Set k to be an odd integer that is greater than $\log(3/\varepsilon)$.
Set $\boldsymbol{x}^{(0)} = \boldsymbol{X}^{-1}\boldsymbol{b}$.
for $i = 1 \cdots k$ **do**
  Set $\boldsymbol{x}^{(i)} = -\boldsymbol{X}^{-1}\boldsymbol{L}\boldsymbol{x}^{(i-1)} + \boldsymbol{X}^{-1}\boldsymbol{b}$.
end
Output $\boldsymbol{x}^{(k)}$.

Figure 13-3: Jacobi Iteration for Solving Linear Systems in an α-bDD Matrix

We first verify that any α-bDD matrix has a good block-diagonal preconditioner.

Lemma 13.3.3. Every α-bDD matrix $\boldsymbol{M}$ can be written in the form $\boldsymbol{X} + \boldsymbol{L}$ where $\boldsymbol{X}$ is block-diagonal, $\boldsymbol{L}$ is bDD, and $\boldsymbol{X} \succcurlyeq \frac{\alpha}{2}\boldsymbol{L}$.

Proof. Write $\boldsymbol{L} = \boldsymbol{Y} - \boldsymbol{A}$ where $\boldsymbol{Y}$ is block-diagonal and $\boldsymbol{A}$ has its diagonal blocks as zeros. Note that $\boldsymbol{L}$ is bDD by definition. Thus, by Corollary 13.6.5, $\boldsymbol{L} = \boldsymbol{Y} - \boldsymbol{A} \succcurlyeq 0$.

Using Lemma 13.6.2, we know that $\boldsymbol{Y} + \boldsymbol{A} \succcurlyeq 0$, or $\boldsymbol{Y} \succcurlyeq -\boldsymbol{A}$. This implies $2\boldsymbol{Y} \succcurlyeq \boldsymbol{L}$.

As $\boldsymbol{M}$ is α-strongly diagonally dominant and $\boldsymbol{M}_{[i,i]} = \boldsymbol{X}_{[i,i]} + \boldsymbol{Y}_{[i,i]}$, we have

$$(\boldsymbol{X} + \boldsymbol{Y})_{[i,i]} \succcurlyeq (1+\alpha)\,\boldsymbol{I}_r \cdot \sum_{j \neq i} \|\boldsymbol{M}_{[i,j]}\| = (1+\alpha)\,\boldsymbol{Y}_{[i,i]}.$$

Manipulating this then gives:

$$\boldsymbol{X} \succcurlyeq \alpha\,\boldsymbol{Y} \succcurlyeq \frac{\alpha}{2}\boldsymbol{L}.$$

□

We now move on to measuring the quality of the operator generated by Jacobi. It can be checked

that running it k steps gives the operator

$$\boldsymbol{Z}^{(k)} \stackrel{\text{def}}{=} \sum_{i=0}^{k} \boldsymbol{X}_{[F,F]}^{-1} \left(-\boldsymbol{L}_{[F,F]} \boldsymbol{X}_{[F,F]}^{-1}\right)^{i}, \tag{13.7}$$

which is equivalent to evaluating a truncation of the Neumann series for $\boldsymbol{M}^{-1}$

Lemma 13.9.1. *Let $\boldsymbol{M}$ be a matrix with splitting $\boldsymbol{M} = \boldsymbol{X} + \boldsymbol{L}$ where $0 \preccurlyeq \boldsymbol{L} \preccurlyeq \beta \boldsymbol{X}$ for some parameter $1 > \beta > 0$. Then, for odd k and for $\boldsymbol{Z}^{(k)}$ as defined in (13.7) we have:*

$$\boldsymbol{X} + \boldsymbol{L} \preceq (\boldsymbol{Z}^{(k)})^{-1} \preceq \boldsymbol{X} + (1+\delta)\,\boldsymbol{L} \tag{13.8}$$

where

$$\delta = \beta^{k} \frac{1+\beta}{1-\beta^{k+1}}.$$

Proof. The left-hand inequality is equivalent to the statement that all the eigenvalues of $\boldsymbol{Z}^{(k)}(\boldsymbol{X}+\boldsymbol{L})$ are at most 1 (see [34, Lemma 2.2] or [243, Proposition 3.3]). To see that this is the case, expand

$$\begin{aligned}
\boldsymbol{Z}^{(k)}(\boldsymbol{X}+\boldsymbol{L}) &= \left(\sum_{i=0}^{k} \boldsymbol{X}^{-1}(-\boldsymbol{L}\boldsymbol{X}^{-1})^{i}\right)(\boldsymbol{X}+\boldsymbol{L}) \\
&= \sum_{i=0}^{k} (-\boldsymbol{X}^{-1}\boldsymbol{L})^{i} - \sum_{i=1}^{k+1} (-\boldsymbol{X}^{-1}\boldsymbol{L})^{i} \\
&= \boldsymbol{I} - (\boldsymbol{X}^{-1}\boldsymbol{L})^{k+1}.
\end{aligned}$$

As all the eigenvalues of an even power of a matrix are nonnegative, all of the eigenvalues of this last matrix are at most 1.

Similarly, the other inequality is equivalent to the assertion that all of the eigenvalues of $\boldsymbol{Z}^{(k)}(\boldsymbol{X} + (1+\delta)\boldsymbol{L})$ are at least one. Expanding this product yields

$$\left(\sum_{i=0}^{k} \boldsymbol{X}^{-1}(-\boldsymbol{L}\boldsymbol{X}^{-1})^{i}\right)(\boldsymbol{X} + (1+\delta)\boldsymbol{L}) = \boldsymbol{I} - (\boldsymbol{X}^{-1}\boldsymbol{L})^{k+1} + \delta \sum_{i=0}^{k} (-1)^{i} (\boldsymbol{X}^{-1}\boldsymbol{L})^{i+1}$$

The eigenvalues of this matrix are precisely the numbers

$$1 - \lambda^{k+1} + \delta \sum_{i=0}^{k} (-1)^{i} \lambda^{i+1}, \tag{13.9}$$

where λ ranges over the eigenvalues of $\boldsymbol{X}^{-1}\boldsymbol{L}$. The assumption $\boldsymbol{L} \preccurlyeq \beta \boldsymbol{X}$ implies that the eigenvalues of $\boldsymbol{X}^{-1}\boldsymbol{L}$ are at most β, so $0 \le \lambda \le \beta$. We have chosen the value of δ precisely to guarantee that, under this condition on λ, the value of (13.9) is at least 1. □

This error crucially depends only on $\boldsymbol{L}$, which for any choice of F can be upper bounded by $\boldsymbol{M}$. Propagating the error this way allows us to prove the guarantees for Jacobi

Proof. (of Theorem 13.3.1) Consider the matrix $\boldsymbol{L}_{[F,F]}$ generated when calling Jacobi with $\boldsymbol{M}_{[F,F]}$. $\boldsymbol{M}$ being bDD means for each i we have

$$\boldsymbol{M}_{[i,i]} \succcurlyeq \boldsymbol{I}_{r} \cdot \sum_{j \neq i} \|\boldsymbol{M}_{[i,j]}\|,$$

which means for all $i \in F$

$$\boldsymbol{M}_{[i,i]} - \boldsymbol{I}_r \cdot \sum_{j \neq i, j \in F} \|\boldsymbol{M}_{[i,j]}\| \succcurlyeq \boldsymbol{I}_r \cdot \sum_{j \notin F} \|\boldsymbol{M}_{[i,j]}\|.$$

Therefore if we extend $\boldsymbol{L}_{[F,F]}$ onto the full matrix by putting zeros everywhere else, we have $\boldsymbol{L} \preccurlyeq \boldsymbol{M}$. Fact 13.5.3 then gives $\boldsymbol{L}_{[F,F]} \preccurlyeq Sc(\boldsymbol{M}, V \setminus F)$.

Lemma 13.3.3 gives that $\boldsymbol{X}_{[F,F]} \succcurlyeq \frac{\alpha}{2}\boldsymbol{L}_{[F,F]}$. As $\alpha \geq 4$, we can invoke Lemma 13.9.1 with $\beta = \frac{1}{2}$. Since $\frac{1+\beta}{1-\beta^{k+1}} \leq (\frac{3}{2})/(\frac{1}{2}) \leq 3$, our choice of $k = \log(3/\varepsilon)$ gives the desired error. Each of these steps involve a matrix vector multiplication in $\boldsymbol{L}_{[F,F]}$ and two linear system solves in $\boldsymbol{X}_{[F,F]}$. The former takes $O(m)$ work and $O(\log n)$ depth since the blocks of $\boldsymbol{L}_{[F,F]}$ are a subset of the blocks of $\boldsymbol{M}$, while the latter takes $O(n)$ work and $O(\log n)$ depth due to $\boldsymbol{X}_{[F,F]}$ being block-diagonal. □

13.10 Existence of Linear-Sized Approximate Inverses

In this section we prove Theorem 13.1.3, which tells us that every *bDD* matrix has a linear-sized approximate inverse. In particular, this implies that for every *bDD* matrix there is a linear-time algorithm that approximately solves systems of equations in that matrix. To save space, we will not dwell on this algorithm, but rather will develop the linear-sized approximate inverses directly. There is some cost in doing so: there are very large constants in the linear-sized approximate inverses that are not present in a simpler linear-time solver.

To obtain a $\boldsymbol{U}^T\boldsymbol{D}\,\boldsymbol{U}$ factorization from an ε-SCC in which each $\boldsymbol{M}^{(i)}{}_{[F_i F_i]}$ is 4-bDD, we employ the procedure in Figure 13-4.

Algorithm 38: $(\boldsymbol{D}, \boldsymbol{U}) =$ DECOMPOSE$\left(\boldsymbol{M}^{(1)}, \ldots, \boldsymbol{M}^{(d)}, F_1, \ldots, F_{d-1}\right)$, where each $\boldsymbol{M}^{(i)}$ is a bDD matrix and each $\boldsymbol{M}^{(i)}{}_{[F_i F_i]}$ is 4-bDD.

Use (13.7) to compute the matrix $\boldsymbol{Z}^{(i)}$ such that JACOBI$(\varepsilon_i, \boldsymbol{M}, \boldsymbol{b}) = \boldsymbol{Z}^{(i)}\boldsymbol{b}$.
For each $i < d$, write $\boldsymbol{M}^{(i)} = \boldsymbol{X}^{(i)} + \boldsymbol{L}^{(i)}$, where $\boldsymbol{X}^{(i)}$ is block diagonal and $\boldsymbol{L}^{(i)}$ is *bDD*, as in JACOBI.
Let $\boldsymbol{X}^{(d)} = \boldsymbol{I}_{|C_{d-1}|}$ and let $\widehat{\boldsymbol{U}}$ be the upper-triangular Cholesky factor of $\boldsymbol{M}^{(d)}$.
Let $\boldsymbol{D}$ be the block diagonal matrix with $\boldsymbol{D}_{[F_i,F_i]} = \boldsymbol{X}^{(i)}$, for $1 \leq i < d$, and $\boldsymbol{D}_{[C_{d-1},C_{d-1}]} = \boldsymbol{I}_{|C_{d-1}|}$.
Let $\boldsymbol{U}$ be the upper-triangular matrix with 1*s* on the diagonal, $\boldsymbol{U}_{[C_{d-1},C_{d-1}]} = \widehat{\boldsymbol{U}}$, and $\boldsymbol{U}_{[F_i,C_i]} = \boldsymbol{Z}^{(i)}\boldsymbol{M}^{(i)}{}_{[F_i,C_i]}$, for $1 \leq i < d$.

Figure 13-4: Converting a vertex sparsifier chain into $\boldsymbol{U}$ and $\boldsymbol{D}$.

Lemma 13.10.1. *On input an ε-SCC of $\boldsymbol{M}^{(0)}$ in which each $\boldsymbol{M}^{(i)}{}_{[F_i F_i]}$ is 4-bDD, the algorithm* DECOMPOSE *produces matrices $\boldsymbol{D}$ and $\boldsymbol{U}$ such that*

$$\boldsymbol{U}^T\boldsymbol{D}\,\boldsymbol{U} \approx_\gamma \boldsymbol{M},$$

where

$$\gamma \leq 2\sum_{i=0}^{d-1} \varepsilon_i + \max_i \varepsilon_i + 1/2.$$

Proof. Consider the inverse of the operator $\boldsymbol{W} = \boldsymbol{W}^{(1)}$ realized by the algorithm ApplyChain, and the operators $\boldsymbol{W}^{(i)}$ that appear in the proof of Lemma 13.3.1.

We have

$$\left(\boldsymbol{W}^{(i)}\right)^{-1} = \begin{bmatrix} \boldsymbol{I} & 0 \\ \boldsymbol{M}_{[C_i,F_i]}\boldsymbol{Z}^{(i)} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \left(\boldsymbol{Z}^{(i)}\right)^{-1} & 0 \\ 0 & \left(\boldsymbol{W}^{(i+1)}\right)^{-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{I} & \boldsymbol{Z}^{(i)}\boldsymbol{M}_{[F_i,C_i]} \\ 0 & \boldsymbol{I} \end{bmatrix},$$

and

$$\left(\boldsymbol{W}^{(d)}\right)^{-1} = \boldsymbol{M}^{(d)} = \widehat{\boldsymbol{U}}^T\widehat{\boldsymbol{U}}.$$

After expanding and multiplying the matrices in this recursive factorization, we obtain

$$\left(\boldsymbol{W}^{(1)}\right)^{-1} = \boldsymbol{U}^T \begin{bmatrix} \left(\boldsymbol{Z}^{(1)}\right)^{-1} & \dots & 0 & 0 \\ 0 & \ddots & 0 & 0 \\ 0 & \dots & \left(\boldsymbol{Z}^{(d-1)}\right)^{-1} & 0 \\ 0 & \dots & 0 & \boldsymbol{I}_{|C_{d-1}|} \end{bmatrix} \boldsymbol{U}.$$

Moreover, we know that this latter matrix is a $2\sum_{i=0}^{d-1}\varepsilon_i$ approximation of $\boldsymbol{M}$. It remains to determine the impact of replacing the matrix in the middle of this expression with $\boldsymbol{D}$.

Lemma 13.9.1 implies that each $\boldsymbol{M}_{[F_iF_i]} \approx_{\varepsilon_i} \left(\boldsymbol{Z}^{(i)}\right)^{-1}$ and Lemma 13.3.3 implies that $\boldsymbol{X}^{(i)} \approx_{1/2} \boldsymbol{M}_{[F_iF_i]}$. So, the loss in approximation quality when we substitute the diagonal matrices is $\max_i \varepsilon_i + 1/2$. □

Invoking this decomposition procedure in conjunction with the the near-optimal sparsification routine from Theorem 13.3.1 gives a nearly-linear work routine. Repeatedly picking subsets using Lemma 13.3.4 gives then gives the linear sized decomposition.

Theorem 13.1.3 (Sparsified Cholesky). *For every bDD matrix $\boldsymbol{M}$ with n block-rows there exists a diagonal matrix $\boldsymbol{D}$ and an upper triangular matrix $\boldsymbol{U}$ with $O(n)$ nonzero blocks so that*

$$\boldsymbol{U}^T\boldsymbol{D}\,\boldsymbol{U} \approx_{3/4} \boldsymbol{M}.$$

Moreover, linear equations in $\boldsymbol{U}$, $\boldsymbol{U}^T$, and $\boldsymbol{D}$ can be solved with linear work in depth $O(\log^2 n)$, and these matrices can be computed in polynomial time.

Proof. We set $\alpha = 4$ throughout and $\varepsilon_i = 1/8(i+2)^2$. Theorem 13.3.1 then guarantees that the average number of nonzero blocks in each column of $\boldsymbol{M}^{(i)}$ is at most $10r/\varepsilon_i^2 = 640r(i+2)^4$. If we now apply Lemma 13.3.4 to find 4-diagonally dominant subsets F_i of each $\boldsymbol{M}^{(i)}$, we find that each such subset contains at least a $1/80$ fraction of the block columns of its matrix and that each column and row of $\boldsymbol{M}^{(i)}$ indexed by F has at most $1280r(i+2)^4$ nonzero entries. This implies that each row of $\boldsymbol{Z}^{(i)}\boldsymbol{M}_{[F_i,C_i]}$ has at most $(1280r(i+2)^4)^{k_i+1}$ nonzero entries.

Let n_i denote the number of block columns of $\boldsymbol{M}^{(i)}$. By induction, we know that

$$n_i \le n\left(1 - \frac{1}{80}\right)^{i-1}.$$

So, the total number of nonzero blocks in $\boldsymbol{U}$ is at most

$$\sum_{i=1}^{d} n_i(1280r(i+2)^4)^{k_i+1} \le n\sum_{i=1}^{d}\left(1-\frac{1}{80}\right)^{i-1}(1280r(i+2)^4)^{k_i+1}.$$

We will show that the term multiplying n in this later expression is upper bounded by a constant. To see this, note that $k_i \leq 1 + \log(2\varepsilon_i^{-1}) \leq \nu \log(i+1)$ for some constant ν. So, there is some other constant μ for which

$$(1280r(i+2)^4)^{k_i+1} \leq \exp(\mu \log^2(i+1)).$$

This implies that the sum is at most

$$\sum_{i \geq 1} \exp(\mu \log^2(i+1) - i/80),$$

which is bounded by a constant.

To bound the quality of the approximation, we compute

$$2\sum_i \varepsilon_i + \max_i \varepsilon_i + 1/2 = 2\sum_i 1/8(i+2)^2 + 1/72 + 1/2 < 3/4.$$

The claimed bound on the work to perform backwards and forwards substitution with $\boldsymbol{U}$ is standard: these operations require work linear in the number of nonzero entries of $\boldsymbol{U}$. The bound on the depth follows from the fact that the substitutions can be performed level-by-level, take depth $O(\log n)$ for each level, and the number of levels, d, is logarithmic in n. □

13.11 Spectral Vertex Sparsification Algorithm

In this section, we give a proof of the following lemma that immediately implies Lemma 13.3.4.

Lemma 13.11.1. *Let $\boldsymbol{M}$ be a bDD matrix with index set V, and m nonzero blocks. Let $F \subseteq V$ be such that $\boldsymbol{M}_{[F,F]}$ is α-bDD for some $\alpha \geq 4$. The algorithm* ApproxSchur$(\boldsymbol{M}, F, \varepsilon)$*, returns a matrix $\widetilde{\boldsymbol{M}}_{SC}$ s.t.*

1. *$\widetilde{\boldsymbol{M}}_{SC}$ has $O(m(\varepsilon^{-1}\log\log\varepsilon^{-1})^{O(\log\log\varepsilon^{-1})})$ nonzero blocks, and*
2. *$\widetilde{\boldsymbol{M}}_{SC} \approx_\varepsilon Sc(\boldsymbol{M}, F)$,*

in $O(m(\varepsilon^{-1}\log\log\varepsilon^{-1})^{O(\log\log\varepsilon^{-1})})$ work and $O(\log n(\log\log\varepsilon^{-1}))$ depth.

Moreover, if $\boldsymbol{T}$ is a matrix such that only the submatrix $\boldsymbol{T}_{[C,C]}$ is nonzero, and $\boldsymbol{M} + \boldsymbol{T}$ is bDD, then ApproxSchur$(\boldsymbol{M} + \boldsymbol{T}, F, \varepsilon) =$ ApproxSchur$(\boldsymbol{M}, F, \varepsilon) + \boldsymbol{T}_{[C,C]}$.

We show how to sparsify the Schur complement of $\boldsymbol{M}$ after eliminating a set of indices F such that $\boldsymbol{M}_{[F,F]}$ is an α-bDD matrix. The procedure ApproxSchur is described in Figure 13-5 and uses two key subroutines SchurSquare and LastStep. SchurSquare allows us to approximate $Sc(\boldsymbol{M}, F)$ as the Schur complement of another matrix $\boldsymbol{M}_1$ such that $\boldsymbol{M}_{1[F,F]}$ is roughly α^2-bDD. LastStep allows us to approximate $Sc(\boldsymbol{M}, F)$ with $O(\varepsilon)$ error, when $\boldsymbol{M}_{[F,F]}$ is roughly $1/\varepsilon$-bDD.

Lemma 13.11.2. *Let $\boldsymbol{M}$ be an bDD matrix with index set V, and m nonzero blocks and let $F \subseteq V$ be such that $\boldsymbol{M}_{[F,F]}$ is α-bDD for some $\alpha \geq 4$. Given $\varepsilon < 1/2$, the algorithm* SchurSquare$(\boldsymbol{M}, F, \varepsilon)$*, returns a bDD matrix $\boldsymbol{M}_1$ in $O(m\varepsilon^{-4})$ work and $O(\log n)$ depth, such that*

1. *$Sc(\boldsymbol{M}, F) \approx_\varepsilon Sc(\boldsymbol{M}_1, F)$, and*
2. *$(\boldsymbol{M}_1)_{[F,F]}$ is $\alpha^2/2$-bDD.*
3. *$\boldsymbol{M}_1$ has $O(m\varepsilon^{-4})$ nonzero blocks,*

Algorithm 39: $\widetilde{\boldsymbol{M}}_{SC} = \textsc{ApproxSchur}\left(\boldsymbol{M}, F, \alpha, \varepsilon\right)$

Initialize $\boldsymbol{M}^{(0)} \leftarrow \boldsymbol{M}$, $d = 2\log_2 \log_{\alpha/2}\left(4\varepsilon^{-1}\right)$
for $i = 1, \cdots, d$ **do**
  $\boldsymbol{M}^{(i)} \leftarrow \textsc{SchurSquare}\left(\boldsymbol{M}^{(i-1)}, F, \frac{\varepsilon}{2d}\right)$.
end
$\widetilde{\boldsymbol{M}}_{SC} \leftarrow \textsc{LastStep}\left(\boldsymbol{M}^{(d)}, F, 4\varepsilon^{-1}, \frac{\varepsilon}{4}\right)$.
Output $\widetilde{\boldsymbol{M}}_{SC}$.

Figure 13-5: Pseudocode for Computing Spectral Vertex Sparsifiers

If $\boldsymbol{T}$ is a matrix such that only the submatrix $\boldsymbol{T}_{[C,C]}$ is nonzero, and $\boldsymbol{M} + \boldsymbol{T}$ is bDD, then

$$\textsc{SchurSquare}(\boldsymbol{M} + \boldsymbol{T}, F, \varepsilon) = \textsc{SchurSquare}(\boldsymbol{M}, F, \varepsilon) + \boldsymbol{T}_{[C,C]}.$$

We can repeatedly applying the above lemma, to approximate $Sc\left(\boldsymbol{M}, F\right)$ as $Sc\left(\boldsymbol{M}_1, F\right)$, where $\boldsymbol{M}_1$ is $O(\varepsilon^{-1})$-bDD. LastStep allows us to approximate the Schur complement for such a strongly block diagonally dominant matrix $\boldsymbol{M}_1$. The guarantees of LastStep are given by the following lemma.

Lemma 13.11.3. *Let $\boldsymbol{M}$ be an bDD matrix with index set V, and m nonzero blocks and let $F \subseteq V$ be such that $\boldsymbol{M}_{[F,F]}$ is α-bDD for some $\alpha \geq 4$. There exist a procedure* LastStep *such that* $\textsc{LastStep}(\boldsymbol{M}, F, \alpha, \varepsilon)$ *returns in $O(m\varepsilon^{-8})$ work and $O(\log n)$ depth a matrix $\widetilde{\boldsymbol{M}}_{SC}$ s.t. $\widetilde{\boldsymbol{M}}_{SC}$ has $O(m\varepsilon^{-8})$ nonzero blocks and $\widetilde{\boldsymbol{M}}_{SC} \approx_{\varepsilon+2/\alpha} Sc\left(\boldsymbol{M}, F\right)$. If $\boldsymbol{T}$ is a matrix such that only the submatrix $\boldsymbol{T}_{[C,C]}$ is nonzero, and $\boldsymbol{M} + \boldsymbol{T}$ is bDD, then* $\textsc{LastStep}(\boldsymbol{M} + \boldsymbol{T}, F, \alpha, \varepsilon) = \textsc{LastStep}(\boldsymbol{M}, F, \alpha, \varepsilon) + \boldsymbol{T}_{[C,C]}$.

Combining the above two lemmas, we obtain a proof of Lemma 13.3.4.

Proof. (of Lemma 13.11.1). By induction, after i steps of the main loop in ApproxSchur,

$$Sc\left(\boldsymbol{M}, F\right) \approx_{\frac{\varepsilon i}{2d}} Sc\left(\boldsymbol{M}^{(i)}, F\right).$$

Lemma 13.11.2 also implies that $\boldsymbol{M}^{(i)}$ is $2(\frac{\alpha}{2})^{2^i}$-bDD. Thus, we have that $\boldsymbol{M}^{(d)}_{FF}$ is $8\varepsilon^{-1}$-strongly diagonally dominant at the last step. Hence, Lemma 13.11.3 then gives

$$Sc\left(\boldsymbol{M}^{(d)}, F\right) \approx_{\frac{1}{2}\varepsilon} \textsc{LastStep}\left(\boldsymbol{M}^{(d)}, F, 4\varepsilon^{-1}, \frac{\varepsilon}{4}\right).$$

Composing this bound with the guarantees of the iterations then gives the bound on overall error.

The property that if $\boldsymbol{T}$ is a matrix such that only the submatrix $\boldsymbol{T}_{[C,C]}$ is nonzero, and $\boldsymbol{M} + \boldsymbol{T}$ is bDD, then $\textsc{ApproxSchur}(\boldsymbol{M} + \boldsymbol{T}, F, \varepsilon) = \textsc{ApproxSchur}(\boldsymbol{M}, F, \varepsilon) + \boldsymbol{T}_{[C,C]}$ follows from Lemma 13.11.2 and 13.11.3, which ensure that this property holds for all our calls to SchurSquare and LastStep.

The work of these steps, and the size of the output graph follow from Lemma 13.11.2 and 13.11.3. □

13.11.1 Iterative Squaring and Sparsification

In this section, we give a proof of Lemma 13.11.2. At the core of procedure is a *squaring* identity that preserves Schur complements, and efficient sparsification of special classes of bDD matrices that we

call product demand block-Laplacians.

Definition 13.11.4. The product demand block-Laplacian of a vector $\boldsymbol{d} \in (\mathbb{C}^{r\times r})^n$, is a bDD matrix $\boldsymbol{L}_{G(\boldsymbol{d})} \in (\mathbb{C}^{r\times r})^{n\times n}$, defined as

$$(\boldsymbol{L}_{G(\boldsymbol{d})})_{[i,j]} = \begin{cases} -\boldsymbol{d}_{[i]}\boldsymbol{d}_{[j]}^* & i \neq j, \\ \boldsymbol{I}_r \cdot \|\boldsymbol{d}_{[i]}\| \sum_{k:k\neq i} \|\boldsymbol{d}_{[k]}\| & \text{otherwise.} \end{cases}$$

Definition 13.11.5. Given a vector $\boldsymbol{d} \in (\mathbb{C}^{r\times r})^n$ and an index set $F \subseteq V$, let $C = V \setminus F$. The bipartite product demand block-Laplacian of $(\boldsymbol{d}, F)$ is a bDD matrix $\boldsymbol{L}_{G(\boldsymbol{d},F)} \in (\mathbb{C}^{r\times r})^{n\times n}$, defined as

$$(\boldsymbol{L}_{G(\boldsymbol{d},F)})_{[i,j]} = \begin{cases} \boldsymbol{I}_r \cdot \|\boldsymbol{d}_{[i]}\| \sum_{k\in C} \|\boldsymbol{d}_{[k]}\| & i = j, \text{ and } i \in F \\ \boldsymbol{I}_r \cdot \|\boldsymbol{d}_{[i]}\| \sum_{k\in F} \|\boldsymbol{d}_{[k]}\| & i = j, \text{ and } i \in C \\ 0 & i \neq j, \text{ and } (i,j \in F \text{ or } i,j \in C) \\ -\boldsymbol{d}_{[i]}\boldsymbol{d}_{[j]}^* & i \neq j, \text{otherwise.} \end{cases}$$

In Section 13.11.4, we prove the following lemmas that allows us to efficiently construct sparse approximations to these matrices.

Lemma 13.11.6. *There is a routine* CliqueSparsification$(\boldsymbol{d}, \varepsilon)$ *such that for any demand vector* $\boldsymbol{d} \in (\mathbb{C}^{r\times r})^n$, *and* $\varepsilon > 0$, CliqueSparsification$(\boldsymbol{d}, \varepsilon)$ *returns in* $O(n\varepsilon^{-4})$ *work and* $O(\log n)$ *depth a bDD matrix* $\boldsymbol{L}_H$ *with* $O(n\varepsilon^{-4})$ *nonzero blocks such that*

$$\boldsymbol{L}_H \approx_\varepsilon \boldsymbol{L}_{G(\boldsymbol{d})}.$$

Lemma 13.11.7. *There is a routine* BipartiteCliqueSparsification$(\boldsymbol{d}, F, \varepsilon)$ *such that for any demand vector* $\boldsymbol{d} \in (\mathbb{C}^{r\times r})^n$, $\varepsilon > 0$, *and* $F \subseteq V$, BipartiteCliqueSparsification$(\boldsymbol{d}, \varepsilon)$ *returns in* $O(n\varepsilon^{-4})$ *work and* $O(\log n)$ *depth a bDD matrix* $\boldsymbol{L}_H$ *with* $O(n\varepsilon^{-4})$ *nonzero blocks such that*

$$\boldsymbol{L}_H \approx_\varepsilon \boldsymbol{L}_{G(\boldsymbol{d},F)}.$$

Moreover, $(\boldsymbol{L}_H)_{[F,F]}$, *and* $(\boldsymbol{L}_H)_{[C,C]}$ *are block-diagonal, where* $C = V \setminus F$.

We now use these efficient sparse approximations to give a proof of the guarantees of the procedure SchurSquare.

Algorithm 40: $\boldsymbol{M}_1 =$ SchurSquare$(\boldsymbol{M}, F, \varepsilon)$

Construct sparse representations of $\boldsymbol{d}_F^{(j)}, \boldsymbol{d}_C^{(j)}, \boldsymbol{f}^{(j)}$ defined by Eqs. (13.11), (13.12), and (13.13).
Construct $\boldsymbol{M}_4$ as given by Eq. (13.14).
Compute $\boldsymbol{M}_3$ as given by Eq. (13.15).
For all $j \in F$, let $\tilde{\boldsymbol{L}}_{G(\boldsymbol{d}_F^{(j)})} \leftarrow$ CliqueSparsification$(\boldsymbol{d}_F, \varepsilon)$.
For all $j \in F$, let $\tilde{\boldsymbol{L}}_{G(\boldsymbol{d}_C^{(j)})} \leftarrow$ CliqueSparsification$(\boldsymbol{d}_C^{(j)}, \varepsilon)$.
For all $j \in F$, let $\tilde{\boldsymbol{L}}_{G(\boldsymbol{f}^{(j)},F)} \leftarrow$ BipartiteCliqueSparsification$(\boldsymbol{f}^{(j)}, \varepsilon)$.
Construct and return $\boldsymbol{M}_1$ defined by Eq. (13.16).

Figure 13-6: Pseudocode for procedure SchurSquare Iterative Squaring and Sparsification

Proof. (of Lemma 13.11.2) Let C denote $V \setminus F$. Write $\boldsymbol{M}_{[F,F]}$ as $\boldsymbol{D} - \boldsymbol{A}$, where $\boldsymbol{D}$ is a block-diagonal, and $\boldsymbol{A}$ has its diagonal blocks as zero. The proof is based on the identity $Sc(\boldsymbol{M}, F) = Sc(\boldsymbol{M}_2, F)$, where $\boldsymbol{M}_2$ is the following matrix.

$$\boldsymbol{M}_2 = \frac{1}{2}\begin{bmatrix} \boldsymbol{D} - \boldsymbol{A}\boldsymbol{D}^{-1}\boldsymbol{A} & \boldsymbol{M}_{[F,C]} + \boldsymbol{A}\boldsymbol{D}^{-1}\boldsymbol{M}_{[F,C]} \\ \boldsymbol{M}_{[C,F]} + \boldsymbol{M}_{[C,F]}\boldsymbol{D}^{-1}\boldsymbol{A} & 2\boldsymbol{M}_{[C,C]} - \boldsymbol{M}_{[C,F]}\boldsymbol{D}^{-1}\boldsymbol{M}_{[F,C]} \end{bmatrix}. \tag{13.10}$$

It is straightforward to prove that $\boldsymbol{M}_2$ satisfies $Sc(\boldsymbol{M}, F) = Sc(\boldsymbol{M}_2, F)$ (see Lemma 13.11.8). is also straightforward to show that $\boldsymbol{D} - \boldsymbol{A}\boldsymbol{D}^{-1}\boldsymbol{A}$ is $((\alpha+1)^2 - 1)$-bDD. Thus, $\boldsymbol{M}_2$ satisfies the first two requirements.

However, $\boldsymbol{M}_2$ is likely to be a dense matrix, and we will not construct it in full. The key observation is that $\boldsymbol{M}_2$ can be written as a sum of an explicit sparse bDD matrix, and several product demand block-Laplacians. Formally, for every $j \in F$, we define $\boldsymbol{d}_F^{(j)}, \boldsymbol{d}_C^{(j)} \in (\mathbb{C}^{r\times r})^n$ as follows

$$(\boldsymbol{d}_F^{(j)})_{[i]} = \begin{cases} \boldsymbol{A}_{[i,j]}\boldsymbol{D}_{[j,j]}^{-1/2} & i \in F, \\ 0 & i \in C. \end{cases} \tag{13.11}$$

$$(\boldsymbol{d}_C^{(j)})_{[i]} = \begin{cases} 0 & i \in F, \\ \boldsymbol{M}_{[i,j]}\boldsymbol{D}_{[j,j]}^{-1/2} & i \in C. \end{cases} \tag{13.12}$$

For every $j \in F$, we need to define a bipartite product demand block-Laplacian given by $\boldsymbol{f}^{(j)} \in (\mathbb{C}^{r\times r})^n$ defined as

$$\boldsymbol{f}_{[i]}^{(j)} = \begin{cases} \boldsymbol{A}_{[i,j]}\boldsymbol{D}_{[j,j]}^{-1/2} & i \in F, \\ -\boldsymbol{M}_{[i,j]}\boldsymbol{D}_{[j,j]}^{-1/2} & i \in C. \end{cases} \tag{13.13}$$

Letting $\deg_j$ denote the number of nonzero blocks in $\boldsymbol{M}_{[j]}$, the number of nonzero blocks in each of $\boldsymbol{d}_F^{(j)}, \boldsymbol{d}_C^{(j)}, \boldsymbol{f}^{(j)}$ is at most $\deg_j$. We construct a sparse representation of each of them using $O(\deg_j)$ work. Thus, the total number of nonzero blocks in all these block-vectors is at most $O(m)$ and we can explicitly construct sparse representations for them using $O(m)$ work and $O(\log n)$ depth.

We can now express $\boldsymbol{M}_2$ as

$$\boldsymbol{M}_2 = \boldsymbol{M}_3 + \tfrac{1}{2}\textstyle\sum_{j\in F} \boldsymbol{L}_{G(\boldsymbol{d}_F^{(j)})} + \tfrac{1}{2}\sum_{j\in F} \boldsymbol{L}_{G(\boldsymbol{d}_C^{(j)})} + \tfrac{1}{2}\sum_{j\in F} \boldsymbol{L}_{G(\boldsymbol{f}^{(j)},F)}.$$

For all $i, j \in V$, we compute $\beta_{i,j} = \|\boldsymbol{f}_i^{(j)}\|$. Define $\boldsymbol{M}_4$ to be the block diagonal matrix such that

$$\boldsymbol{M}_{4[i,i]} = \tfrac{1}{2}\boldsymbol{I}_r \cdot \textstyle\sum_{j\in F} \beta_{i,j} \left(\left(\sum_{k\in V} \beta_{k,j}\right) - \beta_{i,j}\right). \tag{13.14}$$

Since at most m of $\beta_{i,j}$ are nonzero, and we can compute $\beta_{i,j}$ and construct $\boldsymbol{M}_4$ using using $O(m)$ work and $O(\log n)$ depth. It is easy to verify that we can express $\boldsymbol{M}_3$ explicitly as

$$\boldsymbol{M}_3 = \tfrac{1}{2}\begin{bmatrix} \boldsymbol{D} & \boldsymbol{M}_{[F,C]} \\ \boldsymbol{M}_{[C,F]} & 2\boldsymbol{M}_{[C,C]} \end{bmatrix} - \boldsymbol{M}_4. \tag{13.15}$$

For each j, we use Lemmas 13.11.6 and 13.11.7 to construct, in $O(\deg_j \varepsilon^{-4})$ work and $O(\log n)$ depth, bDD matrices $\tilde{\boldsymbol{L}}_{G(\boldsymbol{d}_F^{(j)})}, \tilde{\boldsymbol{L}}_{G(\boldsymbol{d}_C^{(j)})}, \tilde{\boldsymbol{L}}_{G(\boldsymbol{f}^{(j)},F)}$, each with $O(\deg_j \varepsilon^{-4})$ nonzero blocks such that

$$\tilde{\boldsymbol{L}}_{G(\boldsymbol{d}_F^{(j)})} \approx_\varepsilon \boldsymbol{L}_{G(\boldsymbol{d}_F^{(j)})}, \quad \tilde{\boldsymbol{L}}_{G(\boldsymbol{d}_C^{(j)})} \approx_\varepsilon \boldsymbol{L}_{G(\boldsymbol{d}_C^{(j)})}, \quad \text{and} \quad \tilde{\boldsymbol{L}}_{G(\boldsymbol{f}^{(j)},F)} \approx_\varepsilon \boldsymbol{L}_{G(\boldsymbol{f}^{(j)},F)}.$$

We can now construct the matrix

$$\boldsymbol{M}_1 = \boldsymbol{M}_3 + \tfrac{1}{2}\textstyle\sum_{j\in F} \tilde{\boldsymbol{L}}_{G(\boldsymbol{d}_F^{(j)})} + \tfrac{1}{2}\sum_{j\in F} \tilde{\boldsymbol{L}}_{G(\boldsymbol{d}_C^{(j)})} + \tfrac{1}{2}\sum_{j\in F} \tilde{\boldsymbol{L}}_{G(\boldsymbol{f}^{(j)},F)}, \tag{13.16}$$

in $O(m\varepsilon^{-4})$ time and $O(\log n)$ depth. $\boldsymbol{M}_1$ has $O(m\varepsilon^{-4})$ nonzero blocks and is our required matrix.

We first show that $\boldsymbol{M}_3$ is bDD. Using $\beta_{k,j} \leq \|\boldsymbol{D}_{[j,j]}^{-1/2}\boldsymbol{M}_{[i,k]}\|$ for all $j \in F, k \in V$, we have for all $i \in F$,

$$2\|\boldsymbol{M}_{4[i,i]}\| \leq \sum_{j\in F}\beta_{i,j}\sum_{k\in V}\beta_{k,j} \leq \sum_{j\in F}\|\boldsymbol{A}_{[i,j]}\|\|\boldsymbol{D}_{[j,j]}^{-1/2}\|^2\sum_{k\in V}\|\boldsymbol{M}_{[j,k]}\| \leq \sum_{j\in F}\|\boldsymbol{A}_{[i,j]}\|, \tag{13.17}$$

where the last inequality uses that $\boldsymbol{M}$ is bDD. Again, using $\boldsymbol{M}$ is bDD, for all $i \in F$, we have

$$\boldsymbol{D}_{[i,i]} \succcurlyeq \boldsymbol{I}_r\sum_{j\in F}\|\boldsymbol{A}_{[i,j]}\| + \boldsymbol{I}_r\sum_{k\in C}\|\boldsymbol{M}_{[i,k]}\|.$$

Now combining with Eqs. (13.15) and (13.17), we obtain $\boldsymbol{M}_{3[i,i]} \succcurlyeq \boldsymbol{I}_r\sum_{k\in C}\|\boldsymbol{M}_{[i,k]}\|$. Thus, $\boldsymbol{M}_3$ is bDD, and hence $\boldsymbol{M}_1$ is bDD.

We also have $\boldsymbol{M}_1 \approx_\varepsilon \boldsymbol{M}_2$. Thus, using Fact 13.5.3, we obtain $Sc(\boldsymbol{M}_1, F) \approx_\varepsilon Sc(\boldsymbol{M}_2, F)$. It remains to show that $\boldsymbol{M}_{1[F,F]}$ is $\alpha^2/2$-bDD.

The key observation is that only the matrices $\tilde{\boldsymbol{L}}_{G(\boldsymbol{d}_C^{(j)})}$ contribute to off-diagonal blocks in $\boldsymbol{M}_{1[F,F]}$. By construction,

$$\forall i,j \in F, \quad \boldsymbol{L}_{G(\boldsymbol{d}_F^{(j)})_{[i,i]}} = \boldsymbol{I}_r\sum_{k\in F:k\neq i}\beta_{i,j}\beta_{k,j}.$$

Thus, for all $i \in F$

$$\begin{aligned}\sum_{j\in F}\|\boldsymbol{L}_{G(\boldsymbol{d}_F^{(j)})_{[i,i]}}\| \leq \sum_{j\in F}\sum_{k\in F}\beta_{i,j}\beta_{k,j} &\leq \sum_{j\in F}\|\boldsymbol{A}_{[i,j]}\|\|\boldsymbol{D}_{[j,j]}^{-1/2}\|^2\sum_{k\in F}\|\boldsymbol{A}_{[j,k]}\| \\ &\leq (1+\alpha)^{-1}\boldsymbol{I}_r\cdot\sum_{j\in F}\|\boldsymbol{A}_{[i,j]}\|, \end{aligned}\tag{13.18}$$

where the last inequality follows since $\boldsymbol{D} - \boldsymbol{A} = \boldsymbol{M}_{[F,F]}$ is α-bDD. Moreover $\boldsymbol{D} - \boldsymbol{A}$ being α-bDD implies that for each $i \in F$, $\boldsymbol{D}_{[i,i]} \succcurlyeq (\alpha+1)\boldsymbol{I}_r\sum_{j\in F}\|\boldsymbol{A}_{[i,j]}\|$. Thus, using Eqs. (13.15) and (13.17), we obtain, $2\boldsymbol{M}_{3[i,i]} \succcurlyeq \alpha\boldsymbol{I}_r\sum_{j\in F}\|\boldsymbol{A}_{[i,j]}\|$.

Since for each j, $\tilde{\boldsymbol{L}}_{G(\boldsymbol{d}^{(j)})} \approx_\varepsilon \tilde{\boldsymbol{L}}_{G(\boldsymbol{d}^{(j)})}$, we have $(\tilde{\boldsymbol{L}}_{G(\boldsymbol{d}^{(j)})})_{[i,i]} \preccurlyeq e^\varepsilon(\boldsymbol{L}_{G(\boldsymbol{d}^{(j)})})_{[i,i]}$. Thus, $\sum_j \tilde{\boldsymbol{L}}_{G(\boldsymbol{d}_F^{(j)})}$ is bDD with sums of norms of off-diagonal blocks at most $e^\varepsilon(1+\alpha)^{-1}\sum_{j\in F}\|\boldsymbol{A}_{[i,j]}\|$. Since these are the only matrices that contribute to off-diagonal blocks in $\boldsymbol{M}_{1[F,F]}$, we get that $\boldsymbol{M}_{1[F,F]}$ is at least $\frac{\alpha}{e^\varepsilon(1+\alpha)^{-1}}$-bDD. Using $\varepsilon < 1/2$ gives us our claim.

It is easy to verify that our transformations maintain that if $\boldsymbol{T}$ is a matrix that is only nonzero inside the submatrix $\boldsymbol{T}_{[C,C]}$, and $\boldsymbol{M} + \boldsymbol{T}$ is bDD, then

$$\textsc{SchurSquare}(\boldsymbol{M} + \boldsymbol{T}, F, \varepsilon) = \textsc{SchurSquare}(\boldsymbol{M}, F, \varepsilon) + \boldsymbol{T}_{[C,C]}.$$

□

We now prove the claims deferred from the above proof.

Lemma 13.11.8. *The matrix $\boldsymbol{M}_2$ defined by Eq. (13.10) satisfies $Sc(\boldsymbol{M}_2, F) = Sc(\boldsymbol{M}, F)$.*

Proof. We need the following identity from [223]:

$$(\boldsymbol{D} - \boldsymbol{A})^{-1} = 1/2\cdot(\boldsymbol{D}^{-1} + (\boldsymbol{I} + \boldsymbol{D}^{-1}\boldsymbol{A})(\boldsymbol{D} - \boldsymbol{A}\boldsymbol{D}^{-1}\boldsymbol{A})^{-1}(\boldsymbol{I} + \boldsymbol{A}\boldsymbol{D}^{-1})). \tag{13.19}$$

We have,

$$\begin{aligned}
Sc(\boldsymbol{M}_2, F) &= \boldsymbol{M}_{[C,C]} - 1/2 \cdot \boldsymbol{M}_{[C,F]} \boldsymbol{D}^{-1} \boldsymbol{M}_{[F,C]} \\
&\quad - 1/2 \cdot \boldsymbol{M}_{[C,F]} (\boldsymbol{I} + \boldsymbol{D}^{-1}\boldsymbol{A})(\boldsymbol{D} - \boldsymbol{A}\boldsymbol{D}^{-1}\boldsymbol{A})^{-1}(\boldsymbol{I} + \boldsymbol{A}\boldsymbol{D}^{-1}) \boldsymbol{M}_{[F,C]} \\
&= \boldsymbol{M}_{[C,C]} \\
&\quad - 1/2 \cdot \boldsymbol{M}_{[C,F]} (\boldsymbol{D}^{-1} + (\boldsymbol{I} + \boldsymbol{D}^{-1}\boldsymbol{A})(\boldsymbol{D} - \boldsymbol{A}\boldsymbol{D}^{-1}\boldsymbol{A})^{-1}(\boldsymbol{I} + \boldsymbol{A}\boldsymbol{D}^{-1})) \boldsymbol{M}_{[F,C]} \\
&= \boldsymbol{M}_{[C,C]} - \boldsymbol{M}_{[C,F]} (\boldsymbol{D} - \boldsymbol{A})^{-1} \boldsymbol{M}_{[F,C]} \\
&= Sc(\boldsymbol{M}, F) .
\end{aligned}$$

□

13.11.2 Schur Complement w.r.t. Highly α-bDD Submatrices

Algorithm 41: $\widetilde{\boldsymbol{M}}_{SC} = \textsc{LastStep}(\boldsymbol{M}, F, \alpha, \varepsilon)$

Compute $\boldsymbol{X}$, $\boldsymbol{D}$, and $\boldsymbol{A}$ as given by equations (13.20), (13.21), and (13.22).
Construct $\boldsymbol{Y}$ as defined by equations (13.33), (13.34), and (13.35).
Construct sparse vectors $\boldsymbol{d}^{(j)}, \boldsymbol{g}^{(j)}$ for all $j \in F$ as defined by equations (13.26) and (13.27).
For all $j \in F$, let $\tilde{\boldsymbol{L}}_{G(\boldsymbol{d}^{(j)},F)} \leftarrow \textsc{BipartiteCliqueSparsification}(\boldsymbol{d}^{(j)}, F, \varepsilon/2)$
For all $j \in F$, let $\tilde{\boldsymbol{L}}_{G(\boldsymbol{g}^{(j)})} \leftarrow \textsc{CliqueSparsification}(\boldsymbol{g}^{(j)}, \varepsilon/2)$.
Compute $\boldsymbol{R}$ as given by equation (13.29).
Construct sparse vectors $\boldsymbol{r}^{(j)}$ for all $j \in F$ as defined by equations (13.30).
For all $j \in F$, let $\tilde{\boldsymbol{L}}_{G(\boldsymbol{r}^{(j)})} \leftarrow \textsc{CliqueSparsification}(\boldsymbol{r}^{(j)}, \varepsilon/2)$.
Compute $\boldsymbol{S}$ as given by equation (13.31).
Output

$$\widetilde{\boldsymbol{M}}_{SC} = \boldsymbol{S} + \sum_i \tilde{\boldsymbol{L}}_{G(\boldsymbol{r}^{(i)})}.$$

Figure 13-7: Pseudocode for procedure LastStep: Computing an approximate Schur complement w.r.t. a highly α-bDD submatrices.

In this section, we describe the LastStep procedure for computing an approximate Schur complement of a bDD matrix $\boldsymbol{M}$ with a highly α-bDD submatrix $\boldsymbol{M}_{[F,F]}$. LastStep is the final step of the ApproxSchur algorithm. The key element of the procedure is a formula for approximating the inverse of $\boldsymbol{M}_{[F,F]}$ that is leveraged to approximate the Schur complement of $\boldsymbol{M}$ as the Schur complement of matrix with the FF submatrix being block diagonal.

We prove guarantees for the LastStep algorithm as stated in Lemma 13.11.3.

One could attempt to deal with the highly α-bDD matrix at the last step by directly replacing it with its diagonal, but this is problematic. Consider the case where F contains u and v with a weight ε edge between them, and u and v are connected to u' and v' in C by weight 1 edges respectively. Keeping only the diagonal results in a Schur complement that disconnects u' and v'. This however can be fixed by taking a step of random walk within F.

Given a bDD matrix $\boldsymbol{M}$, s.t. $\boldsymbol{M}_{[F,F]}$ is α-bDD we define a block diagonal matrix $\boldsymbol{X} \in (\mathbb{C}^{r \times r})^{|F| \times |F|}$ s.t. for each $i \in F$

$$\boldsymbol{X}_{[i,i]} = \frac{\alpha}{\alpha + 1} \boldsymbol{M}_{[i,i]} \tag{13.20}$$

and another block diagonal matrix $\boldsymbol{D} \in (\mathbb{C}^{r\times r})^{|F|\times|F|}$ s.t. for each $i \in F$

$$\boldsymbol{D}_{[i,i]} = \frac{1}{\alpha+1}\boldsymbol{M}_{[i,i]}, \tag{13.21}$$

and we define a matrix $\boldsymbol{A} \in (\mathbb{C}^{r\times r})^{|F|\times|F|}$

$$\boldsymbol{A}_{[i,j]} = \begin{cases} 0 & i = j \\ -\boldsymbol{M}_{[i,j]} & \text{otherwise.} \end{cases} \tag{13.22}$$

Thus $\boldsymbol{M}_{[F,F]} = \boldsymbol{X} + \boldsymbol{D} - \boldsymbol{A}$. One can check that because $\boldsymbol{M}$ is bDD and $\boldsymbol{M}_{[F,F]}$ is α-bDD, it follows that $\boldsymbol{D} - \boldsymbol{A}$ is bDD and the matrix

$$\begin{bmatrix} \boldsymbol{X} & \boldsymbol{M}_{[F,C]} \\ \boldsymbol{M}_{[C,F]} & \boldsymbol{M}_{[C,C]} \end{bmatrix}$$

is also bDD.

We will consider the linear operator

$$\boldsymbol{Z}^{(last)} \stackrel{\text{def}}{=} \frac{1}{2}\boldsymbol{X}^{-1} + \frac{1}{2}\boldsymbol{X}^{-1}\left(\boldsymbol{X} - \boldsymbol{D} + \boldsymbol{A}\right)\boldsymbol{X}^{-1}\left(\boldsymbol{X} - \boldsymbol{D} + \boldsymbol{A}\right)\boldsymbol{X}^{-1}. \tag{13.23}$$

We define

$$\boldsymbol{M}^{(last)} = \begin{pmatrix} \left(\boldsymbol{Z}^{(last)}\right)^{-1} & \boldsymbol{M}_{[F,C]} \\ \boldsymbol{M}_{[F,C]} & \boldsymbol{M}_{[C,C]} \end{pmatrix}$$

Lemma 13.11.9.

$$\boldsymbol{M} \preccurlyeq \boldsymbol{M}^{(last)} \preccurlyeq \left(1 + \frac{2}{\alpha}\right)\boldsymbol{M}.$$

We defer the proof of Lemma 13.11.9 to Section 13.11.3.

To utilize $\boldsymbol{Z}^{(last)}$, define

$$\boldsymbol{M}_2^{(last)} \stackrel{\text{def}}{=} \begin{bmatrix} \frac{1}{2}\boldsymbol{X} & \frac{1}{2}\left(\boldsymbol{X} - \boldsymbol{D} + \boldsymbol{A}\right)\boldsymbol{X}^{-1}\boldsymbol{M}_{[F,C]} \\ \frac{1}{2}\boldsymbol{M}_{[C,F]}\boldsymbol{X}^{-1}\left(\boldsymbol{X} - \boldsymbol{D} + \boldsymbol{A}\right) & \boldsymbol{M}_{[C,C]} - \frac{1}{2}\boldsymbol{M}_{[C,F]}\boldsymbol{X}^{-1}\boldsymbol{M}_{[F,C]}. \end{bmatrix} \tag{13.24}$$

and note

$$Sc\left(\boldsymbol{M}^{(last)}, F\right) = Sc\left(\boldsymbol{M}_2^{(last)}, F\right) \tag{13.25}$$

Lemma 13.11.9 tells us that for large enough α, we can approximate the Schur complement of $\boldsymbol{M}$ by approximating the the Schur complement of $\boldsymbol{M}_2^{(last)}$.

The next lemma tells us that M_2 is bDD and that we can write the matrix as a sum of an explicit bDD matrix and sparse implicitly represented product demand block-Laplacians and bipartite product demand block-Laplacians.

Lemma 13.11.10. *Consider a bDD matrix $\boldsymbol{M}$, where $\boldsymbol{M}_{[F,F]}$ is α-bDD for some $\alpha \geq 4$, and let $\boldsymbol{M}_2^{(last)}$ be the associated matrix defined by equation (13.24). Let m be the number of nonzero blocks of $\boldsymbol{M}$.*

For $j \in F$, we define $\boldsymbol{d}^{(i)} \in (\mathbb{C}^{r\times r})^n$

$$\boldsymbol{d}^{(j)}_{[i]} = \begin{cases} \boldsymbol{A}_{[i,j]}\boldsymbol{X}^{-1/2}_{[j,j]} & \text{for } i \in F \\ \boldsymbol{M}_{[i,j]}\boldsymbol{X}^{-1/2}_{[j,j]} & \text{for } i \in C \end{cases} \tag{13.26}$$

For $j \in F$, we define $\boldsymbol{g}^{(j)} \in (\mathbb{C}^{r\times r})^n$

$$\boldsymbol{g}^{(j)}_{[i]} = \begin{cases} 0 & \text{for } i \in F \\ \boldsymbol{M}_{[i,j]}\boldsymbol{X}^{-1/2}_{[j,j]} & \text{for } i \in C \end{cases} \tag{13.27}$$

Then

$$\boldsymbol{M}_2^{(last)} = \boldsymbol{Y} + \frac{1}{2}\sum_{j\in F}\boldsymbol{L}_{G(\boldsymbol{g}^{(i)})} + \frac{1}{2}\sum_{j\in F}\boldsymbol{L}_{G(\boldsymbol{d}^{(i)},F)}$$

where $\boldsymbol{Y}$ is bDD and has $O(m)$ nonzero blocks, and the total number of nonzero blocks in $\boldsymbol{d}^{(i)}$ and $\boldsymbol{g}^{(i)}$ for all i combined is also $O(m)$.

$\boldsymbol{Y}$ as well as $\boldsymbol{d}^{(i)}$ and $\boldsymbol{g}^{(i)}$ for all i can be computed in $O(m)$ time and $O(\log n)$ depth.

If $\boldsymbol{T}$ is a matrix that is only nonzero inside the submatrix $\boldsymbol{T}_{[C,C]}$, then if we apply the transformation of Eq. 13.24, to $\boldsymbol{M}+\boldsymbol{T}$ instead of $\boldsymbol{M}$, we find $(\boldsymbol{M}+\boldsymbol{T})_2^{(last)} = \boldsymbol{M}_2^{(last)} + \boldsymbol{T}_{[C,C]}$, and in particular $\boldsymbol{Y}(\boldsymbol{M}+\boldsymbol{T}) = \boldsymbol{Y}(\boldsymbol{M}) + \boldsymbol{T}_{[C,C]}$.

We defer the proof of Lemma 13.11.10 to Section 13.11.3.

Proof. (of Lemma 13.11.3) The procedure LastStep$(M, F, \alpha, \varepsilon)$ first computes $\boldsymbol{Y}$ and $\boldsymbol{d}^{(i)}$ and $\boldsymbol{g}^{(i)}$ for all i s.t.

$$\boldsymbol{M}_2^{(last)} = \boldsymbol{Y} + \frac{1}{2}\sum_{j\in F}\boldsymbol{L}_{G(\boldsymbol{g}^{(i)})} + \frac{1}{2}\sum_{j\in F}\boldsymbol{L}_{G(\boldsymbol{d}^{(i)},F)}, \tag{13.28}$$

where $\boldsymbol{M}_2^{(last)} \approx_{2/\alpha} \boldsymbol{M}$. Let $n_{\boldsymbol{d}^{(i)}}$ and $n_{\boldsymbol{g}^{(i)}}$ denote the number of nonzero blocks in each $\boldsymbol{d}^{(i)}$ and $\boldsymbol{g}^{(i)}$. By Lemma 13.11.6, we may call CliqueSparsification$(\boldsymbol{g}^{(i)}, \varepsilon/2)$ in $O(n_{\boldsymbol{g}^{(i)}}\varepsilon^{-4})$ time to sparsify $\boldsymbol{L}_{G(\boldsymbol{g}^{(i)})}$, producing a bDD matrix $\tilde{\boldsymbol{L}}_{G(\boldsymbol{g}^{(i)})} \approx_{\varepsilon/2} \boldsymbol{L}_{G(\boldsymbol{g}^{(i)})}$ with $O(n_{\boldsymbol{g}^{(i)}}\varepsilon^{-4})$ nonzero blocks. By Lemma 13.11.7, we may call BipartiteCliqueSparsification$(\boldsymbol{d}^{(i)}, F, \varepsilon/2)$ in $O(n_{\boldsymbol{d}^{(i)}}\varepsilon^{-4})$ time to sparsify $\boldsymbol{L}_{G(\boldsymbol{d}^{(i)},F)}$, producing a bDD matrix $\tilde{\boldsymbol{L}}_{G(\boldsymbol{d}^{(i)},F)} \approx_{\varepsilon/2} \boldsymbol{L}_{G(\boldsymbol{d}^{(i)},F)}$ with $O(n_{\boldsymbol{d}^{(i)}}\varepsilon^{-4})$ nonzero blocks. The total running time of this is $O(\varepsilon^{-4}\sum_i(n_{\boldsymbol{d}^{(i)}} + n_{\boldsymbol{g}^{(i)}})) = O(\varepsilon^{-4}m)$, and the total number of nonzero blocks in

$$\frac{1}{2}\sum_{j\in F}\tilde{\boldsymbol{L}}_{G(\boldsymbol{g}^{(i)})} + \frac{1}{2}\sum_{j\in F}\tilde{\boldsymbol{L}}_{G(\boldsymbol{d}^{(i)},F)},$$

is $O(\varepsilon^{-4}(\sum_i n_{\boldsymbol{d}^{(i)}} + n_{\boldsymbol{g}^{(i)}})) = O(\varepsilon^{-4}m)$. We define

$$\boldsymbol{R} = \boldsymbol{Y} + \frac{1}{2}\sum_{j\in F}\tilde{\boldsymbol{L}}_{G(\boldsymbol{g}^{(i)})} + \frac{1}{2}\sum_{j\in F}\tilde{\boldsymbol{L}}_{G(\boldsymbol{d}^{(i)},F)}. \tag{13.29}$$

which we can compute in $O(\varepsilon^{-4}m)$ time and $O(\log n)$ depth. We have $\boldsymbol{M} \approx_{2/\alpha} \boldsymbol{M}_2^{(last)}$ and $\boldsymbol{M}_2^{(last)} \approx_{\varepsilon/2} \boldsymbol{R}$, so that $\boldsymbol{M} \approx_{2/\alpha+\varepsilon/2} \boldsymbol{R}$. It follows from Fact 13.5.3 that $Sc(\boldsymbol{M}, F) \approx_{2/\alpha+\varepsilon/2} Sc(\boldsymbol{R}, F)$.

Because the sparsifiers computed by BipartiteCliqueSparsificationpreserve the graph bipartition, $\boldsymbol{R}_{[F,F]}$ is block diagonal.

We can use the block diagonal structure of $\boldsymbol{R}_{[F,F]}$ quickly compute a sparse approximation to $Sc(\boldsymbol{R}, F)$.

For $j \in F$, we define $\boldsymbol{g}^{(j)} \in (\mathbb{C}^{r\times r})^{|C|}$

$$\boldsymbol{r}_i^{(j)} = \boldsymbol{R}_{[i,j]}\boldsymbol{R}^{-1/2}_{[j,j]} \text{ for } i \in C \tag{13.30}$$

Then

$$Sc(\boldsymbol{R}, F) = \boldsymbol{R}_{[C,C]} - \boldsymbol{R}_{[C,F]}\boldsymbol{R}^{-1}_{[F,F]}\boldsymbol{R}_{[F,C]} = \boldsymbol{S} + \sum_{j\in F}\boldsymbol{L}_{G(\boldsymbol{r}^{(i)})}.$$

where

$$\begin{aligned}\boldsymbol{S} =& \boldsymbol{R}_{[C,C]} - \operatorname{Diag}_{i\in C}\left(\boldsymbol{I}_r \sum_{j\in F}\sum_{k\in C\setminus\{i\}} \|\boldsymbol{R}_{[i,j]}\boldsymbol{R}_{[j,j]}^{-1/2}\|\|\boldsymbol{R}_{[k,j]}\boldsymbol{R}_{[j,j]}^{-1/2}\|\right) \\ & - \operatorname{Diag}_{i\in C}\sum_{j\in F}\left(\boldsymbol{R}_{[i,j]}\boldsymbol{R}_{[j,j]}^{-1}\boldsymbol{R}_{[j,i]}\right).\end{aligned} \tag{13.31}$$

Let us define, for each block row $i \in F$, $\rho_i = \sum_{j\in C}\|\boldsymbol{R}_{[i,j]}\|$, and for each block row $i \in C$, $\rho_i = \sum_{j\in F}\|\boldsymbol{R}_{[i,j]}\|$. From $\boldsymbol{R}$ being bDD, we then conclude for each $i \in F$

$$\boldsymbol{R}_{[i,i]} \succcurlyeq \boldsymbol{I}_r\rho_i$$

and for each $i \in C$

$$\boldsymbol{R}_{[i,i]} \succcurlyeq \boldsymbol{I}_r\left(\rho_i + \sum_{j\in C}\|\boldsymbol{R}_{[i,j]}\|\right).$$

With this in mind, we check that each for $i \in C$ of $\boldsymbol{S}$ is bDD.

The diagonal block satisfies

$$\begin{aligned}\boldsymbol{S}_{[i,i]} &\succcurlyeq \boldsymbol{I}_r\left(\sum_{j\in C}\|\boldsymbol{R}_{[i,j]}\| + \rho_i - \sum_{j\in F}\sum_{k\in C}\|\boldsymbol{R}_{[i,j]}\boldsymbol{R}^{-1/2}{}_{[j,j]}\|\|\boldsymbol{R}_{[k,j]}\boldsymbol{R}^{-1/2}{}_{[j,j]}\|\right) \\ &\succcurlyeq \boldsymbol{I}_r\left(\sum_{j\in C}\|\boldsymbol{R}_{[i,j]}\| + \rho_i - \sum_{j\in F}\sum_{k\in C}\frac{1}{\rho_j}\|\boldsymbol{R}_{[i,j]}\|\|\boldsymbol{R}_{[k,j]}\|\right) \\ &\succcurlyeq \boldsymbol{I}_r\left(\sum_{j\in C}\|\boldsymbol{R}_{[i,j]}\| + \rho_i - \sum_{j\in F}\frac{1}{\rho_j}\|\boldsymbol{R}_{[i,j]}\|\right) \\ &\succcurlyeq \boldsymbol{I}_r\sum_{j\in C}\|\boldsymbol{R}_{[i,j]}\|\end{aligned}$$

We can compute $\boldsymbol{S}$ and all $\boldsymbol{r}_i^{(j)}$ in $O(m\varepsilon^{-4})$ time and $O(\log n)$ depth, since this is an upper bound to the number of nonzero blocks in $\boldsymbol{R}$. Let $n_{\boldsymbol{r}^{(i)}}$ denote the number of nonzero blocks in $\boldsymbol{r}^{(i)}$. By Lemma 13.11.6, we may call CliqueSparsification$(\boldsymbol{r}^{(i)}, \varepsilon/2)$ in $O(n_{\boldsymbol{r}^{(i)}}\varepsilon^{-4})$ time to sparsify $\boldsymbol{L}_{G(\boldsymbol{r}^{(i)})}$, producing a bDD matrix $\tilde{\boldsymbol{L}}_{G(\boldsymbol{r}^{(i)})} \approx_{\varepsilon/2} \boldsymbol{L}_{G(\boldsymbol{r}^{(i)})}$ with $O(n_{\boldsymbol{r}^{(i)}}\varepsilon^{-4})$ nonzero blocks.

The total number of nonzero blocks in

$$\boldsymbol{S} + \sum_i \tilde{\boldsymbol{L}}_{G(\boldsymbol{r}^{(i)})}$$

is also $O(\varepsilon^{-8}m)$, and $\boldsymbol{S} + \sum_i \tilde{\boldsymbol{L}}_{G(\boldsymbol{r}^{(i)})} \approx_{\varepsilon/2} Sc(\boldsymbol{R}, F)$. So

$$\widetilde{\boldsymbol{M}}_{SC} = \boldsymbol{S} + \sum_i \tilde{\boldsymbol{L}}_{G(\boldsymbol{r}^{(i)})} \approx_{\varepsilon+2/\alpha} Sc(\boldsymbol{M}, F)$$

The total amount of work to compute these $\tilde{\boldsymbol{L}}_{G(\boldsymbol{r}^{(i)})}$ is $O(\varepsilon^{-4}(\sum_i n_{\boldsymbol{r}^{(i)}})) = O(\varepsilon^{-8}m)$. The depth for this computation is $O(\log n)$.

Suppose $\boldsymbol{T}$ is a matrix that is only nonzero inside the submatrix $\boldsymbol{T}_{[C,C]}$, and $\boldsymbol{M} + \boldsymbol{T}$ is bDD. We can show that LastStep$(\boldsymbol{M} + \boldsymbol{T}, F, \alpha, \varepsilon)$ = LastStep$(\boldsymbol{M}, F, \alpha, \varepsilon) + \boldsymbol{T}_{[C,C]}$, by first noting that

this type of property holds for $\boldsymbol{Y}$ by Lemma 13.11.10, and from this concluding that similarly if we consider $\boldsymbol{R}$ as a function of $\boldsymbol{M}$ then $\boldsymbol{R}_{[C,C]}(\boldsymbol{M}+\boldsymbol{T}) = \widetilde{\boldsymbol{R}_{[C,C]}(\boldsymbol{M})} + \boldsymbol{T}_{[C,C]}$, and finally considering $\widetilde{\boldsymbol{M}}_{SC}$ as a function of $\boldsymbol{M}$, we can then easily show that $(\widetilde{\boldsymbol{M}+\boldsymbol{T}})_{SC} = \widetilde{\boldsymbol{M}}_{SC} + \boldsymbol{T}_{[C,C]}$. □

13.11.3 Deferred Proofs from Section 13.11.2

To help us prove Lemma 13.11.9, we first prove the next lemma.

Lemma 13.11.11. *If $\boldsymbol{M}_{[F,F]} = \boldsymbol{X} + \boldsymbol{D} - \boldsymbol{A}$ be a α-bDD matrix for some $\alpha \geq 4$, then the operator $\boldsymbol{Z}^{(last)}$ as defined in Equation 13.23 satisfies:*

$$\boldsymbol{M}_{[F,F]} \preccurlyeq \left(\boldsymbol{Z}^{(last)}\right)^{-1} \preccurlyeq \boldsymbol{M}_{[F,F]} + \frac{2}{\alpha}(\boldsymbol{D}-\boldsymbol{A}).$$

Proof. Composing both sides by $\boldsymbol{X}^{-1/2}$ and substituting in $\mathcal{L} = \boldsymbol{X}^{-1/2}(\boldsymbol{D}-\boldsymbol{A})\boldsymbol{X}^{-1/2}$ means it suffices to show

$$\boldsymbol{I} + \mathcal{L} \preccurlyeq \left(\frac{1}{2}\boldsymbol{I} + \frac{1}{2}(\boldsymbol{I}-\mathcal{L})^2\right)^{-1} \preccurlyeq \boldsymbol{I} + \mathcal{L} + \frac{2}{\alpha}\mathcal{L}.$$

We can use the fact that $\boldsymbol{M}_{[F,F]}$ is α-strongly diagonally dominant to show $0 \preccurlyeq \boldsymbol{L} \preccurlyeq \frac{2}{\alpha}\boldsymbol{X}$, and equivalently $0 \preccurlyeq \mathcal{L} \preccurlyeq \frac{2}{\alpha}\boldsymbol{I}$, as follows:

Firstly, $\boldsymbol{D}-\boldsymbol{A} \succcurlyeq 0$ as $\boldsymbol{D}-\boldsymbol{A}$ is bDD, similarly, $\boldsymbol{D}+\boldsymbol{A} \succcurlyeq 0$ as $\boldsymbol{D}+\boldsymbol{A}$ is also bDD. From the latter $\boldsymbol{D} \succcurlyeq -\boldsymbol{A}$, so $2\boldsymbol{D} \succcurlyeq \boldsymbol{D}-\boldsymbol{A}$. Finally, $\boldsymbol{X} \succcurlyeq \alpha\boldsymbol{D} \succcurlyeq \frac{\alpha}{2}(\boldsymbol{D}-\boldsymbol{A})$.

As $\mathcal{L}$ and $\boldsymbol{I}$ commute, the spectral theorem means it suffices to show this for any scalar $0 \leq t \leq \frac{2}{\alpha}$. Note that

$$\frac{1}{2} + \frac{1}{2}(1-t)^2 = 1 - t + \frac{1}{2}t^2$$

Taking the difference between the inverse of this and the 'true' value of $1+t$ gives:

$$\left(1 - t + \frac{1}{2}t^2\right)^{-1} - (1+t) = \frac{1-(1+t)\left(1-t+\frac{1}{2}t^2\right)}{1-t+\frac{1}{2}t^2} = \frac{\frac{1}{2}t^2(1-t)}{1-t+\frac{1}{2}t^2}$$

Incorporating the assumption that $0 \leq t \leq \frac{2}{\alpha}$ and $\alpha \geq 4$ gives that the denominator is at least $1 - \frac{2}{\alpha} \geq \frac{1}{2}$, and the numerator term can be bounded by $0 \leq \frac{t^2}{2}(1-t) \leq \frac{t}{\alpha}$. Combining these two bounds then gives the result. □

Lemma 13.11.9 allows us to extend the approximation of $\boldsymbol{M}_{[F,F]}$ by the inverse of $\boldsymbol{Z}^{(last)}$ to the entire matrix $\boldsymbol{M}$.

Proof. (of Lemma 13.11.9) Recall that when a matrix $\boldsymbol{T}$ is PSD,

$$\begin{pmatrix} \boldsymbol{T} & 0 \\ 0 & 0 \end{pmatrix} \succcurlyeq 0. \tag{13.32}$$

The left-hand inequality of our lemma follows immediately from Eq. 13.32 and the left-hand side of the guarantee of Lemma 13.11.11. To prove the right-hand inequality we apply Eq. 13.32 and the right-hand side of the guarantee of Lemma 13.11.11. to conclude

$$\begin{pmatrix} \left(\boldsymbol{Z}^{(last)}\right)^{-1} & \boldsymbol{M}_{[F,C]} \\ \boldsymbol{M}_{[F,C]} & \boldsymbol{M}_{[C,C]} \end{pmatrix} \preccurlyeq \begin{pmatrix} \boldsymbol{M}_{[F,F]} + \frac{2}{\alpha}\boldsymbol{L} & \boldsymbol{M}_{[F,C]} \\ \boldsymbol{M}_{[F,C]} & \boldsymbol{M}_{[C,C]} \end{pmatrix} = \boldsymbol{M} + \frac{2}{\alpha}\begin{pmatrix} \boldsymbol{L} & 0 \\ 0 & 0 \end{pmatrix}.$$

The matrix $\begin{bmatrix} \boldsymbol{X} & \boldsymbol{M}_{[F,C]} \\ \boldsymbol{M}_{[C,F]} & \boldsymbol{M}_{[C,C]} \end{bmatrix}$ is bDD and hence PSD. It follows that $\begin{pmatrix} \boldsymbol{D}-\boldsymbol{A} & 0 \\ 0 & 0 \end{pmatrix} \preccurlyeq \boldsymbol{M}$, by which we may conclude that $\boldsymbol{M} + \frac{2}{\alpha}\begin{pmatrix} \boldsymbol{D}-\boldsymbol{A} & 0 \\ 0 & 0 \end{pmatrix} \preccurlyeq \boldsymbol{M} + \frac{2}{\alpha}\boldsymbol{M}$. □

Proof. (of Lemma 13.11.10)

Each product demand clique $\boldsymbol{L}_{G(\boldsymbol{g}^{(i)})}$ and bipartite product demand clique $\boldsymbol{L}_{G(\boldsymbol{d}^{(i)},F)}$ is bDD.

We now have to find an expression for $\boldsymbol{Y}$ and show that $\boldsymbol{Y}$ is bDD. Let us write $\boldsymbol{Y}$ in terms of its blocks From

$$\boldsymbol{Y} = \boldsymbol{M} - \left(\frac{1}{2}\sum_{j\in F} \boldsymbol{L}_{G(\boldsymbol{g}^{(i)})} + \frac{1}{2}\sum_{j\in F} \boldsymbol{L}_{G(\boldsymbol{d}^{(i)},F)} \right)$$

it then follows by a simple check that

$$\boldsymbol{Y}^*_{[C,F]} = \boldsymbol{Y}_{[F,C]} = \frac{1}{2}(\boldsymbol{X}-\boldsymbol{D})\boldsymbol{X}^{-1}\boldsymbol{M}_{[F,C]} = \frac{1}{2}\left(1-\frac{1}{\alpha}\right)\boldsymbol{M}_{[F,C]} \tag{13.33}$$

and that

$$\boldsymbol{Y}_{[F,F]} = \frac{1}{2}\boldsymbol{X} - \frac{1}{2}\operatorname*{Diag}_{i\in F}\left(\boldsymbol{I}_r \sum_{j\in F}\sum_{k\in C} \|\boldsymbol{A}_{[i,j]}\boldsymbol{X}^{-1/2}_{[j,j]}\| \|\boldsymbol{M}_{[k,j]}\boldsymbol{X}^{-1/2}{}_{[j,j]}\| \right) \tag{13.34}$$

and

$$\begin{aligned} \boldsymbol{Y}_{[C,C]} =& \boldsymbol{M}_{[C,C]} - \frac{1}{2}\operatorname*{Diag}_{i\in C}\left(\boldsymbol{M}_{[i,F]}\boldsymbol{X}^{-1}\boldsymbol{M}_{[F,i]}\right) \\ & - \frac{1}{2}\operatorname*{Diag}_{i\in C}\left(\boldsymbol{I}_r \sum_{j\in F}\sum_{k\in C\setminus\{i\}} \|\boldsymbol{M}_{[i,j]}\boldsymbol{X}^{-1/2}_{[j,j]}\| \|\boldsymbol{M}_{[k,j]}\boldsymbol{X}^{-1/2}_{[j,j]}\| \right) \\ & - \frac{1}{2}\operatorname*{Diag}_{i\in C}\left(\boldsymbol{I}_r \sum_{j\in F}\sum_{k\in F} \|\boldsymbol{M}_{[i,j]}\boldsymbol{X}^{-1/2}_{[j,j]}\| \|\boldsymbol{A}_{[k,j]}\boldsymbol{X}^{-1/2}{}_{[j,j]}\| \right). \end{aligned} \tag{13.35}$$

Next, we check that $\boldsymbol{Y}$ is bDD. First, let us define, for each block row $i \in F$, $\rho_i = \sum_{j\in F}\|\boldsymbol{A}_{[i,j]}\|$. From $\boldsymbol{M}_{[F,F]}$ being α-bDD, we then conclude $\boldsymbol{M}_{[i,i]} \succcurlyeq \boldsymbol{I}_r(1+\alpha)\rho_i$. And from $\boldsymbol{M}$ being bDD, we find that $\sum_{j\in C}\|\boldsymbol{M}_{[i,j]}\| \le \alpha\rho_i$. We now check that each block row $i \in F$ of $\boldsymbol{Y}$ is bDD. The off-diagonal block norm sum is at most $\frac{1}{2}\left(1-\frac{1}{\alpha}\right)\alpha\rho_i$.

The diagonal satisfies

$$\begin{aligned} \boldsymbol{Y}_{[i,i]} &\succcurlyeq \boldsymbol{I}_r\left(\frac{1}{2}\alpha\rho_i - \frac{1}{2}\left(\sum_{j\in F}\sum_{k\in C}\|\boldsymbol{A}_{[i,j]}\boldsymbol{X}^{-1/2}_{[j,j]}\|\|\boldsymbol{M}_{[k,j]}\boldsymbol{X}^{-1/2}_{[j,j]}\|\right)\right) \\ &\succcurlyeq \boldsymbol{I}_r\left(\frac{1}{2}\alpha\rho_i - \frac{1}{2}\left(\sum_{j\in F}\sum_{k\in C}\frac{1}{\alpha\rho_j}\|\boldsymbol{A}_{[i,j]}\|\|\boldsymbol{M}_{[k,j]}\|\right)\right) \\ &\succcurlyeq \boldsymbol{I}_r\left(\frac{1}{2}\alpha\rho_i - \frac{1}{2}\left(\sum_{j\in F}\|\boldsymbol{A}_{[i,j]}\|\right)\right) = \boldsymbol{I}_r\left(\frac{1}{2}\left(1-\frac{1}{\alpha}\right)\alpha\rho_i.\right) \end{aligned}$$

So the block rows with $i \in F$ are bDD.

Next we check the block rows for $i \in C$. Let us define for each $i \in C$, $\rho_i = \sum_{j \in F} \|\boldsymbol{M}_{[i,j]}\|$. Thus for $\boldsymbol{Y}$, the sum of block norms of row i over columns $j \in F$

$$\sum_{j \in F} \|\boldsymbol{Y}_{[i,j]}\| = \frac{1}{2}\left(1 - \frac{1}{\alpha}\right) \sum_{j \in F} \|\boldsymbol{M}_{[i,j]}\| = \frac{1}{2}\left(1 - \frac{1}{\alpha}\right) \rho_i.$$

For $\boldsymbol{Y}$, the sum of block norms of row i over columns $j \in C$ is $\sum_{j \in C \setminus \{i\}} \|\boldsymbol{M}_{[i,j]}\|$. So the total sum of the blocks norms of off-diagonals is

$$\frac{1}{2}\rho_i \left(1 - \frac{1}{\alpha}\right) + \sum_{j \in C \setminus \{i\}} \|\boldsymbol{M}_{[i,j]}\|$$

The diagonal block satisfies

$$\begin{aligned}
\boldsymbol{Y}_{[i,i]} &\succcurlyeq \boldsymbol{I}_r \left(\rho_i - \frac{1}{2} \|\boldsymbol{M}_{[i,F]} \boldsymbol{X}^{-1} \boldsymbol{M}_{[F,i]}\| \right. \\
&\quad - \frac{1}{2} \left(\sum_{j \in F} \sum_{k \in C \setminus \{i\}} \|\boldsymbol{M}_{[i,j]} \boldsymbol{X}_{[j,j]}^{-1/2}\| \|\boldsymbol{M}_{[k,j]} \boldsymbol{X}_{[j,j]}^{-1/2}\| \right) \\
&\quad \left. - \frac{1}{2} \left(\sum_{j \in F} \sum_{k \in F} \|\boldsymbol{M}_{[i,j]} \boldsymbol{X}_{[j,j]}^{-1/2}\| \|\boldsymbol{A}_{[k,j]} \boldsymbol{X}_{[j,j]}^{-1/2}\| \right) \right) \\
&\succcurlyeq \boldsymbol{I}_r \left(\rho_i + \sum_{j \in C \setminus \{i\}} \|\boldsymbol{M}_{[i,j]}\| \right. \\
&\quad - \frac{1}{2} \left(\sum_{j \in F} \sum_{k \in C} \frac{1}{\alpha \rho_j} \|\boldsymbol{M}_{[i,j]}\| \|\boldsymbol{M}_{[k,j]}\| \right) \\
&\quad \left. - \frac{1}{2} \left(\sum_{j \in F} \sum_{k \in F} \frac{1}{\alpha \rho_j} \|\boldsymbol{M}_{[i,j]}\| \|\boldsymbol{A}_{[k,j]}\| \right) \right) \\
&\succcurlyeq \boldsymbol{I}_r \left(\rho_i + \sum_{j \in C \setminus \{i\}} \|\boldsymbol{M}_{[i,j]}\| - \frac{1}{2}\rho_i - \frac{1}{2}\frac{1}{\alpha}\rho_i \right) \\
&= \boldsymbol{I}_r \left(\frac{1}{2}\rho_i \left(1 - \frac{1}{\alpha}\right) + \sum_{j \in C \setminus \{i\}} \|\boldsymbol{M}_{[i,j]}\| \right).
\end{aligned}$$

For these block rows are also bDD, and hence $\boldsymbol{Y}$ is bDD. It is clear from the definitions that $\boldsymbol{Y}$ as well as $\boldsymbol{d}^{(i)}$ and $\boldsymbol{g}^{(i)}$ for all i can be computed in $O(m)$ time and $O(\log n)$ depth.

It is easy to verify that if $\boldsymbol{T}$ is a matrix that is only nonzero inside the submatrix $\boldsymbol{T}_{[C,C]}$, then if we apply the transformation of Eq. 13.24, to $\boldsymbol{M}+\boldsymbol{T}$ instead of $\boldsymbol{M}$, we find $(\boldsymbol{M}+\boldsymbol{T})_2^{(last)} = \boldsymbol{M}_2^{(last)} + \boldsymbol{T}_{[C,C]}$, and in particular $\boldsymbol{Y}(\boldsymbol{M}+\boldsymbol{T}) = \boldsymbol{Y}(\boldsymbol{M}) + \boldsymbol{T}_{[C,C]}$. □

■ 13.11.4 Sparsifying Product Demand Block-Laplacians

In this section, we show how to efficiently sparsify product demand block-Laplacians and their bipartite analogs. We prove the following key lemma later in this section that allows us to transfer results on

graph sparsification to sparsifying these product block-Laplacians.

Lemma 13.11.12. *Suppose we have two graphs on n vertices, $H^{(1)}$ and $H^{(2)}$ such that $H^{(1)} \approx_\varepsilon H^{(2)}$. Given $\boldsymbol{d} \in (\mathbb{C}^{r\times r})^n$, define the bDD matrix $\boldsymbol{L}^{(\ell)} \in (\mathbb{C}^{r\times r})^{n\times n}$ for each $\ell = 1, 2$, as*

$$\boldsymbol{L}^{(\ell)}{}_{[i,j]} = \begin{cases} -\frac{h^{(\ell)}_{i,j}}{\|\boldsymbol{d}_{[i]}\|\|\boldsymbol{d}_{[j]}\|}\boldsymbol{d}_{[i]}\boldsymbol{d}^*_{[j]} & i \neq j, \\ \boldsymbol{I}_r \cdot \sum_{k:k\neq i} h^{(\ell)}_{i,k} & \text{otherwise,} \end{cases}$$

where $h^{(\ell)}_{i,j}$ denotes the weight of the edge i, j in $H^{(\ell)}$. Then, $\boldsymbol{L}^{(1)} \approx_\varepsilon \boldsymbol{L}^{(2)}$.

We now introduce scalar versions of product block-Laplacian matrices that will be useful.

Definition 13.11.13. The product demand graph of a vector $\boldsymbol{d} \in (\mathbb{R}_{>0})^n$, $G(\boldsymbol{d})$, is a complete weighted graph on n vertices whose weight between vertices i and j is given by $\boldsymbol{w}_{ij} = \boldsymbol{d}_i\boldsymbol{d}_j$.

The Laplacian of $G(\boldsymbol{d})$, denoted $\boldsymbol{L}_{G(\boldsymbol{d})}$ is called the product demand Laplacian of $\boldsymbol{d}$.

Definition 13.11.14. The bipartite product demand graph of two vectors $\boldsymbol{d}^A \in (\mathbb{R}_{>0})^{|A|}$, $\boldsymbol{d}^B \in (\mathbb{R}_{>0})^{|B|}$, $G(\boldsymbol{d}^A, \boldsymbol{d}^B)$, is a weighted bipartite graph on vertices $A \cup B$, whose weight between vertices $i \in A$ and $j \in B$ is given by $\boldsymbol{w}_{ij} = \boldsymbol{d}^A_i\boldsymbol{d}^B_j$.

The Laplacian of $G(\boldsymbol{d}^A, \boldsymbol{d}^B)$, denoted $\boldsymbol{L}_{G(\boldsymbol{d}^A,\boldsymbol{d}^B)}$ is called the bipartite product demand Laplacian of $(\boldsymbol{d}^A, \boldsymbol{d}^B)$.

In Section 13.14, we give results on efficiently constructing approximations to product demand Laplacians that can be summarized as follows:

Lemma 13.11.1. There is a routine WeightedExpander$(\boldsymbol{d}, \varepsilon)$ such that for any $\varepsilon > 0$, and a demand vector $\boldsymbol{d} \in (\Re_{>0})^n$, WeightedExpander$(\boldsymbol{d}, \varepsilon)$ returns in $O(n\varepsilon^{-4})$ work and $O(\log n)$ depth a graph H with $O(n\varepsilon^{-4})$ edges such that

$$\boldsymbol{L}_H \approx_\varepsilon \boldsymbol{L}_{G(\boldsymbol{d})}.$$

Lemma 13.11.2. There is a routine WeightedBipartiteExpander$(\boldsymbol{d}^A, \boldsymbol{d}^B, \varepsilon)$ such that for any demand vectors $\boldsymbol{d}^A$ and $\boldsymbol{d}^B$ of total length n, and a parameter ε, it returns in $O(n\varepsilon^{-4})$ work and $O(\log n)$ depth a bipartite graph H between A and B with $O(n\varepsilon^{-4})$ edges such that

$$\boldsymbol{L}_H \approx_\varepsilon \boldsymbol{L}_{G(\boldsymbol{d}^A,\boldsymbol{d}^B)}.$$

Finally, we need to define an operation on graphs: Given a graph G, define $G^{(K2)}$ to be the graph obtained by duplicating each vertex in G, and for each edge (i, j) in G, add a 2×2 bipartite clique between the two copies of i and j.

We now combine the above construction of sparsifiers fo product demand graphs with Lemma 13.11.12 to efficiently construct sparse approximations to product demand block-Laplacians.

Proof. (of Lemma 13.11.6) The procedure CliqueSparsification$(\boldsymbol{d}, \varepsilon)$ returns the matrix $\boldsymbol{L}$ where

$$\boldsymbol{L}_{[i,j]} = \begin{cases} -\frac{h_{i,j}}{\|\boldsymbol{d}_{[i]}\|\|\boldsymbol{d}_{[j]}\|}\boldsymbol{d}_{[i]}\boldsymbol{d}^*_{[j]} & i \neq j, \\ \boldsymbol{I}_r \cdot \sum_{k:k\neq i} h_{i,k} & \text{otherwise.} \end{cases}$$

By construction $\boldsymbol{L}_H \approx_\varepsilon \boldsymbol{L}_{G(\boldsymbol{w})}$. Thus, applying Lemma 13.11.12, with $H^{(1)} = H$ and $H^{(2)} = G(\boldsymbol{w})$, we know that $\boldsymbol{L} \approx_\varepsilon \boldsymbol{L}^{(2)}$, where $\boldsymbol{L}^{(2)}$ is given by

$$\boldsymbol{L}^{(2)}{}_{[i,j]} = \begin{cases} -\frac{\boldsymbol{w}_i\boldsymbol{w}_j}{\|\boldsymbol{d}_{[i]}\|\|\boldsymbol{d}_{[j]}\|}\boldsymbol{d}_{[i]}\boldsymbol{d}^*_{[j]} & i \neq j, \\ \boldsymbol{I}_r \cdot \boldsymbol{w}_i \sum_{k:k\neq i} \boldsymbol{w}_k & \text{otherwise.} \end{cases}$$

Algorithm 42: $\boldsymbol{L} = \textsc{CliqueSparsification}(\boldsymbol{d}, \varepsilon)$

Initialize $\boldsymbol{L} \in (\mathbb{C}^{r\times r})^{n\times n}$ to 0.
Construct $\boldsymbol{w}$ where $\boldsymbol{w}_i = \|\boldsymbol{d}_{[i]}\|$.
$H \leftarrow \textsc{WeightedExpander}(\boldsymbol{w}, \varepsilon)$.
for *edge* $(i,j) \in H$, *with weight* $h_{i,j}$, **do**
  $\boldsymbol{L}_{[i,i]} \leftarrow \boldsymbol{L}_{[i,i]} + h_{i,j}\boldsymbol{I}_r$.
  $\boldsymbol{L}_{[j,j]} \leftarrow \boldsymbol{L}_{[j,j]} + h_{i,j}\boldsymbol{I}_r$.
  $\boldsymbol{L}_{[i,j]} \leftarrow \boldsymbol{L}_{[i,j]} - \frac{h_{i,j}}{\boldsymbol{w}_i\boldsymbol{w}_j}\boldsymbol{d}_{[i]}\boldsymbol{d}^*_{[j]}$.
  $\boldsymbol{L}_{[j,i]} \leftarrow \boldsymbol{L}_{[j,i]} - \frac{h_{i,j}}{\boldsymbol{w}_i\boldsymbol{w}_j}\boldsymbol{d}_{[j]}\boldsymbol{d}^*_{[i]}$.
end
Output $\boldsymbol{L}$.

Figure 13-8: Pseudocode for Sparsifying Product Demand Block-Laplacians

Algorithm 43: $\boldsymbol{L} = \textsc{BipartiteCliqueSparsification}(\boldsymbol{b}, F, \varepsilon)$

Initialize $\boldsymbol{L} \in (\mathbb{C}^{r\times r})^{n\times n}$ to 0.
Construct $\boldsymbol{w}$ where $\boldsymbol{w}_i = \|\boldsymbol{d}_{[i]}\|$. Let $C = V \setminus F$.
$H \leftarrow \textsc{WeightedBipartiteExpander}(\boldsymbol{w}|_F, \boldsymbol{w}|_C, \varepsilon)$.
for *edge* $(i,j) \in H$, *with weight* $h_{i,j}$, **do**
  $\boldsymbol{L}_{[i,i]} \leftarrow \boldsymbol{L}_{[i,i]} + h_{i,j}\boldsymbol{I}_r$.
  $\boldsymbol{L}_{[j,j]} \leftarrow \boldsymbol{L}_{[j,j]} + h_{i,j}\boldsymbol{I}_r$.
  $\boldsymbol{L}_{[i,j]} \leftarrow \boldsymbol{L}_{[i,j]} - \frac{h_{i,j}}{\boldsymbol{w}_i\boldsymbol{w}_j}\boldsymbol{d}_{[i]}\boldsymbol{d}^*_{[j]}$.
  $\boldsymbol{L}_{[j,i]} \leftarrow \boldsymbol{L}_{[j,i]} - \frac{h_{i,j}}{\boldsymbol{w}_i\boldsymbol{w}_j}\boldsymbol{d}_{[j]}\boldsymbol{d}^*_{[i]}$.
end
Output $\boldsymbol{L}$.

Figure 13-9: Pseudocode for Sparsifying Bipartite Product Demand Block-Laplacians

Since $\boldsymbol{w}_i = \|\boldsymbol{d}_{[i]}\|$, we have $\boldsymbol{L}^{(2)} = \boldsymbol{L}_{G(\boldsymbol{d})}$, proving our claim. □

Proof. (of Lemma 13.11.7) The procedure $\textsc{BipartiteCliqueSparsification}(\boldsymbol{d}, F, \varepsilon)$ returns the matrix $\boldsymbol{L}$ where

$$\boldsymbol{L}_{[i,j]} = \begin{cases} -\frac{h_{i,j}}{\|\boldsymbol{d}_{[i]}\|\|\boldsymbol{d}_{[j]}\|}\boldsymbol{d}_{[i]}\boldsymbol{d}^*_{[j]} & i \neq j, \\ \boldsymbol{I}_r \cdot \sum_{k:k\neq i} h_{i,k} & \text{otherwise.} \end{cases}$$

Since H returned by $\textsc{WeightedBipartiteExpander}$ is guaranteed to be bipartite using Lemma 13.11.2, we obtain that $(\boldsymbol{L}_H)_{[F,F]}$, and $(\boldsymbol{L}_H)_{[C,C]}$ are block-diagonal.

By construction $\boldsymbol{L}_H \approx_\varepsilon \boldsymbol{L}_{G(\boldsymbol{w}|_F, \boldsymbol{w}|_C)}$. Thus, applying Lemma 13.11.12, with $H^{(1)} = H$ and $H^{(2)} = G(\boldsymbol{w}|_F, \boldsymbol{w}|_C)$, we know that $\boldsymbol{L} \approx_\varepsilon \boldsymbol{L}^{(2)}$, where $\boldsymbol{L}^{(2)}$ is given by

$$\boldsymbol{L}^{(2)}{}_{[i,j]} = \begin{cases} \boldsymbol{I}_r \cdot \boldsymbol{w}_i \sum_{k\in C} \boldsymbol{w}_k & i = j, \text{ and } i \in F, \\ \boldsymbol{I}_r \cdot \boldsymbol{w}_i \sum_{k\in F} \boldsymbol{w}_k & i = j, \text{ and } i \in C, \\ 0 & i \neq j, \text{ and } (i,j \in F \text{ or } i,j \in C), \\ -\frac{\boldsymbol{w}_i\boldsymbol{w}_j}{\|\boldsymbol{d}_{[i]}\|\|\boldsymbol{d}_{[j]}\|}\boldsymbol{d}_{[i]}\boldsymbol{d}^*_{[j]} & i \neq j, \text{otherwise.} \end{cases}$$

Since $\boldsymbol{w}_i = \|\boldsymbol{d}_{[i]}\|$, we have $\boldsymbol{L}^{(2)} = \boldsymbol{L}_{G(\boldsymbol{d}|_F, \boldsymbol{d}|_C)}$, proving our claim. □

Finally, we give a proof of Lemma 13.11.12.

Proof. (of Lemma 13.11.12) Using Fact 13.5.2, we can write each $\boldsymbol{d}_{[i]}$ as $\frac{1}{2}\|\boldsymbol{d}_{[i]}\|(\boldsymbol{Q}_i^{(1)} + \boldsymbol{Q}_i^{(2)})$, where $\boldsymbol{Q}_i^{(1)}(\boldsymbol{Q}_i^{(1)})^* = \boldsymbol{Q}_i^{(2)}(\boldsymbol{Q}_i^{(2)})^* = \boldsymbol{I}_r$. Construct the matrix $\boldsymbol{F} \in (\mathbb{C}^{r\times r})^{2n\times 2n}$ as follows:

$$\boldsymbol{F} = \begin{bmatrix} \boldsymbol{Q}_1^{(1)} & \boldsymbol{Q}_1^{(2)} & 0 & 0 & \dots \\ 0 & 0 & \boldsymbol{Q}_2^{(1)} & \boldsymbol{Q}_2^{(2)} & \dots \\ \vdots & \vdots & & & \ddots \end{bmatrix}.$$

Now, observe that for each $\ell = 1, 2$,

$$\boldsymbol{L}^{(\ell)}{}_{[i,i]} = \frac{1}{2}\left(\sum_{k:k\neq i} h_{i,k}^{(\ell)}\right) \cdot \left(\boldsymbol{Q}_i^{(1)}(\boldsymbol{Q}_i^{(1)})^* + \boldsymbol{Q}_i^{(2)}(\boldsymbol{Q}_i^{(2)})^*\right) = \frac{1}{4}\left(\boldsymbol{F}(\boldsymbol{L}_{(H^{(\ell)})^{(K2)}} \otimes \boldsymbol{I}_r)\boldsymbol{F}^*\right)_{[i,i]},$$

where $\boldsymbol{L}_{(H^{(\ell)})^{(K2)}}$ is the Laplacian of the graph $(H^{(\ell)})^{(K2)}$ defined above, and $\otimes$ tensor product. Also, for $i \neq j$,

$$\boldsymbol{L}^{(\ell)}{}_{[i,j]} = -\frac{h_{i,j}^{(\ell)}}{\boldsymbol{w}_i\boldsymbol{w}_j}\boldsymbol{d}_{[i]}\boldsymbol{d}_{[j]}^* = -\frac{h_{i,j}^{(\ell)}}{4} \cdot \left(\boldsymbol{Q}_i^{(1)} + \boldsymbol{Q}_i^{(2)}\right)\left(\boldsymbol{Q}_j^{(1)} + \boldsymbol{Q}_j^{(2)}\right)^* = \frac{1}{4}\left(\boldsymbol{F}(\boldsymbol{L}_{(H^{(\ell)})^{(K2)}} \otimes \boldsymbol{I}_r)\boldsymbol{F}^*\right)_{[i,j]},$$

Thus,

$$\boldsymbol{L}^{(\ell)} = \frac{1}{4}\boldsymbol{F}(\boldsymbol{L}_{(H^{(\ell)})^{(K2)}} \otimes \boldsymbol{I}_r)\boldsymbol{F}^*.$$

By assumption, we have $\boldsymbol{L}_{H^{(1)}} \approx_\varepsilon \boldsymbol{L}_{H^{(2)}}$. Using Lemma 13.11.15, we get that $\boldsymbol{L}_{H^{(1)(K2)}} \approx_\varepsilon L_{H^{(2)(K2)}}$. This implies

$$\boldsymbol{L}_{(H^{(1)})^{(K2)}} \otimes \boldsymbol{I}_r \approx_\varepsilon \boldsymbol{L}_{(H^{(2)})^{(K2)}} \otimes \boldsymbol{I}_r,$$

and thus, $\boldsymbol{L}^{(1)} \approx_\varepsilon \boldsymbol{L}^{(2)}$. □

13.11.5 Constructing sparsifiers for lifts of graphs

Given a graph G, define $G^{(K2)}$ to be the graph obtained by duplicating each vertex in G, and for each edge (i, j) in G, add a 2×2 bipartite clique between the two copies of i and j.

Lemma 13.11.15. *If H is a sparsifier for G, i.e., $L_G \approx_\varepsilon L_H$, then $H^{(K2)}$ is a sparsifier for $G^{(K2)}$, i.e. $L_{G^{(K2)}} \approx_\varepsilon L_{H^{(K2)}}$.*

Proof. Since $L_G \approx_\varepsilon L_H$, we have $e_i^\top L_G e_i \approx_\varepsilon e_i^\top L_H e_i$. Thus, if D_G denotes the diagonal matrix of degrees of G, we have $D_G \approx_\varepsilon D_H$. The Laplacian for $G^{(K2)}$ is

$$L_{G^{(K2)}} = L_G \otimes \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} + D_G \otimes \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}.$$

Similarly,

$$L_{H^{(K2)}} = L_H \otimes \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} + D_H \otimes \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}.$$

Observe that we can write

$$L_G \otimes \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 & \dots \\ 0 & 0 & 1 & 1 & \dots \\ \vdots & \vdots & & \ddots & \end{bmatrix} L_G \begin{bmatrix} 1 & 1 & 0 & 0 & \dots \\ 0 & 0 & 1 & 1 & \dots \\ \vdots & \vdots & & \ddots & \end{bmatrix}^\top.$$

Since $L_G \approx_\varepsilon L_H$, this implies

$$L_G \otimes \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \approx_\varepsilon L_H \otimes \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}.$$

Similarly, we get,

$$D_G \otimes \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \approx_\varepsilon D_H \otimes \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}.$$

Adding the above two, we get $L_{G^{(K2)}} \approx_\varepsilon L_{H^{(K2)}}$. □

13.12 Estimating Leverage Scores by Undersampling

We will control the densities of all intermediate bDD matrices using the uniform sampling technique introduced in Chapter 3. It relies on sampling columns [2] of matrices by upper bounds of their true leverage scores,

$$\boldsymbol{\tau}_i(\boldsymbol{A}) = \boldsymbol{a}_i^T (\boldsymbol{A}\boldsymbol{A}^*)^{-1} \boldsymbol{a}_i.$$

These upper bounds are measured w.r.t. a different matrix, giving generalized leverage scores of the form:

$$\tau_i^{\boldsymbol{B}}(\boldsymbol{A}) = \begin{cases} \boldsymbol{a}_i^*(\boldsymbol{B}\boldsymbol{B}^*)^{-1}\boldsymbol{a}_i & \text{if } \boldsymbol{a}_i \perp \ker(\boldsymbol{B}), \\ 1 & \text{otherwise.} \end{cases}$$

We introduced *unitary edge-vertex transfer matrices* in Definition 13.6.4. Sparsifying bDD matrices can be transformed into the more general setting described above via a unitary edge-vertex transfer matrix, which is analogous to the edge-vertex incidence matrix.

Lemma 13.6.5 proves that every bDD matrix $\boldsymbol{M} \in (\mathbb{C}^{r\times r})^{n\times n}$ with m nonzero off-diagonal blocks can be written as $\boldsymbol{X} + \boldsymbol{B}\boldsymbol{B}^*$ where $\boldsymbol{B} \in (\mathbb{C}^{r\times r})^{n\times 2m}$ is a unitary edge-vertex transfer matrix and $\boldsymbol{X}$ is a block diagonal PSD matrix. Additionally, for every block diagonal matrix $\boldsymbol{Y}$ s.t. $\boldsymbol{M} - \boldsymbol{Y}$ is bDD, we have $\boldsymbol{X} \succcurlyeq \boldsymbol{Y}$. This decomposition can be found in $O(m)$ time and $O(\log n)$ depth. We rely on this $\boldsymbol{X}$ to detect some cases of high leverage scores as samples of $\boldsymbol{B}$ may have lower rank.

We will reduce the number of nonzero blocks in $\boldsymbol{M}$ by sampling columns blocks from this matrix. This is more restrictive than sampling individual columns. Nonetheless, it can be checked via matrix concentration bounds [3, 251] that it suffices to sample the block by analogs of leverage scores:

$$\boldsymbol{\tau}_{[i]} = \operatorname{tr}\left(\boldsymbol{B}_{[i]}^*(\boldsymbol{X} + \boldsymbol{B}\boldsymbol{B}^*)^{-1}\boldsymbol{B}_{[i]}\right) \tag{13.36}$$

As in Chapter 3, we recursively estimate upper bounds for these scores, leading to the pseudocode given in Figure 13-10.

Lemma 13.12.1. *Given a positive definite bDD matrix $\boldsymbol{M} \in (\mathbb{C}^{r\times r})^{n\times n}$ with m nonzero blocks.*

Assume that for any positive definite bDD matrix $\boldsymbol{M}' \in (\mathbb{C}^{r\times r})^{n\times n}$ with $\widehat{m}$ nonzero blocks, we can find an implicit representation of a matrix $\boldsymbol{W}$ such that $\boldsymbol{W} \approx_1 (\boldsymbol{M}')^{-1}$ in depth $d_{con}(\widehat{m}, n)$ and work $w_{con}(\widehat{m}, n)$ and for any vector b, we can evaluate $\boldsymbol{W}\boldsymbol{b}$ in depth $d_{eval}(\widehat{m}, n)$ and work $w_{eval}(\widehat{m}, n)$.

[2] In Chapter 3, we sample rows of matrices. Here, we sample columns instead in order to use a more natural set of notations.

Algorithm 44: $\widetilde{\boldsymbol{M}} = \textsc{Sparsify}(\boldsymbol{M}, \varepsilon, K)$

Write $\boldsymbol{M}$ into the form $\boldsymbol{X} + \boldsymbol{B}\boldsymbol{B}^*$ where $\boldsymbol{X}$ is a block diagonal positive definite matrix $\boldsymbol{X}$ and $\boldsymbol{B}$ is an unitary edge-vertex transfer matrix.
Uniformly sample $\frac{m}{K}$ column blocks of $\boldsymbol{B}$ to form $\boldsymbol{C}$.
Find an implicit representation of $(\boldsymbol{X} + \boldsymbol{C}\boldsymbol{C}^*)^{-1}$, $\boldsymbol{W}$.
Approximate $\operatorname{tr}\left(\boldsymbol{B}_{[i]}^*(\boldsymbol{X} + \boldsymbol{C}\boldsymbol{C}^*)^{-1}\boldsymbol{B}_{[i]}\right)$ using Johnson-Lindenstrauss projections.
This involves applying $\boldsymbol{W}$ to $O(\log n)$ random vectors.
Use these estimates to sample blocks of $\boldsymbol{B}$ to form $\tilde{\boldsymbol{B}}$.
Return $\boldsymbol{X} + \tilde{\boldsymbol{B}}\tilde{\boldsymbol{B}}^*$.

Figure 13-10: Sparsification via. Uniform Sampling

For any $K \geq 1$, $1 \geq \varepsilon > 0$, the algorithm $\textsc{Sparsify}(\boldsymbol{M}, \varepsilon, K)$ outputs an explicit positive definite bDD matrix $\widetilde{\boldsymbol{M}}$ with $O(Kn \log n/\varepsilon^2)$ nonzero blocks and $\widetilde{\boldsymbol{M}} \approx_\varepsilon \boldsymbol{M}$.

Also, this algorithm runs in

$$d_{con}\left(\frac{m}{K}, n\right) + O\left(d_{eval}\left(\frac{m}{K}, n\right) + \log n\right)$$

depth and

$$w_{con}\left(\frac{m}{K}, n\right) + O\left(w_{eval}\left(\frac{m}{K}, n\right)\log n + m \log n\right)$$

work.

The guarantees of this process can be obtained from the main result from Chapter 3:

Theorem 13.12.2. *Let $\boldsymbol{A}$ be a n by m matrix, and $\alpha \in (0, 1]$ a density parameter. Consider the matrix $\boldsymbol{A}'$ consisting of a random α fraction of the columns of $\boldsymbol{A}$. Then, with high probability we have $\boldsymbol{u}_i = \tau_i^{\boldsymbol{A}'}(\boldsymbol{A})$ is a vector of leverage score overestimates, i.e. $\tau_i(\boldsymbol{A}) \leq \boldsymbol{u}_i$, and*

$$\sum_{i=1}^{m} \boldsymbol{u}_i \leq \frac{3n}{\alpha}.$$

Proof. (Sketch of Lemma 13.12.1) Instead of sampling m/K column blocks, consider sampling m/K individual columns to form $\widehat{\boldsymbol{C}}$. Theorem 13.12.2 gives that the total leverage scores of all the columns of $\boldsymbol{B}$ w.r.t. $\widehat{\boldsymbol{C}}$ is at most $O(nKr^2)$.

As $\boldsymbol{C}$ is formed by taking blocks instead of columns, we have

$$\boldsymbol{X} + \boldsymbol{C}\boldsymbol{C}^* \succcurlyeq \boldsymbol{X} + \widehat{\boldsymbol{C}}\widehat{\boldsymbol{C}}^*,$$

so the individual leverage scores computed w.r.t. $\boldsymbol{C}$ (and in turn $\boldsymbol{W}$) sums up to less than the ones computed w.r.t. $\widehat{\boldsymbol{C}}$.

Let us use $\boldsymbol{b}_{[i],j} \in \mathbb{C}^n$ to denote the j^{th} *column* of the $\boldsymbol{B}_{[i]}$ block column. Note that

$$\boldsymbol{B}_{[i]}\boldsymbol{B}_{[i]}^* = \sum_j \boldsymbol{b}_{[i],j}\boldsymbol{b}_{[i],j}^*.$$

Define $\tau_{[i],j}^{\boldsymbol{X}+\boldsymbol{C}\boldsymbol{C}^*} = \operatorname{tr}\left(\boldsymbol{b}_{[i],j}^*(\boldsymbol{X} + \boldsymbol{C}\boldsymbol{C}^*)^{-1}\boldsymbol{b}_{[i],j}\right)$.

We then have

$$\operatorname{tr}\left(\boldsymbol{B}_{[i]}^*(\boldsymbol{X} + \boldsymbol{C}\boldsymbol{C}^*)^{-1}\boldsymbol{B}_{[i]}\right) = \sum_j \tau_{[i],j}^{\boldsymbol{X}+\boldsymbol{C}\boldsymbol{C}^*},$$

so the total sum of the estimates we obtain is at most $O(nr^2K)$, which gives $\tilde{\boldsymbol{B}}$ with $O(Kn\log n\varepsilon^{-2})$ blocks.

To approximate $\sum_j \tau_{[i],j}^{\boldsymbol{X}+\boldsymbol{C}\boldsymbol{C}^*}$, we apply a standard technique given by [242, 23, 170]. The rough idea is to write

$$\tau_{[i],j}^{\boldsymbol{X}+\boldsymbol{C}\boldsymbol{C}^*} = \|\boldsymbol{C}^*(\boldsymbol{X}+\boldsymbol{C}\boldsymbol{C}^*)^{-1}\boldsymbol{b}_{[i],j}\|^2 + \|\sqrt{\boldsymbol{X}}(\boldsymbol{X}+\boldsymbol{C}\boldsymbol{C}^*)^{-1}\boldsymbol{b}_{[i],j}\|^2$$

By Johnson-Lindenstrauss Lemma, for a random Gaussian matrix $\boldsymbol{G}$ with $\Theta(\log(n))$ rows, we know that

$$\tau_{[i],j}^{\boldsymbol{X}+\boldsymbol{C}\boldsymbol{C}^*} \approx_2 \|\boldsymbol{G}\boldsymbol{C}^*(\boldsymbol{X}+\boldsymbol{C}\boldsymbol{C}^*)^{-1}\boldsymbol{b}_{[i],j}\|^2 + \|\boldsymbol{G}\sqrt{\boldsymbol{X}}(\boldsymbol{X}+\boldsymbol{C}\boldsymbol{C}^*)^{-1}\boldsymbol{b}_{[i],j}\|^2$$

for high probability. Since $\boldsymbol{G}$ has $O(\log(n))$ rows, this can be approximated by applying the approximate inverse $\boldsymbol{W}$ to $O(\log(n))$ vectors. Hence, step 3 runs in $O(d_{eval}(\frac{m}{K}, n) + \log n)$ depth and $O(w_{eval}(\frac{m}{K}, n)\log n + m\log n)$ work. □

We remark that the extra factor of r^2 can likely be improved by modifying the proof of Theorem 13.12.2 to work with blocks. We omit this improvement for simplicity, especially since we're already treating r as a constant.

13.13 Recursive Construction of Schur Complement Chains

We now give the full details for the recursive algorithm that proves the running times as stated in Theorem 13.1.2:

Theorem 13.1.2 (Sparsified Multigrid)**.** *There is an algorithm that, when given a bDD matrix $\boldsymbol{M}$ with n block rows and m nonzero blocks, produces a solver for $\boldsymbol{M}$ in $O(m\log n + n\log^{2+o(1)} n)$ work and $O(n^{o(1)})$ depth so that the solver finds ε-approximate solutions to systems of equations in $\boldsymbol{M}$ in $O((m + n\log^{1+o(1)} n)\log(1/\varepsilon))$ work and $O(\log^2 n \log\log n \log(1/\varepsilon))$ depth.*

This argument can be viewed as a more sophisticated version of the one in Section 13.3.6. The main idea is to invoke routines for reducing edges and vertices in a slightly unbalanced recursion, where several steps of vertex reductions take place before a single edge reduction. The routines that we will call are:

1. bDDSubset given by Lemma 13.3.2 proven in Section 13.8 for finding a large set of α-bDD subset.

2. ApproxSchur given by Lemma 13.11.1 proven in Section 13.11 that gives sparse approximations to Schur complements.

3. Sparsify given by Lemma 13.12.1 proven in Section 13.12 that allows us to sparsify a bDD matrix by recursing on a uniform subsample of its non-zero blocks.

An additional level of complication comes from the fact that the approximation guarantees of our constructions rely on gradually decreasing errors down the Schur complement chain. This means that the density increases faster and larger reduction factors K are required as the iteration goes on. Pseudocode of our algorithm is given in Figure 13-11.

Note that since $\boldsymbol{M}^{(0)}$ may be initially dense, we first make a recursive call to Sparsify before computing the approximate Schur complements.

In our analysis, we will use $n^{(i)}$ to denote the number of non-zero column/row blocks in $\boldsymbol{M}^{(i)}$, and $m^{(i)}$ to denote the number of non-zero blocks in $\boldsymbol{M}^{(i)}$. These are analogous to dimension and

Algorithm 45: $(\boldsymbol{M}^{(1)}, \boldsymbol{M}^{(2)}, \cdots; F_1, F_2, \cdots) = \text{RecursiveConstruct}(\boldsymbol{M}^{(0)})$

Set $i \leftarrow 0$, and k and c are parameters to be decided.
while $\boldsymbol{M}^{(i)}$ *has more than* $\Theta(1)$ *blocks* **do**
  If $i \bmod k = 0$, Then$\boldsymbol{M}^{(i)} \leftarrow \text{Sparsify}(\boldsymbol{M}^{(i)}, (i+8)^{-2}, 2^{2ck\log^2(i+1)})$.
  $F_{i+1} \leftarrow \text{bDDSubset}\left(\boldsymbol{M}^{(i)}, 4\right)$.
  $\boldsymbol{M}^{(i+1)} \leftarrow \text{ApproxSchur}(\boldsymbol{M}^{(i)}, F_{i+1}, 4, (i+8)^{-2})$.
  $i \leftarrow i+1$.
end

Figure 13-11: Pseudocode for Recursively Constructing Schur Complement Chains, the recursive calls happen through Sparsify, which constructs its own Schur Complement Chains to perform the sparsification

number of non-zeros in the matrix. It is also useful to refer to the steps between calls to Sparsify as phases. Specifically, phase j consists of iterations $(j-1)k$ to $jk-1$. We will also use K_j to denote the reduction factor used to perform the sparsification at the start of phase j, aka.

$$K_j = 2^{2ck\log^2((j-1)k+1)}.$$

A further technicality is that Lemma 13.12.1 require a strictly positive definite block-diagonal part to facilitate the detection of vectors in the null space of the sample. We do so by padding $\boldsymbol{M}$ with a small copy of the identity, and will check below that this copy stays throughout the course of this algorithm.

Lemma 13.13.1. *For any bDD matrix $\boldsymbol{M}^{(0)}$ expressible as $\xi\boldsymbol{I} + \boldsymbol{B}^{(0)}(\boldsymbol{B}^{(0)})^*$ for some $\xi > 0$ and unitary edge-vertex transfer matrix $\boldsymbol{B}$, all intermediate matrices $\boldsymbol{M}'$ generated during the algorithm* $\text{RecursiveConstruct}(\boldsymbol{M}^{(0)})$ *can be expressed as $\xi\boldsymbol{I} + \boldsymbol{B}'(\boldsymbol{B}')^*$.*

Proof. There are two mechanisms by which this recursive algorithm generates new matrices: through uniform sampling within Sparsify and via ApproxSchur. We will show inductively down the algorithmic calls that all matrices satisfy this property.

Lemma 13.12.1 gives that this $\boldsymbol{X}$ is preserved in the sample.

For calls to $\text{ApproxSchur}(\boldsymbol{M}, C)$, the inductive hypothesis gives that $\boldsymbol{M}_{[C,C]}$ is expressible as $\widehat{\boldsymbol{M}} + \xi\boldsymbol{I}_C$. The last condition in Lemma 13.3.4 then gives that the result also has a $\xi\boldsymbol{I}_C$ part, which gives the inductive hypothesis for it as well. □

As the interaction between this padding and our recursion is minimal, we will simply state the input conditions as $\boldsymbol{M} = \xi\boldsymbol{I} + \boldsymbol{B}\boldsymbol{B}^*$ for some $\xi > 0$. We will first bound the sizes of the matrices in the Schur complement chain produced by RecursiveConstruct.

Lemma 13.13.2. *For any bDD matrix $\boldsymbol{M}^{(0)} = \xi\boldsymbol{I} + \boldsymbol{B}^{(0)}(\boldsymbol{B}^{(0)})^*$ for some $\xi > 0$, the algorithm* $\text{RecursiveConstruct}(\boldsymbol{M}^{(0)})$ *returns a Schur complement chain $(\boldsymbol{M}^{(1)}, \boldsymbol{M}^{(2)}, \cdots; F_1, F_2, \cdots)$ such that:*

1. *$n^{(i)} \leq \beta^i n^{(0)}$ for some absolute constant $\beta < 1$.*
2. *The number of non-zero blocks in any iteration i of phase j is at most $2^{3ck\log^2(jk)} n^{((j-1)k)} \log n^{((j-1)k)}$.*

Proof. Lemma 13.3.2 and the choice of $\alpha = 4$ ensures $|F_k| = \Omega(n^{(i)})$, which means there is constant $\beta < 1$ such that $n^{(i)} \leq \beta^k n$. Furthermore, we do not increase vertex count at any point in this recursion, and all recursive calls are made to smaller graphs. Therefore, the recursion terminates.

Lemma 13.11.1 shows that after computing each approximate Schur complement,

$$\begin{aligned} m^{(i+1)} &= O(m^{(i)}(i^2 \log(i+8))^{O(\log(i+8))}) \\ &\leq 2^{O\left(\log^2(i+1)\right)} m^{(i)}. \end{aligned}$$

Hence, by picking c appropriately we can guarantee that the density increases by a factor of most $2^{c\log^2(i+1)}$ during each iteration. This size increase is controlled by calls to Sparsify. Specifically, Lemma 13.12.1 gives that at the start of phase j we have:

$$\frac{m^{((j-1)k)}}{n^{((j-1)k)} \log n^{((j-1)k)}} \leq 2^{2ck \log^2((j-1)k)}.$$

Then, since we go at most k steps without calling Sparsify, this increase in density can be bounded by:

$$2^{2ck\log^2((j-1)k)} 2^{c\log^2(jk-1)} \cdots 2^{c\log^2((j-1)k+1)} \leq 2^{3ck\log^2(jk)}.$$

□

Lemma 13.13.3. *For any bDD matrix $\boldsymbol{M}^{(0)} = \xi \boldsymbol{I} + \boldsymbol{B}^{(0)}(\boldsymbol{B}^{(0)})^*$ for some $\xi > 0$ that has n blocks, the algorithm* RecursiveConstruct$(\boldsymbol{M}^{(0)})$ *returns a Schur Complement Chain $(\boldsymbol{M}^{(1)}, \cdots; F_1, \cdots)$ whose corresponding the linear operator $\boldsymbol{W}$ satisfies*

$$\boldsymbol{W} \approx_{O(1)} \left(\boldsymbol{M}^{(0)}\right)^{-1}.$$

Also, we can evaluate $\boldsymbol{W}\boldsymbol{b}$ in $O(\log^2 n \log\log n)$ depth and $2^{O(k\log^2 k)} n \log n$ work for any vector $\boldsymbol{b}$.

Proof. We first bound the quality of approximation between $\boldsymbol{W}$ and $\left(\boldsymbol{M}^{(0)}\right)^{-1}$. The approximate Schur complement $\boldsymbol{M}^{(i+1)}$ was constructed so that $\boldsymbol{M}^{(i+1)} \approx_{(i+8)^{-2}} Sc\left(\boldsymbol{M}^{(i)}, F_i\right)$. The other source of error, Sparsify, is called only for some i. In those iterations, Lemma 13.12.1 guarantee that $\boldsymbol{M}^{(i)}$ changes only by $(i+8)^{-2}$ factor. This means overall we have $\boldsymbol{M}^{(i+1)} \approx_{2(i+8)^{-2}} Sc\left(\boldsymbol{M}^{(i)}, F_i\right)$. By Lemma 13.3.1, we have:

$$\boldsymbol{W} \approx_{1/2+4\sum_i (i+8)^{-2}} \left(\boldsymbol{M}^{(0)}\right)^{-1},$$

and it can be checked that $\sum_i (i+8)^{-2}$ is a constant.

The cost of ApplyChain is dominated by the sequence of calls to Jacobi. As each F_i is chosen to be 4-bDD, the number of iterations required is $O(\log(\varepsilon_i)) = O(\log i)$. As matrix-vector multiplications take $O(\log n)$ depth, the total depth can be bounded by.

$$O\left(\sum_i \log i \log n^{(i)}\right) = O\left(\log^2 n \log\log n\right)$$

The total work of these steps depend on the number of non-zero blocks, $m^{(i)}$. Substituting the bounds from Lemma 13.13.2 into Jacobi gives a total of:

$$O\left(\sum_i 2^{3ck\log^2(i+k)} n^{(i)} \log n^{(i)} \log i\right) \leq 2^{O(k\log^2 k)} n \log n$$

where the inequality follows from the fact that the $n^{(i)}$s are geometrically decreasing. □

This allows us to view the additional overhead of Sparsify as a black box, and analyze the total cost incurred by the non-recursive parts of RecursiveConstruct during each phase.

Lemma 13.13.4. *The total cost of* RecursiveConstruct *during phase j, including the linear system solves made by* Sparsify *at iteration $(j-1)k$ (but not its recursive invocation to* RecursiveConstruct*) is*

$$2^{O(k\log^2(jk))}n^{((j-1)k)}\log^2 n^{((j-1)k)}$$

in work and

$$O\left(k\log(jk)\log^2 n^{((j-1)k)}\right)+O\left(\log^2 n^{((j-1)k)}\log\log n^{((j-1)k)}\right)$$

in depth.

Proof. Let $m^{(i)}$ and $n^{(i)}$ be the number of non zeros and variables in $\boldsymbol{M}^{(i)}$ before the Sparsify call if there is. Lemmas 13.3.2 and 13.11.1 show that the i^{th} iteration takes $O(m^{(i)}+m^{(i+1)})$ work and $O(\log i\log n^{(i)})$ depth. By Lemma 13.13.2 the cost during these iterations excluding the call to Sparsify is:

$$\begin{aligned}\sum_{i=(j-1)k}^{jk-1}O\left(m^{(i)}+m^{(i+1)}\right) &\leq \sum_{i=(j-1)k}^{jk-1}2^{O(k\log^2 i)}n^{(i)}\log n^{(i)}\\ &\leq 2^{O(k\log^2(jk))}n^{((j-1)k)}\log n^{((j-1)k)}\end{aligned}$$

work and

$$\sum_{i=(j-1)k}^{jk-1}O\left(\log i\log n^{(i)}\right) \leq O(k\log((j-1)k)\log n^{((j-1)k)})$$

depth.

We now consider the call to Sparsify made during iteration $(j-1)k$. Access to a fast solver for the sampled bDD matrix is obtained via recursive calls to RecursiveConstruct. The guarantees of the chain given by Lemma 13.13.3 above means each solve takes depth

$$d_{eval}=\log^2 n^{((j-1)k)}\log\log n^{((j-1)k)},$$

and work

$$w_{eval}=n^{((j-1)k)}\log n^{((j-1)k)}$$

Incorporating these parameters into Lemma 13.12.1 allows us to bound the overhead from these solves by

$$2^{O(k\log^2(jk)))}n^{((j-1)k)}\log^2 n^{((j-1)k)}$$

work and

$$O\left(k\log(jk)\log n^{(jk)}\right)+O\left(\log^2 n^{((j-1)k)}\log\log\left(n^{((j-1)k)}\right)\right)$$

depth. □

Note that at the end of the j^{th} phase, the time required to construct an extra Schur complement chain for the Sparsify call is less than the remaining cost after the j^{th} phase. This is the reason why we use $2^{2ck\log^2(i+1)}$ as the reduction factor for the Sparsify call. The following theorem takes account for the recursive call and show the total running time for the algorithm.

Lemma 13.13.5. *With high probability,* RecursiveConstruct $\left(\boldsymbol{M}^{(0)}\right)$ *takes* $2^{O(\log n/k)}$ *depth and* $m \log n + 2^{O(k \log^2 k)} n \log^2 n$ *work.*

Proof. Lemma 13.13.2 shows that the call to Sparsify at the start of each phase j requires the construction of an extra Schur complement chain on a matrix with $n^{((j-1)k)}$ row/column blocks and at most $2^{ck \log^2((j-1)k)} n^{((j-2)k)} \log n^{((j-2)k)}$ non-zeros blocks. The guarantees of Lemma 13.12.1 gives that the number of non-zero block in the sparsified matrix is at most

$$C 2^{2ck \log^2((j-1)k)} n^{((j-1)k)} \log n^{((j-1)k)}$$

for some absolute constant C. Therefore the cost of this additional call is less than the cost of constructing the rest of the phases during the construction process. Therefore, every recursive call except the first one can be viewed as an extra branching factor of 2 at each subsequent phase.

Depth can be bounded by the total number of recursive invocations to RecursiveConstruct. The fact that $n^{(i)}$ is geometrically decreasing means there are $O(\log n/k)$ phases. Choosing k so that $k \log^2 k = o(\log n)$ gives a depth of:

$$\begin{aligned}
& \sum_{j=1}^{O(\log n/k)} 2^j \left(k \log(jk) \log n^{(jk)} + \log^2 n^{(jk)} \log\log n^{(jk)}\right) \\
= \; & 2^{O(\log n/k)} O\left(k \log\log n \log n^{(last)} + \log^2 n^{(last)} \log\log n^{(last)}\right) \\
= \; & 2^{O(\log n/k)} O\left(k^2 \log\log n + k^2 \log k\right) \\
= \; & 2^{O(\log n/k)} k^2 \log\log(n) \\
= \; & 2^{O(\log n/k)}.
\end{aligned}$$

For bounding work, we start with the sparse case since all intermediate matrices generated during the construction process have density at most $2^{O(k \log^2 k)}$. In this case, the extra branching factor of 2 at each phase can be accounted for by weighting the cost of iteration j by 2^j, giving:

$$\begin{aligned}
& \sum_{j=1}^{O(\log n/k)} 2^j \left(2^{O(k \log^2(jk))} n^{(jk)} \log^2 n^{(jk)}\right) \\
= \; & 2^{O(k \log^2 k)} n \log^2 n.
\end{aligned}$$

For the dense case, the first recursive call to Sparsify$(\boldsymbol{M}^{(0)}, O(1), 2^{2ck})$ is made to a graph whose edge count is much less. This leads to a geometric series, and an overhead of $O(m \log n)$ work at each step. As this can happen at most $O(\log n)$ times, it gives an additional factor of $O(\log n)$ in depth, which is still $2^{O(\log n/k)}$. The work obeys the recurrence:

$$W(m)_{dense} \leq \begin{cases} 2^{O(k \log^2 k)} n \log^2 n & \text{if } m \leq 2^{O(k \log^2 k)} n \log n, \text{ and} \\ W_{dense}(m/2) + m \log n + 2^{O(k \log^2 k)} n \log^2 n & \text{otherwise.} \end{cases}$$

which solves to:

$$W_{dense}(m) \leq O(m \log n) + 2^{O(k \log^2 k)} n \log^2 n.$$

□

To obtain Theorem 13.1.2, we simply choose an appropriate initial padding and set the parameter k.

Proof. (of Theorem 13.1.2) Lemma 13.2.2 allows us to solve the linear system $\boldsymbol{M} + \varepsilon\mu\boldsymbol{I}$ instead. Invoking Lemma 13.13.5 with $k = \log\log\log n$ gives a depth of $2^{O(\log n/\log\log\log n)}$, and work of $m\log n + 2^{O(\log\log\log n \log\log\log\log^4 n)} n\log^2 n$. The depth term can be simplified to $n^{o(1)}$ while the work term can be simplified to $2^{O(\log\log\log^2 n)} = \log^{o(1)} n$. □

■ 13.14 Weighted Expander Constructions

In this section, we give a linear time algorithm for computing linear sized spectral sparsifiers of complete and bipartite product demand graphs. These routines give Lemmas 13.11.2 and 13.11.1, which are crucial for controlling the densities of intermediate matrices in the spectral vertex sparsification algorithm from Section 13.11. Recall that the *product demand graph* with vertex set V and demands $\boldsymbol{d} : V \to \mathbb{R}_{>0}$ is the complete graph in which the weight of edge (u, v) is the product $d_u d_v$. Similarly, the *bipartite demand graph* with vertex set $U \cup V$ and demands $\boldsymbol{d} : U \cup V \to \mathbb{R}_{>0}$ is the complete bipartite graph on which the weight of the edge (u, v) is the product $d_u d_v$. Our routines are based on reductions to the unweighted, uniform case. In particular, we

1. Split all of the high demand vertices into many vertices that all have the same demand. This demand will still be the highest.
2. Given a graph in which almost all of the vertices have the same highest demand, we
   (a) drop all of the edges between vertices of lower demand,
   (b) replace the complete graph between the vertices of highest demand with an expander, and
   (c) replace the bipartite graph between the high and low demand vertices with a union of stars.
3. To finish, we merge back together the vertices that split off from each original vertex.

We start by showing how to construct the expanders that we need for step (2b). We state formally and analyze the rest of the algorithm for the complete case in the following two sections. We explain how to handle the bipartite case in Section 13.14.3.

Expanders give good approximations to unweighted complete graphs, and our constructions will use the spectrally best expanders—Ramanujan graphs. These are defined in terms of the eigenvalues of their adjacency matrices. We recall that the adjacency matrix of every d-regular graph has eigenvalue d with multiplicity 1 corresponding to the constant eigenvector. If the graph is bipartite, then it also has an eigenvalue of $-d$ corresponding to an eigenvector that takes value 1 on one side of the bipartition and -1 on the other side. These are called the *trivial* eigenvalues. A d-regular graph is called a Ramanujan graph if all of its non-trivial eigenvalues have absolute value at most $2\sqrt{d-1}$. Ramanujan graphs were constructed independently by Margulis [188] and Lubotzky, Phillips, and Sarnak [181]. The following theorem and proposition summarizes part of their results.

Theorem 13.14.1. *Let p and q be unequal primes congruent to 1 modulo 4. If p is a quadratic residue modulo q, then there is a non-bipartite Ramanujan graph of degree $p+1$ with $q^2(q-1)/2$ vertices. If p is not a quadratic residue modulo q, then there is a bipartite Ramanujan graph of degree $p+1$ with $q^2(q-1)$ vertices.*

The construction is explicit.

Proposition 13.14.2. *If $p < q$, then the graph guaranteed to exist by Theorem 13.14.1 can be constructed in parallel depth $O(\log n)$ and work $O(n)$, where n is its number of vertices.*

Proof. When p is a quadratic residue modulo q, the graph is a Cayley graph of $PSL(2, Z/qZ)$. In the other case, it is a Cayley graph of $PGL(2, Z/qZ)$. In both cases, the generators are determined by the $p+1$ solutions to the equation $p = a_0^2 + a_1^2 + a_2^2 + a_3^2$ where $a_0 > 0$ is odd and a_1, a_2, and a_3 are even. Clearly, all of the numbers a_0, a_1, a_2 and a_3 must be at most $\sqrt{p}$. So, we can compute a list of all sums $a_0^2 + a_1^2$ and all of the sums $a_2^2 + a_3^2$ with work $O(p)$, and thus a list of all $p+1$ solutions with work $O(p^2) < O(n)$.

As the construction requires arithmetic modulo q, it is convenient to compute the entire multiplication table modulo q. This takes time $O(q^2) < O(n)$. The construction also requires the computation of a square root of -1 modulo q, which may be computed from the multiplication table. Given this data, the list of edges attached to each vertex of the graph may be produced using linear work and logarithmic depth. □

For our purposes, there are three obstacles to using these graphs:

1. They do not come in every degree.
2. They do not come in every number of vertices.
3. Some are bipartite and some are not.

We handle the first two issues by observing that the primes congruent to 1 modulo 4 are sufficiently dense. To address the third issue, we give a procedure to convert a non-bipartite expander into a bipartite expander, and *vice versa*.

An upper bound on the gaps between consecutive primes congruent to 1 modulo 4 can be obtained from the following theorem of Tchudakoff.

Theorem 13.14.3. *For two integers a and b, let p_i be the ith prime congruent to a modulo b. For every $\varepsilon > 0$,*

$$p_{i+1} - p_i \leq O(p_i^{3/4+\varepsilon}).$$

Corollary 13.14.4. *There exists an n_0 so that for all $n \geq n_0$ there is a prime congruent to 1 modulo 4 between n and $2n$.*

We now explain how we convert between bipartite and non-bipartite expander graphs. To convert a non-bipartite expander into a bipartite expander, we take its double-cover. We recall that if $G = (V, E)$ is a graph with adjacency matrix $\boldsymbol{A}$, then its double-cover is the graph with adjacency matrix

$$\begin{pmatrix} 0 & \boldsymbol{A} \\ \boldsymbol{A}^T & 0 \end{pmatrix}.$$

It is immediate from this construction that the eigenvalues of the adjacency matrix of the double-cover are the union of the eigenvalues of $\boldsymbol{A}$ with the eigenvalues of $-\boldsymbol{A}$.

Proposition 13.14.5. *Let G be a connected, d-regular graph in which all matrix eigenvalues other than d are bounded in absolute value by λ. Then, all non-trivial adjacency matrix eigenvalues of the double-cover of G are also bounded in absolute value by λ.*

To convert a bipartite expander into a non-bipartite expander, we will simply collapse the two vertex sets onto one another. If $G = (U \cup V, E)$ is a bipartite graph, we specify how the vertices of V are mapped onto U by a permutation $\pi : V \to U$. We then define the *collapse* of G induced by π to be the graph with vertex set U and edge set

$$\{(u, \pi(v)) : (u, v) \in E\}.$$

Note that the collapse will have self-loops at vertices u for which $(u, v) \in E$ and $u = \pi(v)$. We assign a weight of 2 to every self loop. When a double-edge would be created, that is when $(\pi(v), \pi^{-1}(u))$ is also an edge in the graph, we give the edge a weight of 2. Thus, the collapse can be a weighted graph.

Proposition 13.14.6. *Let G be a d-regular bipartite graph with all non-trivial adjacency matrix eigenvalues bounded by λ, and let H be a collapse of G. Then, every vertex in H has weighted degree $2d$ and all adjacency matrix eigenvalues of H other than d are bounded in absolute value by 2λ.*

Proof. To prove the bound on the eigenvalues, let G have adjacency matrix

$$\begin{pmatrix} 0 & \boldsymbol{A} \\ \boldsymbol{A}^T & 0 \end{pmatrix}.$$

After possibly rearranging rows and columns, we may assume that the adjacency matrix of the collapse is given by

$$\boldsymbol{A} + \boldsymbol{A}^T.$$

Note that the self-loops, if they exist, correspond to diagonal entries of value 2. Now, let $\boldsymbol{x}$ be a unit vector orthogonal to the all-1s vector. We have

$$\boldsymbol{x}^T(\boldsymbol{A} + \boldsymbol{A}^T)\boldsymbol{x} = \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{x} \end{pmatrix}^T \begin{pmatrix} 0 & \boldsymbol{A} \\ \boldsymbol{A}^T & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{x} \end{pmatrix} \leq \lambda \| \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{x} \end{pmatrix} \|^2 \leq 2\lambda,$$

as the vector $[\boldsymbol{x}; \boldsymbol{x}]$ is orthogonal to the eigenvectors of the trivial eigenvalues of the adjacency matrix of G. □

We now state how bounds on the eigenvalues of the adjacency matrices of graphs lead to approximations of complete graphs and complete bipartite graphs.

Proposition 13.14.7. *Let G be a graph with n vertices, possibly with self-loops and weighted edges, such that every vertex of G has weighted degree d and such that all non-trivial eigenvalues of the adjacency matrix of G have absolute value at most $\lambda \leq d/2$. If G is not bipartite, then $(n/d)\boldsymbol{L}_G$ is an ε-approximation of K_n for $\varepsilon = (2 \ln 2)(\lambda)/d$. If G is bipartite, then $(n/d)\boldsymbol{L}_G$ is an ε-approximation of $K_{n,n}$ for $\varepsilon = (2 \ln 2)(\lambda)/d$.*

Proof. Let $\boldsymbol{A}$ be the adjacency matrix of G. Then,

$$\boldsymbol{L}_G = d\boldsymbol{I} - \boldsymbol{A}.$$

In the non-bipartite case, we observe that all of the non-zero eigenvalues of $\boldsymbol{L}_{K_n}$ are n, so for all vectors x orthogonal to the constant vector,

$$x^T \boldsymbol{L}_{K_n} x = n x^T x.$$

As all of the non-zero eigenvalues of $\boldsymbol{L}_G$ are between $d - \lambda$ and $d + \lambda$, for all vectors x orthogonal to the constant vector

$$n\left(1 - \frac{\lambda}{d}\right) x^T x \leq x^T (n/d) \boldsymbol{L}_G x \leq n\left(1 + \frac{\lambda}{d}\right) x^T x.$$

Thus,

$$\left(1 - \frac{\lambda}{d}\right) \boldsymbol{L}_{K_n} \preccurlyeq \boldsymbol{L}_G \preccurlyeq \left(1 + \frac{\lambda}{d}\right) \boldsymbol{L}_{K_n}.$$

In the bipartite case, we naturally assume that the bipartition is the same in both G and $K_{n,n}$. Now, let $\boldsymbol{x}$ be any vector on the vertex set of G. Both the graphs $K_{n,n}$ and $(n/d)G$ have Laplacian

matrix eigenvalue 0 with the constant eigenvector, and eigenvalue $2n$ with eigenvector $[\mathbf{1}; -\mathbf{1}]$. The other eigenvalues of the Laplacian of $K_{n,n}$ are n, while the other eigenvalues of the Laplacian of $(n/d)G$ are between

$$n\left(1-\frac{\lambda}{d}\right) \quad \text{and} \quad n\left(1+\frac{\lambda}{d}\right).$$

Thus,

$$\left(1-\frac{\lambda}{d}\right)\boldsymbol{L}_{K_{n,n}} \preccurlyeq \boldsymbol{L}_G \preccurlyeq \left(1+\frac{\lambda}{d}\right)\boldsymbol{L}_{K_{n,n}}.$$

The proposition now follows from our choice of ε, which guarantees that

$$e^{-\varepsilon} \leq 1-\lambda/d \quad \text{and} \quad 1+\lambda/d \leq e^{\varepsilon},$$

provided that $\lambda/d \leq 1/2$. □

Lemma 13.14.8. *There are algorithms that on input n and $\varepsilon > n^{-1/6}$ produce a graph having $O(n/\varepsilon^2)$ edges that is an $O(\varepsilon)$ approximation of $K_{n'}$ or $K_{n',n'}$ for some $n \leq n' \leq 8n$. These algorithms run in $O(\log n)$ depth and $O(n/\varepsilon^2)$ work.*

Proof. We first consider the problem of constructing an approximation of $K_{n',n'}$. By Corollary 13.14.4 there is a constant n_0 so that if $n > n_0$, then there is a prime q that is equivalent to 1 modulo 4 so that $q^2(q-1)$ is between and n and $8n$. Let q be such a prime and let $n' = q^2(q-1)$. Similarly, for ε sufficiently small, there is a prime p equivalent to 1 modulo 4 that is between $\varepsilon^{-2}/2$ and ε^{-2}. Our algorithm should construct the corresponding Ramanujan graph, as described in Theorem 13.14.1 and Proposition 13.14.2. If the graph is bipartite, then Proposition 13.14.7 tells us that it provides the desired approximation of $K_{n',n'}$. If the graph is not bipartite, then we form its double cover to obtain a bipartite graph and use Proposition 13.14.5 and Proposition 13.14.7 to see that it provides the desired approximation of $K_{n',n'}$.

The non-bipartite case is similar, except that we require a prime q so that $q^2(q-1)/2$ is between n and $8n$, and we use a collapse to convert a bipartite expander to a non-bipartite one, as analyzed in Proposition 13.14.6. □

For the existence results in Section 13.10, we just need to know that there exist graphs of low degree that are good approximations of complete graphs. We may obtain them from the recent theorem of Marcus, Spielman and Srivastava that there exist bipartite Ramanujan graphs of every degree and number of vertices [187].

Lemma 13.14.9. *For every integer n and even integer d, there is a weighted graph on n vertices of degree at most d that is a $4/\sqrt{d}$ approximation of K_n.*

Proof. The main theorem of [187] tells us that there is a bipartite Ramanujan graph on $2n$ vertices of degree k for every $k \leq n$. By Propositions 13.14.6 and 13.14.7, a collapse of this graph is a weighted graph of degree at most $2k$ that is a $(4\ln 2)/\sqrt{k}$ approximation of $K_{n,n}$. The result now follows by setting $d = 2k$. □

■ 13.14.1 Sparsifying Complete Product Demand Graphs

In the rest of the section, we will adapt these expander constructions to weighted settings. We start with product demand graphs.

Algorithm 46: $G' = \text{WeightedExpander}(\boldsymbol{d}, \varepsilon)$

1. Let $\hat{n}$ be the least integer greater than $2n/\varepsilon^2$ such that the algorithm described in Lemma 13.14.8 produces an ε-approximation of $K_{\hat{n}}$.
2. Let $t = \frac{\sum_k d_k}{\hat{n}}$.
3. Create a new product demand graph $\widehat{G}$ with demand vector $\hat{\boldsymbol{d}}$ by splitting each vertex i into a set of $\lceil d_i/t \rceil$ vertices, S_i:
   (a) $\lfloor d_i/t \rfloor$ vertices with demand t.
   (b) one vertex with demand $d_i - t\lfloor d_i/t \rfloor$.
4. Let H be a set of $\hat{n}$ vertices in $\widehat{G}$ with demand t, and let L contain the other vertices. Set $k = |L|$.
5. Partition H arbitrarily into sets $V_1, \ldots, V_k$, so that $|V_i| \geq \lfloor \hat{n}/k \rfloor$ for all $1 \leq i \leq k$.
6. Use the algorithm described in Lemma 13.14.8 to produce $\tilde{K}_{HH}$, an ε-approximation of the complete graph on H. Set
$$\widetilde{G} = t^2 \tilde{K}_{HH} + \sum_{l \in L} \frac{|H|}{|V_l|} \sum_{h \in V_l} \hat{d}_l \hat{d}_h (l, h).$$
7. Let G' be the graph obtained by collapsing together all vertices in each set S_i.

Lemma 13.11.1. There is a routine WeightedExpander($\boldsymbol{d}, \varepsilon$) such that for any $\varepsilon > 0$, and a demand vector $\boldsymbol{d} \in (\Re_{>0})^n$, WeightedExpander($\boldsymbol{d}, \varepsilon$) returns in $O(n\varepsilon^{-4})$ work and $O(\log n)$ depth a graph H with $O(n\varepsilon^{-4})$ edges such that

$$\boldsymbol{L}_H \approx_\varepsilon \boldsymbol{L}_{G(\boldsymbol{d})}.$$

Our algorithm for sparsifying complete product demand graphs begins by splitting the vertices of highest demands into many vertices. By *splitting* a vertex, we mean replacing it by many vertices whose demands sum to its original demand. In this way, we obtain a larger product demand graph. We observe that we can obtain a sparsifier of the original graph by sparsifying the larger graph, and then collapsing back together the vertices that were split.

Proposition 13.14.10. *Let G be a product demand graph with vertex set $\{1, \ldots, n\}$ and demands $\boldsymbol{d}$, and let $\widehat{G} = (\widehat{V}, \widehat{E})$ be a product demand graph with demands $\hat{\boldsymbol{d}}$. If there is a partition of $\widehat{V}$ into sets $S_1, \ldots, S_n$ so that for all $i \in V$, $\sum_{j \in S_i} \hat{d}_j = d_i$, then $\widehat{G}$ is a splitting of G and there is a matrix $\boldsymbol{M}$ so that*

$$\boldsymbol{L}_G = \boldsymbol{M} \boldsymbol{L}_{\widehat{G}} \boldsymbol{M}^T.$$

Proof. The (i, j) entry of matrix $\boldsymbol{M}$ is 1 if and only if $j \in S_i$. Otherwise, it is zero. □

We now show that we can sparsify G by sparsifying $\widehat{G}$.

Proposition 13.14.11. *Let $\widehat{G}_1$ and $\widehat{G}_2$ be graphs on the same vertex set $\widehat{V}$ such that $\widehat{G}_1 \approx_\varepsilon \widehat{G}_2$ for some ε. Let $S_1, \ldots, S_n$ be a partition of $\widehat{V}$, and let G_1 and G_2 be the graphs obtained by collapsing together all the vertices in each set S_i and eliminating any self loops that are created. Then*

$$G_1 \approx_\varepsilon G_2.$$

Proof. Let $\boldsymbol{M}$ be the matrix introduced in Proposition 13.14.10. The proof follows from the fact that

$$\boldsymbol{L}_{G_1} = \boldsymbol{M} \boldsymbol{L}_{\widehat{G}_1} \boldsymbol{M}^T \quad \text{and} \quad \boldsymbol{L}_{G_2} = \boldsymbol{M} \boldsymbol{L}_{\widehat{G}_2} \boldsymbol{M}^T.$$

□

For distinct vertices i and j, we let $\boldsymbol{(i, j)}$ denote the graph with an edge of weight 1 between vertex i and vertex j. If $i = j$, we let $\boldsymbol{(i, j)}$ be the empty graph. With this notation, we can express the product demand graph as

$$\sum_{i<j} d_i d_j \boldsymbol{(i, j)} = \frac{1}{2} \sum_{i,j \in V} d_i d_j \boldsymbol{(i, j)}.$$

This notation also allows us to precisely express our algorithm for sparsifying product demand graphs. This section and the next are devoted to the analysis of this algorithm. Given Proposition 13.14.11, we just need to show that $\widetilde{G}$ is a good approximation to $\widehat{G}$.

Proposition 13.14.12. *The number of vertices in $\widehat{G}$ is at most $n + \hat{n}$.*

Proof. The number of vertices in $\widehat{G}$ is

$$\sum_{i \in V} \lceil d_i / t \rceil \leq n + \sum_{i \in V} d_i / t = n + \hat{n}.$$

□

So, $k \leq n$ and $\hat{n} \geq 2k/\varepsilon^2$. That is, $|H| \geq 2 |L| / \varepsilon^2$. In the next section, we prove the lemmas that show that for these special product demand graphs $\widehat{G}$ in which almost all weights are the maximum, our algorithm produces a graph $\widetilde{G}$ that is a good approximation of $\widehat{G}$.

Proof. (of Lemma 13.11.1) The number of vertices in the graph $\widehat{G}$ will be between $n + 2n/\varepsilon^2$ and $n + 16n/\varepsilon^2$. So, the algorithm described in Lemma 13.14.8 will take $O(\log n)$ depth and $O(n/\varepsilon^4)$ work to produce an ε approximation of the complete graph on $\hat{n}$ vertices. This dominates the computational cost of the algorithm.

Proposition 13.14.11 tells us that G' approximates G at least as well as $\widetilde{G}$ approximates $\widehat{G}$. To bound how well $\widetilde{G}$ approximates $\widehat{G}$, we use two lemmas that are stated in the next section. Lemma 13.14.14 shows that

$$\widehat{G}_{HH} + \widehat{G}_{LH} \approx_{O(\varepsilon^2)} \widehat{G}.$$

Lemma 13.14.16 shows that

$$\widehat{G}_{HH} + \widehat{G}_{LH} \approx_{4\varepsilon} \widehat{G}_{HH} + \sum_{l \in L} \frac{|H|}{|V_l|} \sum_{h \in V_l} \hat{d}_l \hat{d}_h \mathbf{(}l, h\mathbf{)}.$$

And, we already know that $t^2 \tilde{K}$ is an ε-approximation of $\widehat{G}_{HH}$. Fact 13.2.1 says that we can combine these three approximations to conclude that $\widetilde{G}$ is an $O(\varepsilon)$-approximation of $\widehat{G}$. □

■ 13.14.2 Product demand graphs with most weights maximal

In this section, we consider product demand graphs in which almost all weights are the maximum. For simplicity, we make a slight change of notation from the previous section. We drop the hats, we let n be the number of vertices in the product demand graph, and we order the demands so that

$$d_1 \leq d_2 \leq \cdots \leq d_k \leq d_{k+1} = \cdots = d_n = 1.$$

We let $L = \{1, \ldots, k\}$ and $H = \{k+1, \ldots, n\}$ be the set of low and high demand vertices, respectively. Let G be the product demand graph corresponding to $\boldsymbol{d}$, and let G_{LL}, G_{HH} and G_{LH} be the subgraphs containing the low-low, high-high and low-high edges respectively. We now show that little is lost by dropping the edges in G_{LL} when k is small.

Our analysis will make frequent use of the following Poincare inequality:

Lemma 13.14.13. *Let $c\mathbf{(}u, v\mathbf{)}$ be an edge of weight c and let P be a path from from u to v consisting of edges of weights $c_1, c_2, \cdots, c_k$. Then*

$$c\mathbf{(}u, v\mathbf{)} \preceq c \left(\sum c_i^{-1} \right) P.$$

As the weights of the edges we consider in this section are determined by the demands of their vertices, we introduce the notation

$$\mathbf{(}i, j\mathbf{)}_2 = d_i d_j \mathbf{(}i, j\mathbf{)}.$$

With this notation, we can express the product demand graph as

$$\sum_{i<j} \mathbf{(}i, j\mathbf{)}_2 = \frac{1}{2} \sum_{i,j \in V} \mathbf{(}i, j\mathbf{)}_2.$$

Lemma 13.14.14. *If $|L| \leq |H|$, then*

$$G_{HH} + G_{LH} \approx_{3\frac{|L|}{|H|}} G.$$

Proof. The lower bound $G_{HH} + G_{LH} \preceq G_{HH} + G_{LH} + G_{LL}$ follows from $G_{LL} \succeq 0$.

Using lemma 13.14.13 and the assumptions $d_l \leq 1$ for $l \in L$ and and $d_h = 1$ for $h \in H$, we derive for every $l_1, l_2 \in L$,

$$(l_1, l_2)_2 = \frac{1}{|H|^2} \sum_{h_1, h_2 \in H} (l_1, l_2)_2$$

(by Lemma 13.14.13)

$$\begin{aligned}
&\preceq \frac{1}{|H|^2} \sum_{h_1, h_2 \in H} d_{l_1} d_{l_2} \left(\frac{1}{d_{l_1} d_{h_1}} + \frac{1}{d_{h_1} d_{h_2}} + \frac{1}{d_{h_2} d_{l_2}} \right) ((l_1, h_1)_2 + (h_1, h_2)_2 + (h_2, l_2)_2) \\
&\preceq \frac{3}{|H|^2} \sum_{h_1, h_2 \in H} ((l_1, h_1)_2 + (h_1, h_2)_2 + (h_2, l_2)_2) \\
&= \frac{3}{|H|} \sum_{h \in H} ((l_1, h)_2 + (l_2, h)_2) + \frac{6}{|H|^2} G_{HH}.
\end{aligned}$$

So,

$$\begin{aligned}
G_{LL} &= \frac{1}{2} \sum_{l_1, l_2 \in L} (l_1, l_2)_2 \\
&\preceq \frac{1}{2} \sum_{l_1, l_2} \left(\frac{3}{|H|} \sum_{h \in H} ((l_1, h)_2 + (l_2, h)_2) + \frac{6}{|H|^2} G_{HH} \right) \\
&= \frac{3\,|L|}{|H|} G_{LH} + \frac{3\,|L|^2}{|H|^2} G_{HH}.
\end{aligned}$$

The assumption $|L| \leq |H|$ then allows us to conclude

$$G_{HH} + G_{LH} + G_{LL} \preceq \left(1 + 3 \frac{|L|}{|H|} \right) (G_{HH} + G_{LH}).$$

□

Using a similar technique, we will show that the edges between L and H can be replaced by the union of a small number of stars. In particular, we will partition the vertices of H into k sets, and for each of these sets we will create one star connecting the vertices in that set to a corresponding vertex in L.

We employ the following consequence of the Poincare inequality in Lemma 13.14.13.

Lemma 13.14.15. *For any $\varepsilon \leq 1$, $l \in L$ and $h_1, h_2 \in H$,*

$$\varepsilon (h_1, l)_2 + (1/2)(h_1, h_2)_2 \approx_{4\sqrt{\varepsilon}} \varepsilon (h_2, l)_2 + (1/2)(h_1, h_2)_2.$$

Proof. By applying Lemma 13.14.13 and recalling that $d_{h_1} = d_{h_2} = 1$ and $d_l \leq 1$, we compute

$$\begin{aligned}
(h_1, l)_2 &\preceq d_{h_1} d_l \left(\frac{\sqrt{\varepsilon}}{d_{h_1} d_{h_2}} + \frac{1}{d_{h_2} d_l} \right) \left(\frac{1}{\sqrt{\varepsilon}} (h_1, h_2)_2 + (h_2, l)_2 \right) \\
&\preceq \frac{1 + \sqrt{\varepsilon}}{\sqrt{\varepsilon}} (h_1, h_2)_2 + (1 + \sqrt{\varepsilon})(h_2, l)_2 \\
&\preceq (1 + \sqrt{\varepsilon})(h_2, l)_2 + \frac{2}{\sqrt{\varepsilon}} (h_1, h_2)_2.
\end{aligned}$$

Multiplying both sides by ε and adding $(1/2)(h_1, h_2)_2$ then gives

$$\begin{aligned}\varepsilon(h_1, l)_2 + (1/2)(h_1, h_2)_2 &\preccurlyeq (1 + \sqrt{\varepsilon})\varepsilon(h_2, l)_2 + (2\sqrt{\varepsilon} + 1/2)(h_1, h_2)_2 \\ &\preccurlyeq (1 + 4\sqrt{\varepsilon})\left(\varepsilon(h_2, l)_2 + (1/2)(h_1, h_2)_2\right) \\ &\preccurlyeq e^{4\sqrt{\varepsilon}}\left(\varepsilon(h_2, l)_2 + (1/2)(h_1, h_2)_2\right).\end{aligned}$$

By symmetry, we also have

$$\varepsilon(h_2, l)_2 + (1/2)(h_1, h_2)_2 \preccurlyeq e^{4\sqrt{\varepsilon}}\left(\varepsilon(h_1, l)_2 + (1/2)(h_1, h_2)_2\right).$$

□

Lemma 13.14.16. *Recall that $L = \{1, \ldots, k\}$ and let $V_1, \ldots, V_k$ be a partition of $H = \{k+1, \ldots, n\}$ so that $|V_l| \geq s$ for all l. Then,*

$$G_{HH} + G_{LH} \approx_{4/\sqrt{s}} G_{HH} + \sum_{l \in L} \frac{|H|}{|V_l|} \sum_{h \in V_l} (l, h)_2.$$

Proof. Observe that

$$G_{LH} = \sum_{l \in L} \sum_{h \in H} (l, h)_2.$$

For each $l \in L$, $h_1 \in H$ and $h_2 \in V_l$ we apply Lemma 13.14.15 to show that

$$\frac{1}{|V_l|}(l, h_1)_2 + \frac{1}{2}(h_1, h_2)_2 \approx_{4/\sqrt{s}} \frac{1}{|V_l|}(l, h_2)_2 + \frac{1}{2}(h_1, h_2)_2.$$

Summing this approximation over all $h_2 \in V_l$ gives

$$(l, h_1)_2 + \sum_{h_2 \in V_l} \frac{1}{2}(h_1, h_2)_2 \approx_{4/\sqrt{s}} \sum_{h_2 \in V_l} \left(\frac{1}{|V_l|}(l, h_2)_2 + \frac{1}{2}(h_1, h_2)_2\right).$$

Summing the left-hand side of this this approximation over all $l \in L$ and $h_1 \in H$ gives

$$\sum_{l \in L, h_1 \in H} (l, h_1)_2 + \sum_{h_2 \in V_l} \frac{1}{2}(h_1, h_2)_2 = \sum_{l \in L, h_1 \in H} (l, h_1)_2 + \frac{1}{2} \sum_{h_1 \in H, l \in L} \sum_{h_2 \in V_l} (h_1, h_2)_2 = G_{LH} + G_{HH}.$$

On the other hand, the sum of the right-hand terms gives

$$G_{HH} + \sum_{l \in L, h_1 \in H} \sum_{h_2 \in V_l} \frac{1}{|V_l|}(l, h_2)_2 = G_{HH} + \sum_{l \in L} \sum_{h_2 \in V_l} \frac{|H|}{|V_l|}(l, h_2)_2.$$

□

13.14.3 Weighted Bipartite Expanders

This construction extends analogously to bipartite product graphs. The bipartite product demand graph of vectors $(\boldsymbol{d}^A, \boldsymbol{d}^B)$ is a complete bipartite graph whose weight between vertices $i \in A$ and $j \in B$ is given by $w_{ij} = d_i^A d_j^B$. The main result that we will prove is:

Lemma 13.11.2. There is a routine WeightedBipartiteExpander$(\boldsymbol{d}^A, \boldsymbol{d}^B, \varepsilon)$ such that for any demand vectors $\boldsymbol{d}^A$ and $\boldsymbol{d}^B$ of total length n, and a parameter ε, it returns in $O(n\varepsilon^{-4})$ work and $O(\log n)$ depth a bipartite graph H between A and B with $O(n\varepsilon^{-4})$ edges such that

$$\boldsymbol{L}_H \approx_\varepsilon \boldsymbol{L}_{G(\boldsymbol{d}^A, \boldsymbol{d}^B)}.$$

Without loss of generality, we will assume $d_1^A \geq d_2^A \geq \cdots \geq d_{n^A}^A$ and $d_1^B \geq d_2^B \geq \cdots \geq d_{n^B}^B$. As the weights of the edges we consider in this section are determined by the demands of their vertices, we introduce the notation

$$(i,j)_2 = d_i^A d_j^B (i,j).$$

Our construction is based on a similar observation that if most vertices on A side have d_i^A equaling to d_1^A and most vertices on B side have d_i^B equaling to d_1^B, then the uniform demand graph on these vertices dominates the graph.

Algorithm 47: $G' = \text{WeightedBipartiteExpander}(\boldsymbol{d}^A, \boldsymbol{d}^B, \varepsilon)$

1. Let $n' = \max(n^A, n^B)$ and $\hat{n}$ be the least integer greater than $2n'/\varepsilon^2$ such that the algorithm described in Lemma 13.14.8 produces an ε-approximation of $K_{\hat{n},\hat{n}}$.
2. Let $t^A = \frac{\sum_k d_k^A}{\hat{n}}$ and $t^B = \frac{\sum_k d_k^B}{\hat{n}}$.
3. Create a new bipartite demand graph $\widehat{G}$ with demands $\hat{\boldsymbol{d}}^A$ and $\hat{\boldsymbol{d}}^B$ follows:
   (a) On the side A of the graph, for each vertex i, create a subset S_i consisting of $\lceil d_i^A/t^A \rceil$ vertices:
      i. $\lfloor d_i^A/t^A \rfloor$ with demand t^A.
      ii. one vertex with demand $d_i^A - t^A \lfloor d_i^A/t^A \rfloor$.
   (b) Let H^A contain $\hat{n}$ vertices of A of with demand t^A, and let L^A contain the rest. Set $k^A = |L^A|$.
   (c) Create the side B of the graph with partition H^B, L^B and demand vector $\hat{\boldsymbol{d}}^B$ similarly.
4. Partition H^A into sets of size $|V_i^A| \geq \lfloor \hat{n}/k^A \rfloor$, one corresponding to each vertex $l \in L^A$. Partition V_B similarly.
5. Let $\tilde{K}_{H^A H^B}$ be a bipartite expander produced by Lemma 13.14.8 that ε-approximates $K_{\hat{n}\hat{n}}$, identified with the vertices H^A and H^B.
6. Set
$$\widetilde{G} = t^A t^B \tilde{K} + \sum_{l \in L^A} \frac{|H^B|}{|V_l^B|} \sum_{h \in V_l^B} \hat{d}_l^A \hat{d}_h^B (l,h) + \sum_{l \in L^B} \frac{|H^A|}{|V_l^A|} \sum_{h \in V_l^A} \hat{d}_l^B \hat{d}_h^A (l,h).$$
7. Let G' be the graph obtained by collapsing together all vertices in each set S_i^A and S_i^B.

Similarly to the non-bipartite case, the Poincare inequality show that the edges between low demand vertices can be completely omitted if there are many high demand vertices which allows the demand routes through high demand vertices.

Lemma 13.14.17. *Let G be the bipartite product demand graph of the demand $(\boldsymbol{d}_i^A, \boldsymbol{d}_j^B)$. Let H^A a subset of vertices on A side with demand higher than the set of remaining vertices L^A on A side. Define H^B, L^B similarly. Assume that $|L^A| \leq |H^A|$ and $|L^B| \leq |H^B|$, then*

$$G_{H^A H^B} + G_{H^A L^B} + G_{L^A H^B} \approx_{3\max\left(\frac{|L^A|}{|H^A|}, \frac{|L^B|}{|H^B|}\right)} G.$$

Proof. The proof is analogous to Lemma 13.14.14, but with the upper bound modified for bipartite

graphs.

For every edge l_A, l_B, we embed it evenly into paths of the form l_A, h_B, h_A, l_B over all choices of h_A and h_B. The support of this embedding can be calculated using Lemma 13.14.13, and the overall accounting follows in the same manner as Lemma 13.14.14. □

It remains to show that the edges between low demand and high demand vertices can be compressed into a few edges. The proof here is also analogous to Lemma 13.14.15: we use the Poincare inequality to show that all demands can routes through high demand vertices. The structure of the bipartite graph makes it helpful to further abstract these inequalities via the following Lemma for four edges.

Lemma 13.14.18. *Let G be the bipartite product demand graph of the demand $(\boldsymbol{d}_i^A, \boldsymbol{d}_j^B)$. Given $h_A, l_A \in A$ and $h_{B,1}, h_{B,2} \in B$. Assume that $d^A_{h_A} = d^B_{h_{B,1}} = d^B_{h_{B,2}} \geq d^A_{l_A}$. For any $\varepsilon < 1$, we have*

$$\varepsilon(l_A, h_{B,1})_2 + (h_A, h_{B,2})_2 + (h_A, h_{B,1})_2 \approx_{3\sqrt{\varepsilon}} \varepsilon(l_A, h_{B,2})_2 + (h_A, h_{B,2})_2 + (h_A, h_{B,1})_2.$$

Proof. Using Lemma 13.14.13 and $d^A_{h_A} = d^B_{h_{B,1}} = d^B_{h_{B,2}} \geq d^A_{l_A}$, we have

$$\begin{aligned}
&(l_A, h_{B,1})_2 \\
&\preceq d^A_{l_A} d^B_{h_{B,1}} \left(\frac{1}{d^A_{l_A} d^B_{h_{B,2}}} + \frac{\sqrt{\varepsilon}}{d^A_{h_A} d^B_{h_{B,2}}} + \frac{\sqrt{\varepsilon}}{d^A_{h_A} d^B_{h_{B,1}}} \right) \left((l_A, h_{B,2})_2 + \frac{1}{\sqrt{\varepsilon}}(h_A, h_{B,2})_2 + \frac{1}{\sqrt{\varepsilon}}(h_A, h_{B,1})_2 \right) \\
&\preceq (1 + 2\sqrt{\varepsilon})(l_A, h_{B,2})_2 + \frac{1 + 2\sqrt{\varepsilon}}{\sqrt{\varepsilon}}(h_A, h_{B,2})_2 + \frac{1 + 2\sqrt{\varepsilon}}{\sqrt{\varepsilon}}(h_A, h_{B,1})_2.
\end{aligned}$$

Therefore,

$$\varepsilon(l_A, h_{B,1})_2 + (h_A, h_{B,2})_2 + (h_A, h_{B,1})_2 \preceq (1 + 3\sqrt{\varepsilon}) \left(\varepsilon(l_A, h_{B,2})_2 + (h_A, h_{B,2})_2 + (h_A, h_{B,1})_2 \right).$$

The other side is similar due to the symmetry. □

(of Lemma 13.11.2) The proof is analogous to Lemma 13.11.1. After the splitting, the demands in H^A are higher than the demands in L^A and so is H^B to L^B. Therefore, Lemma 13.14.17 shows that that

Proof.

$$\widehat{G}_{H^A H^B} + \widehat{G}_{H^A L^B} + \widehat{G}_{L^A H^B} \approx_{3\varepsilon^2/2} \widehat{G}.$$

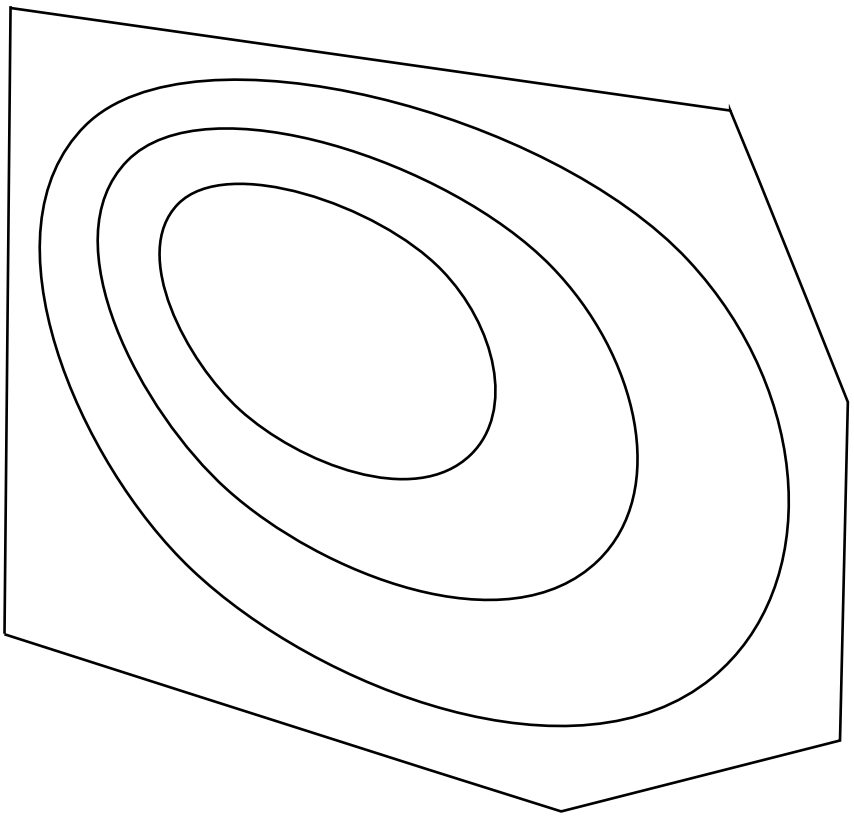
By a proof analogous to Lemma 13.14.16, one can use Lemma 13.14.18 to show that

$$\widehat{G}_{H^A H^B} + \widehat{G}_{H^A L^B} + \widehat{G}_{L^A H^B} \approx_{O(\varepsilon)} \widehat{G}_{H^A H^B} + \frac{|H^B|}{|V_l^B|} \sum_{h \in V_l^B} \hat{d}_l^A \hat{d}_h^B (l, h) + \sum_{l \in L^B} \frac{|H^A|}{|V_l^A|} \sum_{h \in V_l^A} \hat{d}_l^B \hat{d}_h^A (l, h).$$

And, we already know that $t^A t^B \tilde{K}$ is an ε-approximation of $\widehat{G}_{H^A H^B}$. Fact 13.2.1 says that we can combine these three approximations to conclude that $\widetilde{G}$ is an $O(\varepsilon)$-approximation of $\widehat{G}$. □

Part IV

Geometry



Chapter 14

Sampling In Sub-Quadratic Steps

14.1 Introduction

In this chapter, we study the problem of sampling a high-dimensional polytope, a fundamental algorithmic problem with many applications. The problem can be solved in randomized polynomial time. Progress on the more general problem of sampling a convex body given only by a membership oracle [78, 79, 175, 176, 177, 131, 179, 178, 261] has lead to a set of general-purpose techniques, both for algorithms and for analysis in high dimension. All known algorithms are based on sampling by discrete-time Markov chains. These include the ball walk [174], hit-and-run [241, 179] and the Dikin walk [132], the last requiring stronger access than a membership oracle. In each case, the main challenge is analyzing the mixing time of the Markov chain. For a polytope defined by m inequalities in $\mathbb{R}^n$, the current best complexity of sampling is roughly the minimum of $n^3 \cdot mn$ and $mn \cdot mn^{\omega-1}$ where the first factor in each term is the mixing time and the second factor is the time to implement one step. In fact, the bound of n^3 on the mixing time (achieved by the ball walk and hit-and-run) holds for arbitrary convex bodies, and $O(mn)$ is just the time to implement a membership oracle. The second term is for the Dikin walk, for which Kannan and Narayanan showed a mixing time of $O(mn)$ for the Dikin walk [132], with each step implementable in roughly matrix multiplication time. For general convex bodies given by membership oracles, $\Omega(n^2)$ is a lower bound on the number of oracle calls for all known walks. A quadratic upper bound would essentially follow from a positive resolution of the KLS hyperplane conjecture (we mention that [59] show a mixing bound of $\tilde{O}(n^2)$ for the ball walk for sampling from a Gaussian distribution restricted to a convex body). The quadratic barrier seems inherent for sampling convex bodies given by membership oracles, holding even for cubes and cylinders for the known walks based on membership oracles. It has not been surpassed thus far for explicitly described polytopes by any means.

For a polytope in $\mathbb{R}^n$, the Euclidean perspective is natural and predominant. The game has been to define a process on the points of the polytope so that the distribution of the current point quickly approaches the uniform (or a desired stationary distribution). The difficulty is that for points near the boundary of a body, steps are necessarily small due to the nature of volume distribution in high dimension. The Dikin walk departs from the standard perspective by making the distribution of the next step depend on the distance(s) of the current point to the boundary of the polytope. At each step, the process picks a random point from a suitable ellipsoid that is guaranteed to almost lie inside. This process adapts to the boundary, but runs into the similar difficulties — the ellipsoid has to shrink as the point approaches the boundary in order to ensure that (a) the stationary distribution is close to uniform and (b) the 1-step distribution has to be *smooth*, both necessary properties for rapidly converging to the uniform distribution. While the walk has the appealing property of being affine-invariant, and thus avoiding having to explicitly *round* the polytope, the current best upper bound is still quadratic, even for cylinders.

An alternative approach for sampling is the simulation of Brownian motion with boundary reflection [114, 67, 64, 43]. While there has been much study of this process, several difficulties arise in

turning it into an efficient algorithm. In particular, if the current point is close to the boundary of the polytope, extra care is needed in simulation and the process effectively slows down. However, if it is deep inside the polytope, we should expect that Brownian motion locally looks like a Gaussian distribution and hence it is easier to simulate. This suggests that the standard Euclidean metric, which does not take into account distance to the boundary, is perhaps not the right notion for getting a fast sampling algorithm.

In this chapter, we combine the use of stochastic differential equations (SDE) with non-Euclidean geometries (Riemannian manifolds) to break the quadratic barrier for mixing in polytopes. As a result we obtain significantly faster sampling algorithms.

Roughly speaking, our work is based on three key conceptual ideas. The first is the use of a Riemannian metric rather than the Euclidean metric. This allows us to scale space as we get closer to the boundary and incorporate boundary information much more smoothly. This idea was already used by Narayanan [198] to extend the Dikin walk to more general classes of convex bodies. The relevant metrics are induced by Hessians of convex barrier functions, objects that have been put to remarkable use for efficient linear and convex optimization [211]. The second idea is to simulate a Stochastic Differential Equation (SDE) on the manifold corresponding to the metric, via its geodesics (shortest-path curves, to be defined presently). Unlike straight lines, geodesics bend away from the boundary and this allows us to take larger steps while staying inside the polytope. The third idea is to use a modification of standard Brownian motion via a *drift* term, i.e., rather than centering the next step at the current point, we first shift the current point deterministically, then take a random step. This drift term compensates the changes of the step size and this makes the process closer to symmetric. Taken together, these ideas allow us to simulate an SDE by a discrete series of ordinary differential equations (ODE), which we are able to solve efficiently to the required accuracy. In order to state our contributions and results more precisely, we introduce some background, under three headings.

Riemannian Geometry. A manifold can be viewed as a surface embedded in a Euclidean space. Each point in the manifold (on the surface), has a tangent space (the linear approximate of the surface at that point) and a local metric. For a point x in a manifold M, the metric at x is defined by a positive definite matrix $g(x)$ and the length of a vector u in the tangent space T_xM is defined as $\|u\|_x \stackrel{\text{def}}{=} u^T g(x) u$. By integration, the length of any curve on the manifold is defined as $\int \|\frac{dc}{dt}\|_{c(t)}$. A basic fact about Riemannian manifolds is that for any point in the manifold, in any direction (from the tangent space), there is locally a shortest path (*geodesic*) starting in that direction. In Euclidean space, this is just a straight line in the starting direction. Previous random walks involve generating a random direction and going along a straight line in that direction. However such straight lines do not take into account the local geometry, while geodesics do. We give formal definitions in Section 14.2.1.

Hessian Manifolds. In this chapter, we are concerned primarily with Riemannian manifolds induced by the Hessians of smooth (infinitely differentiable) convex functions. More precisely, for any such function ϕ, the metric (of the manifold induced by ϕ) at a point $x \in M$ is given by the Hessian of ϕ at x, i.e., $\nabla^2\phi(x)$. Since ϕ is convex, the Hessian is positive definite and hence the Riemannian manifold induced by ϕ is well-defined and is called a Hessian manifold. In the context of convex optimization, we are interested in a class of convex functions called self-concordant barriers. Such convex function is smooth in a certain sense and blows up on the boundary of a certain convex set. The class of Hessian manifolds corresponding to self-concordant barriers has been studied and used to study interior-point methods (IPM) [137, 214, 212].

The barrier we are particularly interested is the logarithmic barrier. For a polytope $Ax > b$, with

$A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$, for any x in the polytope, the logarithmic barrier is given by

$$\phi(x) = -\sum_i \ln(Ax-b)_i.$$

In this chapter, we study the logarithmic barrier and show how to use this to develop a faster algorithm for sampling polytopes.

Stochastic Differential Equations. Given a self-concordant barrier ϕ on a convex set K, there is an "unique" Brownian motion with drift on the Hessian manifold M induced by ϕ that has uniform stationary distribution. In the Euclidean coordinate system, the SDE is given by

$$dx_t = \mu(x_t)dt + \left(\nabla^2\phi(x_t)\right)^{-1/2} dW_t \tag{14.1}$$

where the first term, called *drift,* is given by:

$$\mu_i(x_t) = \frac{1}{2}\sum_{j=1}^{n} \frac{\partial}{\partial x_j}\left(\left(\nabla^2\phi(x_t)\right)^{-1}\right)_{ij}. \tag{14.2}$$

This suggests an approach for generating a random point in a polytope, namely to simulate the SDE. The running time of such an algorithm depends on the convergence rate of the SDE and the cost of simulating the SDE in discrete time steps.

Since the SDE is defined on the Riemannian manifold M, it is natural to consider the following geodesic walk:

$$x^{(j+1)} = \exp_{X^{(j)}}(\sqrt{h}w + \frac{h}{2}\mu(x^{(j)})) \tag{14.3}$$

where $\exp_{x^{(j)}}$ is a map from $T_{x^{(j)}}M$ back to the manifold, w is a random Gaussian vector on $T_{x^{(j)}}M$, $\mu(x^{(j)}) \in T_{x^{(j)}}M$ is the drift term and h is the step size. The coefficient of the drift term depends on the coordinate system we use; as we show in Lemma 14.3.1, for the coordinate system induced by a geodesic, the drift term is $\mu/2$ instead of μ as in (14.1). The Gaussian vector w has mean 0 and variance 1 in the metric at x, i.e.$\mathbf{E}_w\|w\|_x^2 = 1$. We write it as $w \sim N_x(0, I)$.

It can be shown that this discrete walk converges to (14.1) as $h \to 0$ and it converges in a rate faster than the walk suggested by Euclidean coordinates (Theorem 14.3.3), namely, $x^{(j+1)} = x^{(j)} + \sqrt{h}w + h\mu(x^{(j)})$. (Note the drift here is proportional to h and not $h/2$.) This is the reason we study the geodesic walk.

14.1.1 Algorithm

The algorithm is a discrete-time simulation of the geodesic process (14.3). For step-size h chosen in advance, let $p(x \xrightarrow{w} y)$ be the probability density (in Euclidean coordinates) of going from x to y using the local step w. In general, the stationary distribution of the geodesic process is not uniform and it is difficult to analyze the stationary distribution unless h is very small, which would lead to a high number of steps. To get around this issue, we use the standard method of rejection sampling to get a uniform stationary distribution. We call this geodesic walk.

To implement the geodesic walk, we need to compute the exponential map $\exp_x$ and $p(x \xrightarrow{w} y)$ efficiently. We show how to implement this in Section 14.4.6.Our goal is to make sure that each iteration of geodesic walks only uses matrix multiplication and matrix inverse for $O(\log^{O(1)} m)$ many $O(m) \times O(m)$-size matrices, and the rejection probability is small.

It turns out that the problems of computing the exponential map and $p(x \xrightarrow{w} y)$ are similar; both involve solving some ordinary differential equations (ODEs) to accuracy $1/n^{\Theta(1)}$. Hence, one can view

Algorithm 48: Geodesic Walk (See Algo 49 for details)

Pick a Gaussian random vector $w \sim N_x(0,1)$, i.e. $\mathbf{E}_w \|w\|_x^2 = 1$.
Compute $y = \exp_x(\sqrt{h}w + \frac{h}{2}\mu(x))$ where $\mu(x)$ is given by (14.2).
Let $p(x \xrightarrow{w} y)$ is the probability density of going from x to y using the above step w.
Compute a corresponding w' s.t. $x = \exp_y(\sqrt{h}w' + \frac{h}{2}\mu(y))$.
With probability $\min\left(1, \frac{p(y \xrightarrow{w'} x)}{p(x \xrightarrow{w} y)}\right)$, go to y;
Otherwise, stay at x.

the geodesic walk as reducing the problem of simulating an SDE 14.1 to solving a sequence of ODEs. Although solving ODEs is well-studied, existing literature seems quite implicit about the dependence on the dimension and hence it is difficult to apply it directly. In Section 14.6, we rederive some existing result about solving ODEs efficiently, but with quantitative estimates of the dependence on the dimension and desired error.

14.1.2 Main results

In this chapter, we analyze the geodesic walk for the logarithmic barrier. The convergence analysis will need tools from Riemannian geometry and stochastic calculus, while the implementation uses efficient (approximate) solution of ODEs. Both aspects appear to be of independent interest. For the reader unfamiliar with these topics, we include an exposition of the relevant background.

We analyze geodesic walk in general and give a bound on the mixing time in terms of a set of manifold parameters (Theorem 14.4.2). Applying this on the logarithmic barrier, we obtain a faster sampling algorithm, going below the mn mixing time of the Dikin walk, while maintaining the same per-step complexity.

Theorem 14.1.1 (Sampling with logarithmic barrier)**.** *Given a polytope $\{Ax \geq b\}$ with an uniformly random initial point in the polytope. Using the logarithmic barrier, we can sample another uniformly random point in the polytope via geodesic walk using $\tilde{O}\left(mn^{\frac{3}{4}}\right)$ steps, with each step taking $\tilde{O}\left(mn^{\omega-1}\right)$ time to implement.*

The implementation of each step of sampling is based on the efficient solution of high-dimensional ODEs (Theorem 14.6.4); this might have other applications.

14.1.3 Discussion and open problems

At a high level, our algorithm is based on the following sequence of natural choices. First, we consider an Brownian motion that gives uniform stationary distribution and such Brownian motion is unique for a given metric (Fokker-Planck equation). Since the set we sample is a polytope, we use the metric given by the Hessian of a self-concordant barrier, a well-studied class of metrics in convex optimization. This allows us to reduce the sampling problem to the problem of simulating an SDE[1]. To simulate an SDE, we apply the Milstein method, well-known in that field. To implement the Milstein method, we perform a change of coordinates to make the metric locally constant and this makes most of the terms in the Milstein approximation method vanish. These coordinates are called normal coordinates

[1]Coincidentally, when we use the best known self-concordant barrier, the canonical barrier, our SDE becomes a Brownian motion with drift on an Einstein manifold. This is similar to how physical particles mix under general relativity, a sampling algorithm that has been executed for over 10 billion years!

and can be calculated by geodesics (Lemma 14.3.1). After all of this, we obtain the step of our walk (14.3).

There are two choices which are not natural. First, it is unclear that Hessians of self-concordant barriers are the best metrics for the sampling problems; after all, these barriers are designed for solving linear programs. Second, it is unclear that Milstein method is the best choice. There are other numerical method for SDE with better convergence rates, such as higher-order Runge-Kutta schemes. However, these methods are very complex and it is not clear how to implement the steps of these methods in $\tilde{O}\left(mn^{\omega-1}\right)$ time.

Clearly, we are not simultaneously experts on Riemannian geometry, numerical SDE, numerical ODE and convex geometry; we view this chapter as a sampling algorithm that connects different areas while improving the state of the art. Although the sampling problem is a harder problem than solving linear programs, the step size of geodesic walk we use is larger than that of the short-step interior point method. We hope that the relations revealed in this chapter might be useful for further development of both samplers and linear programming solvers.

There are several avenues for improving the sampling complexity further. One is to take longer steps and use a higher-order simulation of the SDE that is accurate up to a larger distance. Another part that is not tight in the analysis is the isoperimetry. At the moment, we incur a linear factor in the mixing time due to the isoperimetry (m for the log barrier, $\tilde{O}(n)$ for the LS barrier). As far as we know, this factor might in fact be as small as $O(1)$. We make this precise via the following generalization of the KLS hyperplane conjecture. If true, it would directly improve our mixing time bound to $\tilde{O}\left(n^{\frac{3}{4}}\right)$.

Conjecture 14.1.2. *For any Hessian manifold M with metric d, let the isoperimetric ratio of a subset S w.r.t. d be defined as*

$$\psi_d(S) = \inf_{\varepsilon>0} \frac{\mathrm{vol}(\{x \in K \setminus S, d(x,y) \le \varepsilon\})}{\varepsilon \cdot \min\{\mathrm{vol}(S), \mathrm{vol}(K \setminus S)\}}$$

and the isoperimetric ratio of d as $\psi = \inf_{S \subset K} \psi_d(S)$. Then there is a subset S defined as a halfspace intersected with K with$\psi_d(S) = O(\psi)$.

14.1.4 Outline

In the next section, we recall the basic definitions and properties of Riemannian geometry and Hessian manifolds. Following that, in Section 14.3, we derive the discrete-time geodesic walk, showing how the magic formula naturally arises. In Section 14.4, we prove the main mixing bound for the geodesic walk in terms of a set of manifold parameters. In subsequent sections, we bound these parameters for the logarithmic barrier, thus proving Theorem 14.1.1. The algorithm for solving ODEs (collocation method) is presented and analyzed in Section 14.6.

14.2 Preliminaries

Throughout the chapter, we use lowercase letter for vectors and vector fields and uppercase letter for matrices and tensors. We use e_k to denote coordinate vectors. We use $\frac{d}{dt}$ for the usual derivative, e.g. $\frac{df(c(t))}{dt}$ is the derivative of some function along a curve parametrized by t, ∇ for the connection (manifold derivative, defined below which takes into account the local metric), D_v for the directional derivative wrt to the vector (or vector field) v, and D_t if the parametrization is clear from the context. We use g for the local metric. Given a point $x \in M$, g is a matrix with entries g_{ij}. Its inverse has

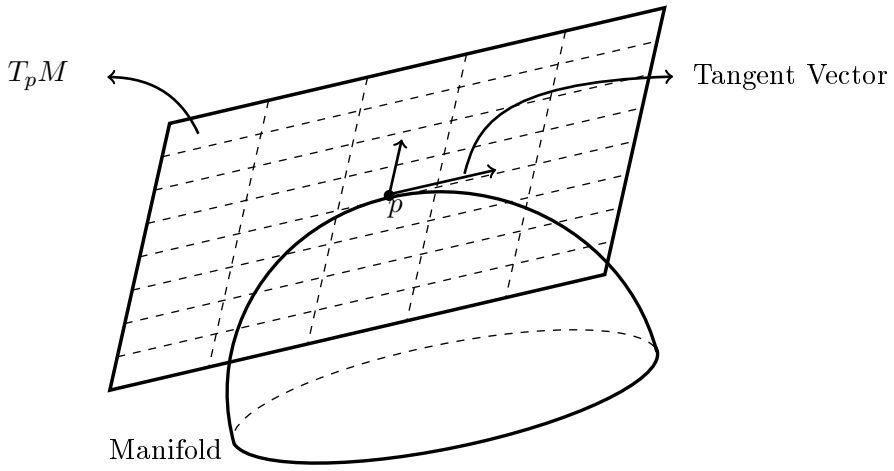


Figure 14-1: Riemannian Manifold and Tangent Space

entries g^{ij}. Also n is the dimension, m the number of inequalities, γ is a geodesic, and ϕ is a smooth convex function.

14.2.1 Basic Definitions of Riemannian geometry

Here we first introduce basic notions of Riemannian geometry. In this chapter, we use lowercase letter for vectors and vector fields and uppercase letter for matrices (this is not the convention used in Riemannian geometry). One can think of a manifold M as a n-dimensional "surface" in $\mathbb{R}^m$ for some $m \geq n$.

1. Tangent space T_pM: For any point p, the tangent space T_pM of M at point p is a linear subspace of $\mathbb{R}^m$. Intuitively, T_pM is the vector space of possible directions that are tangential to the manifold at x. Equivalently, it can be thought as the first-order linear approximation of the manifold M at p. For any curve c on M, the direction $\frac{d}{dt}c(t)$ is tangent to M and hence lies in $T_{c(t)}M$. When it is clear from context, we define $c'(t) = \frac{dc}{dt}(t)$. For any open subset M of $\mathbb{R}^n$, we can identify T_pM with $\mathbb{R}^n$ because all directions can be realized by derivatives of some curves in $\mathbb{R}^n$.

2. Riemannian metric: For any $v, u \in T_pM$, the inner product (Riemannian metric) at p is given by $\langle v, u \rangle_p$ and this allows us to define the norm of a vector $\|v\|_p = \sqrt{\langle v, v \rangle_p}$. We call a manifold a Riemannian manifold if it is equipped with a Riemannian metric. When it is clear from context, we define $\langle v, u \rangle = \langle v, u \rangle_p$. In $\mathbb{R}^n$, $\langle v, u \rangle_p$ is the usual ℓ_2 inner product.

3. Pushforward d: Given a function f from a manifold M to a manifold N, we define $df(x)$ as the linear map from T_xM to $T_{f(x)}N$ such that

$$df(x)(c'(0)) = (f \circ c)'(0)$$

for any curve c on M starting at $x = c(0)$. When M and N are Euclidean spaces, $df(x)$ is the Jacobian of f at x.

4. Hessian manifold: We call M a Hessian manifold if M is an open subset of $\mathbb{R}^n$ with the Rieman-

nian metric at a point $p \in M$ defined by

$$\langle v, u \rangle_p = v^T \nabla^2 \phi(p) u$$

where $v, u \in T_p M$ and ϕ is a smooth convex function on M.

5. Length: For any curve $c : [0, 1] \to M$, we define its length by

$$L(c) = \int_0^1 \| \frac{d}{dt} c(t) \|_{c(t)} dt.$$

6. Distance: For any $x, y \in M$, we define $d(x, y)$ be the infimum of the lengths of all paths connecting x and y. In $\mathbb{R}^n$, $d(x, y) = \|x - y\|_2$.

7. Geodesic: We call a curve $\gamma(t) : [a, b] \to M$ a geodesic if

   (a) The curve $\gamma(t)$ is parameterized with constant velocity. Namely, there is some c such that $\| \frac{d}{dt} \gamma(t) \|_{\gamma(t)} = c$ for all $a \leq t \leq b$.

   (b) The curve is the locally shortest length curve between $\gamma(a)$ and $\gamma(b)$. Namely, for any family of curve $c(t, s)$ with $c(t, 0) = \gamma(t)$ and $c(0, a) = \gamma(a)$ and $c(0, b) = \gamma(b)$, we have that $\frac{d}{ds}\big|_{s=0} \int_a^b \| \frac{d}{dt} c(t, s) \|_{c(t,s)} dt = 0$.

   Note that, if $\gamma(t)$ is a geodesic, then $\gamma(\alpha t)$ is a geodesic for any α. Intuitively, geodesics are local shortest-path curves. In $\mathbb{R}^n$, geodesics are straight lines.

8. Exponential map: The map $exp_p : T_p M \to M$ is defined as

$$exp_p(v) = \gamma_v(1)$$

   where γ_v is the unique geodesic starting at p with initial velocity $\gamma_v'(0)$ equals to v. The exponential map takes a straight line $tv \in T_p M$ to a geodesic $\gamma_{tv}(1) = \gamma_v(t) \in M$. Intuitively, the exponential map can be thought as vector addition in a manifold. In $\mathbb{R}^n$, we have $\exp_p(v) = p + v$.

9. Parallel transport: Given any geodesic $c(t)$ and a vector v such that $\langle v, c'(0) \rangle_{c(0)} = 0$, we define the parallel transport of v along $c(t)$ by the following process: Take h to be infinitesimally small and $v_0 = v$. For $i = 1, 2, \cdots, 1/h$, we let v_{ih} be the vector orthogonal to $c'(ih)$ that minimizes the distance between $\exp_{c(ih)}(hv_{ih})$ and $\exp_{c((i-1)h)}(hv_{(i-1)h})$. Intuitively, the parallel transport finds the vectors on the curve such that their end points are closest to the end points of v. For general vector $v \in T_{c'(0)}$, we write $v = \alpha c'(0) + w$ and we define the parallel transport of v along $c(t)$ is the sum of $\alpha c'(t)$ and the parallel transport of w along $c(t)$. For non-geodesic curve, see the definition in Fact 14.2.1.

10. Orthonormal frame: Given a vector fields $v_1, v_2, \cdots, v_n$ on a subset of M, we call $\{v_i\}_{i=1}^n$ is an orthonormal frame if $\langle v_i, v_j \rangle_x = \delta_{ij}$ for all x. Given a curve $c(t)$ and an orthonormal frame at $c(0)$, we can extend it on the whole curve by parallel transport and it remains orthonormal on the whole curve.

11. Directional derivatives and Levi-Civita connection: Given any vector $v \in T_p M$ and a vector field u in a neighborhood of p. Let γ_v is the unique geodesic starting at p with initial velocity $\gamma_v'(0) = v$, we define

$$\nabla_v u = \lim_{h \to 0} \frac{u(h) - u(0)}{h}$$

where $u(h)$ is the parallel transport of $u(\gamma(h))$ from $\gamma(h)$ to $\gamma(0)$. Intuitively, Levi-Civita connection is the directional derivative of u along direction v, *taking the metric into account.* In particular, for $\mathbb{R}^n$, we have $\nabla_v u(x) = \frac{d}{dt}u(x+tv)$. When u is defined on a curve c, we define $D_t u = \nabla_{c'(0)} u$. In $\mathbb{R}^n$, we have $D_t u(\gamma(t)) = \frac{d}{dt}u(\gamma(t))$. We reserve $\frac{d}{dt}$ for the usual derivative with Euclidean coordinates.

We list some basic facts about the definitions introduced above that are useful for computation and intuition.

Fact 14.2.1. *Given a manifold M, a curve $c(t) \in M$, a vector v and vector fields u, w on M, we have the following:*

1. *(alternative definition of parallel transport) $v(t)$ is the parallel transport of v along $c(t)$ if and only if $\nabla_{c'(t)} v(t) = 0$.*
2. *(alternative definition of geodesic) c is a geodesic if and only if $\nabla_{c'(t)} c'(t) = 0$.*
3. *(linearity) $\nabla_v(u+w) = \nabla_v u + \nabla_v w$.*
4. *(product rule) For any scalar-valued function f, $\nabla_v(f \cdot u) = \frac{\partial f}{\partial v} u + f \cdot \nabla_v u$.*
5. *(metric preserving) $\frac{d}{dt}\langle u, w\rangle_{c(t)} = \langle D_t u, w\rangle + \langle u, D_t w\rangle$.*
6. *(torsion free-ness) For any map $c(t,s)$ from a subset of $\mathbb{R}^2$ to M, we have that $D_s \frac{\partial c}{\partial t} = D_t \frac{\partial c}{\partial s}$ where $D_s = \nabla_{\frac{\partial c}{\partial s}}$ and $D_t = \nabla_{\frac{\partial c}{\partial t}}$.*
7. *(alternative definition of Levi-Civita connection) $\nabla_v u$ is the unique linear mapping from the product of vector and vector field to vector field that satisfies (3), (4), (5) and (6).*

14.2.1.1 Curvature

Here, we define various notions of curvature. Roughly speaking, they measure the amount by which a manifold deviates from Euclidean space.

Given vector $u, v \in T_p M$, in this section, we define uv be the point obtained from moving from x along direction u with distance $\|u\|_p$ (using geodesic), then moving along direction "v" with distance $\|v\|_p$ where "v" is the parallel transport of v along the path u. In $\mathbb{R}^n$, uv is exactly $p+u+v$ and hence $uv = vu$, namely, parallelograms close up. For a manifold, parallelograms almost close up, namely, $d(uv, vu) = o(\|u\|\|v\|)$. This property is called being *torsion-free.*

1. Riemann curvature tensor: Three-dimensional parallelepipeds might not close up, and the curvature tensor measures how far they are from closing up. Given vector $u, v, w \in T_p M$, we define uvw as the point obtained from moving from uv along direction "w" for distance $\|w\|_p$ where "w" is the parallel transport of w along the path uv. In a manifold, parallelepipeds do not close up and the Riemann curvature tensor how much uvw deviates from vuw. Formally, for vector fields v, w, we define $\tau_v w$ be the parallel transport of w along the vector field v for one unit of time. Given vector field v, w, u, we define the Riemann curvature tensor by

$$R(u,v)w = \frac{d}{ds}\frac{d}{dt}\tau_{su}^{-1}\tau_{tv}^{-1}\tau_{su}\tau_{tv} w\Big|_{t,s=0}. \tag{14.4}$$

   Riemann curvature tensor is a tensor, namely, $R(u,v)w$ at point p depends only on $u(p)$, $v(p)$ and $w(p)$.

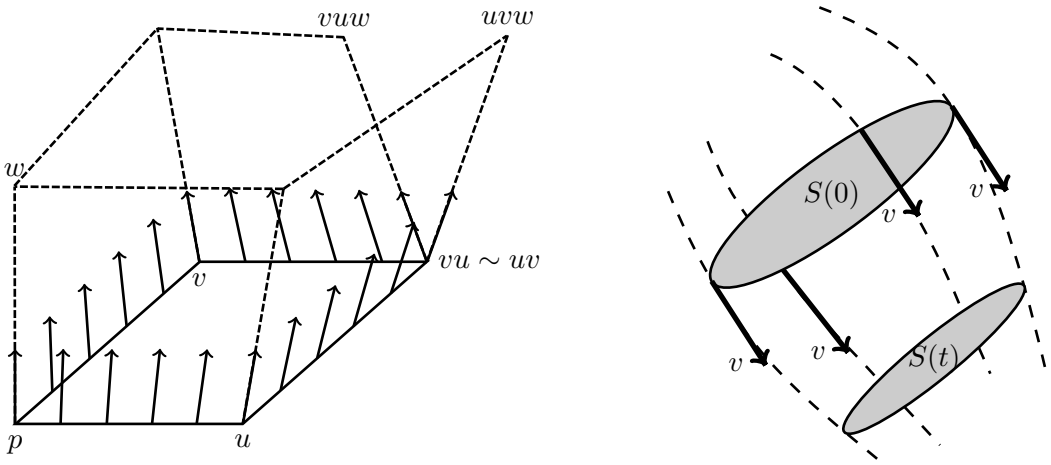


Figure 14-2: Riemann curvature tensor measures the deviation of parallelepipeds (14.4) and Ricci curvature measures the change of volumes (14.5).

2. Ricci curvature: Given a vector $v \in T_pM$, the Ricci curvature $\mathrm{Ric}(v)$ measures if the geodesics starting around p with direction v converge together. Positive Ricci curvature indicates the geodesics converge while negative curvature indicates they diverge. For example, let $S(0)$ be a small shape around p and $S(t)$ be the set of point obtained from moving $S(0)$ along direction v with t unit of time. Then,

$$\mathrm{vol}S(t) = \mathrm{vol}S(0)(1 - \frac{t^2}{2}\mathrm{Ric}(v) + \text{smaller terms}). \tag{14.5}$$

Formally, we define

$$\mathrm{Ric}(v) = \sum_{u_i} \langle R(v, u_i)u_i, v\rangle$$

where u_i is an orthonormal basis of T_pM. Equivalently, we have $\mathrm{Ric}(v) = \mathbf{E}_{u\sim N(0,I)} \langle R(v, u)u, v\rangle$. For $\mathbb{R}^n$, $\mathrm{Ric}(v) = 0$. For a sphere in $n+1$ dimension with radius r, $\mathrm{Ric}(v) = \frac{n-1}{r^2}\|v\|^2$.

Fact 14.2.2 (Alternative definition of Riemann curvature tensor). *Given vector fields x, y, z on M. Then, we have*

$$R(x, y)z = \nabla_x\nabla_y z - \nabla_y\nabla_x z - \nabla_{[x,y]}z$$

where $[x, y]$ is the vector field such that $\nabla_{[x,y]}f = \nabla_x\nabla_y f - \nabla_y\nabla_x f$ for any smooth scalar function f.

Given any M-valued function $c(t, s)$, we have vector fields $\frac{\partial c}{\partial t}$ and $\frac{\partial c}{\partial s}$ on M. Then, we have that

$$R(\frac{\partial c}{\partial t}, \frac{\partial c}{\partial s})z = \nabla_{\frac{\partial c}{\partial t}}\nabla_{\frac{\partial c}{\partial s}} z - \nabla_{\frac{\partial c}{\partial s}}\nabla_{\frac{\partial c}{\partial t}} z.$$

Sometimes, we simply write it as $R(\partial_t c, \partial_s c)z = D_t D_s z - D_s D_t z$.

Fact 14.2.3. *Given vector fields v, u, w, z on M. Then, we have*

$$\langle R(v, u)w, z\rangle = \langle R(w, z)v, u\rangle = -\langle R(u, v)w, z\rangle = -\langle R(v, u)z, w\rangle .$$

14.2.1.2 Jacobi field

Let $c : [0, \ell] \to M$ be a geodesic and $c(t, s)$ be a variation of $c(t)$ (i.e. $c(t, 0) = c(t)$) such that $c_s(t) \stackrel{\text{def}}{=} c(t, s)$ is a geodesic for all s. Then, $x(t) \stackrel{\text{def}}{=} \frac{\partial}{\partial s} c(t, s)\big|_{s=0}$ satisfies the following equation

$$D_t D_t x + R(x, \frac{dc}{dt})\frac{dc}{dt} = 0$$

where $R(\cdot, \cdot)\cdot$ is Riemann curvature tensor defined before. We call any vector field satisfying the equation above a *Jacobi field*. In $\mathbb{R}^n$, geodesics are straight lines and Jacobi field is linear, namely, $x(t) = x(0) + x'(0)t$.

Fact 14.2.4. *Every Jacobi field x can be split into the tangential part x_1 and the normal part x_2 such that*

1. $x = x_1 + x_2$,
2. *x_1 and x_2 are Jacobi fields, namely, $D_t D_t x_1 + R(x_1, \frac{dc}{dt})\frac{dc}{dt} = 0$ and $D_t D_t x_2 + R(x_2, \frac{dc}{dt})\frac{dc}{dt} = 0$,*
3. *x_1 is parallel to $\frac{dc}{dt}$ and is linear, namely, $x_1(t) = \left(\left\langle x(0), \frac{dc}{dt}(0) \right\rangle_{c(0)} + \left\langle D_t x(0), \frac{dc}{dt}(0) \right\rangle_{c(0)} t \right) \frac{dc}{dt}(t)$,*
4. *x_2 is orthogonal to $\frac{dc}{dt}$, namely, $\left\langle x_2(t), \frac{dc}{dt}(t) \right\rangle_{c(t)} = 0$.*

14.2.1.3 Hessian manifolds

Recall that a manifold is called Hessian if it is a subset of $\mathbb{R}^n$ and its metric is given by $g_{ij} = \frac{\partial^2}{\partial x^i \partial x^j}\phi$ for some smooth convex function ϕ. We let g^{ij} be entries of the inverse matrix of g_{ij}. For example, we have $\sum_j g^{ij} g_{jk} = \delta_{ik}$. We use ϕ_{ij} to denote $\frac{\partial^2}{\partial x^i \partial x^j}\phi$ and ϕ_{ijk} to denote $\frac{\partial^2}{\partial x^i \partial x^j \partial x^k}\phi$.

Since Hessian manifold is a subset of Euclidean space, we identify tangent spaces $T_p M$ by Euclidean coordinates. The following lemma give formulas for Levi-Civita connection and curvature under the Euclidean coordinates.

Lemma 14.2.5 ([250])**.** *Given a Hessian manifold M, vector fields v, u, w, z on M, we have the following:*

1. *(Levi-Civita connection) $\nabla_v u = \sum_{ik} v_i \frac{\partial u_k}{\partial x_i} e_k + \sum_{ijk} v_i u_j \Gamma_{ij}^k e_k$ where e_k are coordinate vectors and the Christoffel symbol*
$$\Gamma_{ij}^k = \frac{1}{2} \sum_l g^{kl} \phi_{ijl}$$
.
2. *(Riemann curvature tensor) $\langle R(u, v)w, z \rangle = \sum_{ijlk} R_{klij} u_i v_i w_l z_k$ where*
$$R_{klij} = \frac{1}{4} \sum_{pq} g^{pq} \left(\phi_{jkp}\phi_{ilq} - \phi_{ikp}\phi_{jlq} \right).$$
3. *(Ricci curvature) $Ric(v) = \frac{1}{4} \sum_{ijlkpq} g^{pq} g^{jl} \left(\phi_{jkp}\phi_{ilq} - \phi_{ikp}\phi_{jlq} \right) v^i v^k$.*

In this chapter, geodesic is used everywhere and we will implicitly using the following lemma in all of our calculations.

Lemma 14.2.6 ([214, Cor 3.1]). *If ϕ is a self-concordant function, namely, $\left|D^3 f(x)[h,h,h]\right| \leq 2\left|D^2 f(x)[h,h]\right|^{3/2}$, then the corresponding Hessian manifold M is geodesically complete, namely, for any $p \in M$, the exponential map is defined on the entire tangent space $T_p M$ and for any two points $p, q \in M$, there is a length minimizing geodesic connecting them.*

In particular, for a polytope $M = \{x : Ax > b\}$, the Hessian manifold induced by the function $\phi(x) = -\sum_i \log(a_i^T x - b_i)$ is geodesically complete.

14.2.1.4 Normal coordinates

For any manifold M, and any $p \in M$, the exponential map $\exp_x$ maps from $T_x M$ to M. Since $T_x M$ is isomorphic to $\mathbb{R}^n$, $\exp_x^{-1}$ gives a local coordinate system of M around x. We call this system the normal coordinates at x. In a normal coordinate system, the metric is locally constant.

Lemma 14.2.7. *In normal coordinates, we have*

$$g_{ij} = \delta_{ij} - \frac{1}{3}\sum_{kl} R_{ikjl} x^k x^l + O(|x|^3).$$

For a Hessian manifold, one can do a linear transformation on the normal coordinate to make this coincide with Euclidean coordinate up to the first order.

Lemma 14.2.8. *Given a Hessian manifold M and any point $x \in M$. We pick a basis of $T_x M$ such that*

$$\exp_x(tv) = x + tv + O(t^2).$$

Let a local coordinate system $F : M \to \mathbb{R}^n$ defined by $F(y) = \exp_x^{-1}$. Then, we have

$$DF[h] = h \text{ and } D^2 F_k(x)[h,h] = h^T \Gamma^k h$$

where F_k is the k^{th} coordinate of F and Γ^k is the matrix with entries Γ^k_{ij} defined in Lemma 14.2.5.

14.2.2 Stochastic calculus

A stochastic differential equation (SDE) describes a stochastic process over a domain Ω. It has the form $dx_t = \mu(x_t, t)dt + \sigma(x_t, t)dW_t$ where x_t is the current point at time t, W_t is a standard Brownian motion, and $\mu(x_t, t), \sigma(x_t, t)$ are the mean and covariance of the next infinitesimal step at time t.

Lemma 14.2.9 (Itō's lemma). *Given a SDE $dx_t = \mu(x_t)dt + \sigma(x_t)dW_t$ and any smooth function f, we have*

$$df(t, x_t) = \left\{\frac{\partial f}{\partial t} + \langle \nabla f, \mu\rangle + \frac{1}{2}\mathrm{Tr}\left[\sigma^T\left(\nabla^2 f\right)\sigma\right]\right\} dt + (\nabla f)^T \sigma dW_t.$$

SDEs are closely related to diffusion equations:

$$\frac{\partial}{\partial t}p(x,t) = \frac{1}{2}\nabla \cdot (A(x,t)\nabla p(x,t))$$

where $p(x,t)$ is the density at point x and time t, $\nabla\cdot$ is the usual divergence operator, ∇p is the gradient of p and the matrix $A(x,t)$ represents the diffusion coefficient at point x and time t. When $A(x,t) = I$, we get the familiar heat equation:

$$\frac{\partial}{\partial t}p(x,t) = \frac{1}{2}\Delta p(x,t).$$

In this chapter, the diffusion coefficient will be a symmetric positive definite matrix given by the Hessian $\left(\nabla^2\phi(x)\right)^{-1}$of a convex function $\phi(x)$.

The Fokker-Planck equation connects an SDE to a diffusion equation.

Theorem 14.2.10 (Fokker-Planck equation). *For any stochastic differential equation (SDE) of the form*

$$dx_t = \mu(x_t, t)dt + \sqrt{A(x_t, t)}dW_t,$$

the probability density of the SDE is given by the diffusion equation

$$\frac{\partial}{\partial t}p(x,t) = -\sum_{i=1}^{n}\frac{\partial}{\partial x_i}[\mu_i(x,t)p(x,t)] + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\frac{\partial^2}{\partial x_i x_j}[A_{ij}(x,t)p(x,t)].$$

14.2.3 Complex analysis

A complex function is said to be *(complex) analytic* (equivalently, *holomorphic*) if it is locally defined by a convergent power series. Hartog's theorem shows that a complex function in several variables $f : \mathbb{C}^n \to \mathbb{C}$ is holomorphic iff it is analytic in each variable (while fixing all the other variables). For any power series expansion, we define the radius of convergence at x as the largest number r such that the series converges on the sphere with radius r centered at x. Roughly speaking, complex analytic functions behave very nicely up to the radius of convergence, and one can avoid complex and tedious computations via general convergence theorems.

Theorem 14.2.11 (Cauchy's Estimates). *Suppose f is holomorphic on a neighborhood of the ball $B \stackrel{\text{def}}{=} \{z \in \mathbb{C} \ : \ |z - z_0| \leq r\}$, then we have that*

$$\left|f^{(k)}(z_0)\right| \leq \frac{k!}{r^k}\sup_{z\in B}|f(z)|.$$

In particular, for any rational function $f(z) = \frac{\prod_{i=1}^{\alpha}(z-a_i)}{\prod_{j=1}^{\beta}(z-b_j)}$ and any ball $B \stackrel{\text{def}}{=} \{z \in \mathbb{C} \ : \ |z - z_0| \leq r\}$ such that $b_j \notin B$, we have that

$$\left|f^{(k)}(z_0)\right| \leq \frac{k!}{r^k}\sup_{z\in B}|f(z)|.$$

A similar estimate holds for analytic functions in several variables.

14.3 Intuition

In this section, we derive the formula of the geodesic walk from first principles.

14.3.1 Derivation of the Geodesic walk

Given a smooth convex function ϕ on the convex domain M, namely that it is convex and is infinitely differentiable at every interior point of M, we consider the corresponding diffusion equation

$$\frac{\partial}{\partial t}p(x,t) \quad = \quad \frac{1}{2}\nabla\cdot\left(\nabla^2\phi\right)^{-1}\nabla p.$$

We can expand it by

$$\begin{aligned}\frac{\partial}{\partial t}p(x,t) &= \frac{1}{2}\sum_{i=1}^{n}\frac{\partial}{\partial x_i}\left(\sum_{j=1}^{n}\left(\left(\nabla^2\phi\right)^{-1}\right)_{ij}\frac{\partial}{\partial x_j}p(x,t)\right)\\ &= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\frac{\partial^2}{\partial x_i x_j}\left(\left(\left(\nabla^2\phi\right)^{-1}\right)_{ij}p(x,t)\right) - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\frac{\partial}{\partial x_i}\left(\frac{\partial}{\partial x_j}\left(\left(\nabla^2\phi\right)^{-1}\right)_{ij}p(x,t)\right).\end{aligned}$$

The uniform distribution is the stationary distribution of this diffusion equation. Now applying the Fokker-Planck equation (Theorem 14.2.10) with $A = \left(\nabla^2\phi\right)^{-1}$, the SDE for the above diffusion is given by:

$$dx_t = \mu(x_t)dt + \left(\nabla^2\phi(x_t)\right)^{-1/2}dW_t.$$

This explains the definition of (14.1). To simplify the notation, we write the SDE as

$$dx_t = \mu(x_t)dt + \sigma(x_t)dW_t \tag{14.6}$$

where $\sigma(x_t) = \left(\nabla^2\phi(x_t)\right)^{-1/2}$. One way to simulate this is via the Euler–Maruyama method, namely

$$x_{(t+1)h} = x_{th} + \mu(x_{th})h + \sigma(x_{th})w_{th}\sqrt{h}$$

where $w_{th} \sim N_{x_{th}}(0, I)$. We find the direction we are heading and take a small step along that direction. However, if we view M as a manifold, then directly adding the direction $\mu(x_{th})h + \sigma(x_{th})w_{th}\sqrt{h}$ to x_{th} is not natural; the Euclidean coordinate is just an arbitrary coordinate system and we could pick any other coordinate systems and add the direction into x_{th}, giving a different step. Instead, we take the step in normal coordinates (Section 14.2.1.4).

In particular, given an initial point x_0, we define $F = \exp_{x_0}^{-1}$ and we note that $F(x_t)$ is another SDE. To see the defining equation of this transformed SDE, we use Itō's lemma to show that the transformed SDE looks the same but with half the drift term. This explains the formulation of geodesic walk: $x^{(j+1)} = \exp_{x^{(j)}}(\sqrt{h}w + \frac{h}{2}\mu(x^{(j)}))$.

Lemma 14.3.1. *Let $F = \exp_{x_0}^{-1}$ and x_t satisfies the SDE (14.6) Then we have*

$$dF(x_0) = \frac{1}{2}\mu(x_0)dt + \sigma(x_0)dW_0. \tag{14.7}$$

Proof. Itō's lemma (Lemma 14.2.9) shows that

$$dF_k(x_t) = \left\{\langle\nabla F_k, \mu\rangle + \frac{1}{2}\mathrm{Tr}\left[\sigma^T\left(\nabla^2 F_k\right)\sigma\right]\right\}dt + \left(\nabla F_k\right)^T\sigma dW_t$$

where F_k indicates the k^{th} coordinate of F. From Lemma 14.2.8, we have that $\langle\nabla F_k(x_0), \mu\rangle = \mu_k$, $\left(\nabla F_k(x_0)\right)^T\sigma = e_k^T\sigma$ and

$$\begin{aligned}\mathrm{Tr}\left[\sigma(x_0)^T\left(\nabla^2 F_k(x_0)\right)\sigma(x_0)\right] &= \sum_i D^2F_k[\sigma e_i, \sigma e_i]\\ &= \sum_i e_i^T\sigma^T\Gamma^k\sigma e_i\\ &= \mathrm{Tr}\left(\sigma^T\Gamma^k\sigma\right)\\ &= \sum e_i^T\Gamma^k\left(\nabla^2\phi\right)^{-1}e_i.\end{aligned}$$

Now, using Lemma 14.2.5, we have that

$$\mathrm{Tr}\left[\sigma(x_0)^T\left(\nabla^2 F_k(x_0)\right)\sigma(x_0)\right] = \frac{1}{2}\sum_{ijl} g^{kl}\phi_{ijl}g^{ji}$$

Hence, we have that

$$dF_k(x_0) = \left\{\mu_k + \frac{1}{4}\sum_{ijl} g^{kl}\phi_{ijl}g^{ji}\right\}dt + e_k^T\sigma dW_t$$

Recall that the drift term (14.2) is given by

$$\begin{aligned}
\mu_k &= \frac{1}{2}\sum_{i=1}^{n}\frac{\partial}{\partial x_i}\left(\left(\nabla^2\phi\right)^{-1}\right)_{ki} \\
&= -\frac{1}{2}\sum_{i} e_k^T\left(\nabla^2\phi\right)^{-1}\frac{\partial}{\partial x_i}\nabla^2\phi\left(\nabla^2\phi\right)^{-1}e_i \\
&= -\frac{1}{2}\sum_{l,i,j} g^{kl}\frac{\partial}{\partial x_i}\phi_{lj}g^{ji} \\
&= -\frac{1}{2}\sum_{ijl} g^{kl}\phi_{ijl}g^{ji}. \qquad (14.8)
\end{aligned}$$

Therefore, we have the result. □

■ 14.3.2 Discussion of Geodesic Walk

In this section, we relate our geodesic walk with the following commonly-used method for solving SDE.

Theorem 14.3.2 (Milstein Method). *Given a SDE $dx_t = \mu(x_t)dt + \sigma(x_t)dW_t$, consider the algorithm*

$$\overline{x}_h = x_0 + \mu(x_0)h + \sigma(x_0)\Delta W + \sum_{j_1,j_2=1}^{m} L^{j_1}\sigma^{k,j_2}I_{j_1j_2}$$

where $L^j = \sum_{k=1}^{d}\sigma^{k,j}\frac{\partial}{\partial x^k}$, $\Delta W_j \sim \int_0^h dW_t^j$ and $I_{j_1j_2} \sim \int_0^h\int_0^{t_1} dW_{t_2}^{j_2}dW_{t_1}^{j_1}$. Then, we have

$$\mathbf{E}\|\overline{x}_h - x_h\|_2 = O(h^2)$$

where the $O(.)$ notation hides constants related to anything except h.

Under normal coordinates, the metric is locally constant (Lemma 14.2.7). Due to this, the term $\sum_{j_1,j_2=1}^{m} L^{j_1}\sigma^{k,j_2}I_{j_1j_2}$ in the Milstein method vanishes. Hence, we have the following result.

Theorem 14.3.3 (Geodesic Walk). *Let X_t satisfies the SDE (14.6). Consider the discrete step*

$$\overline{x}_h = \exp_{x_0}\left(\frac{1}{2}\mu(x_0)h + \sigma(x_0)\Delta W\right)$$

where $\Delta W \sim N(0, hI)$. Then,

$$\mathbf{E}\|\overline{x}_h - x_h\|_2 = O(h^2)$$

where the $O(.)$ notation hides constants related to anything except h.

Proof. In normal coordinates, we consider the step

$$\overline{F(x_h)} = \frac{1}{2}\mu(x_0)h + \sigma(x_0)\Delta W.$$

Since $F(x_h)$ satisfies 14.7 and σ is locally constant at X_0 (Lemma 14.2.7), Theorem 14.3.2 (Milstein Method) shows that

$$\|\overline{F(x_h)} - F(x_h)\|_2 = O(h^2).$$

The result follows from $\overline{x}_h = \exp_{X_0}(\overline{F(x_h)})$, $x_h = \exp_{X_0}(F(x_h))$ and the smoothness of $\exp_{x_0}$. □

For the Euler-Maruyama method, $\overline{x}_h = x_0 + \mu(x_0)h + \sigma(x_0)\Delta W$, we only get $O(h^{1.5})$ as the bound on the error, rather than $O(h^2)$. This is one of the reasons we use geodesic instead of just adding up the vector in Euclidean coordinates.

■ 14.4 Convergence of the Geodesic Walk

The geodesic walk is a Metropolis-filtered Markov chain, whose stationary distribution is the uniform distribution over the polytope to be sampled. We will prove that the conductance of this chain is large with an appropriate choice of the step-size parameter. Therefore, its mixing time to converge to the stationary distribution will be small. The proof of high conductance involves showing (a) the acceptance probability of the Metropolis filter is at least a constant (b) the induced metric satisfies a strong isoperimetric inequality (c) two points that are close in metric distance are also close in probabilistic distance, namely, the one-step distributions from them have large overlap. Besides bounding the number of steps, we also have to show that each step of the Markov chain can be implemented efficiently. We describe the implementation via reduction to solving ODEs in Section 14.4.6. We do this in later sections via an efficient algorithm for approximately solving ODEs.

In this section, we present the general conductance proof for Hessian manifolds. The proof will need bounds on seven parameters determined by the specific barrier function. In Section 14.5, we will bound these parameters for the logarithmic barrier.

For a Markov chain with state space M, stationary distribution Q and next step distribution $P_u(\cdot)$ for any $u \in M$, the conductance of the Markov chain is

$$\phi = \inf_{S \subset M} \frac{\int P_u(M \setminus S) dQ(u)}{\min\{Q(S), Q(M \setminus S)\}}.$$

The conductance of an ergodic Markov chain allows us to bound its mixing time, i.e., the rate of convergence to its stationary distribution, e.g., via the following theorem of Lovász and Simonovits.

Theorem 14.4.1 ([177]). *Let Q_t be the distribution of the current point after t steps of a Markov chain with stationary distribution Q and conductance at least ϕ, starting from initial distribution Q_0. Then,*

$$d_{TV}(Q_t, Q) \leq \sqrt{d_0}\left(1 - \frac{\phi^2}{2}\right)^t$$

where $d_0 = \mathbf{E}_{Q_0}(dQ_0(u)/dQ(u))$ is a measure of the distance of the starting distribution from the stationary and d_{TV} is the total variation distance.

14.4.1 Hessian parameters

The mixing of the walk depends on the maximum values of several smoothness parameters of the manifold. For some of them, we only need to bound the parameters with high probability over the choice of steps of the geodesic walk. For $x \in M$, let $\widetilde{\Pi}_x$ be the set of geodesics used in one step of the geodesic walk starting at x with the parameterization $\gamma : [0, \ell] \to M$ and $\ell = \sqrt{nh}$. Parameters D_1, G_1, R_1, R_2 below will be bounded with high probability over the choice of the next step (the others hold for all points in the domain). Let Π_x denote a $1 - \exp(n^{-\Theta(1)})$ probability subset of $\widetilde{\Pi}_x$ and $\Pi = \bigcup_{x \in M} \Pi_x$. These will be fixed when bounding the parameters for specific manifolds.

1. The maximum norm of the drift, $D_0 = \sup_{x \in M} \|\mu(x)\|_x$.

2. The smoothness of drift norm, $D_1 = \sup_{\gamma \in \Pi, 0 \le t \le \ell} \frac{d}{dt} \|\mu(\gamma(t))\|^2$.

3. The smoothness of the drift, $D_2 = \sup_{x \in M, \|s\|_x \le 1} \|\nabla_s \mu(x)\|_x$.

4. The smoothness of the volume, $G_1 = \sup_{\gamma \in \Pi, 0 \le t \le \ell} |\log \det(g(\gamma(t)))'''|$ where $g(x)$ is the metric at x.

5. The smoothness of the metric, $G_2 = \sup \frac{d(x,y)}{d_H(x,y)}$ where d_H is the Hilbert distance defined in Section 14.4.3 and d is the shortest path distance in M.

6. The stability of the Jacobian field, $R_1 = \sup_{\gamma \in \Pi, 0 \le t \le \ell} \|R(t)\|_F$ where $R(t)_{ij} = \langle R(X_i, \gamma'(t))\gamma'(t), X_j \rangle$ where $\{X_i\}$ is any orthonormal frame at $\gamma(t)$.

7. The smoothness of the Ricci curvature, $R_2 = \sup_{\gamma \in \Pi} \left| \frac{d}{ds} Ric(\gamma'_s(t)) \right|$ (see Definition 14.4.16).

We refer to these as the smoothness parameters of a Hessian manifold. Our main theorem for convergence can be stated as follows.

Theorem 14.4.2. *On a Hessian manifold with smoothness parameters $D_0, D_1, D_2, G_1, G_2, R_1, R_2$ and assumption 14.4.23, the geodesic walk with step size*

$$0 < h \le \Theta(1) \min \left\{ \frac{1}{(nD_0R_1)^{2/3}}, \frac{1}{D_2}, \frac{1}{nR_1}, \frac{1}{n^{1/3}D_1^{2/3}}, \frac{1}{nG_1^{2/3}}, \frac{1}{(nR_2)^{2/3}} \right\}$$

has conductance $\Omega(\sqrt{h}/G_2)$ and its mixing time is $O(G_2^2/h)$.

14.4.2 1-step distribution

We derive a formula for the drift term — it is in fact a Newton step of the volumetric barrier function $\log \det \nabla^2 \phi(x)$.

Lemma 14.4.3. *We have*

$$\mu(x) = -\left(\nabla^2 \phi(x)\right)^{-1} \nabla \psi(x)$$

where $\psi(x) = \frac{1}{2} \log \det \nabla^2 \phi(x)$.

Proof. We note that

$$\begin{aligned}\frac{\partial}{\partial x_j}\log\det\left(\nabla^2\phi\right)^{-1} &= \mathrm{Tr}\left(\left(\nabla^2\phi\right)\frac{\partial}{\partial x_j}\left(\nabla^2\phi\right)^{-1}\right)\\ &= -\mathrm{Tr}\left(\left(\nabla^2\phi\right)\left(\nabla^2\phi\right)^{-1}\left(\frac{\partial}{\partial x_j}\nabla^2\phi\right)\left(\nabla^2\phi\right)^{-1}\right)\\ &= -\sum_k e_k^T\left(\frac{\partial}{\partial x_j}\nabla^2\phi\right)\left(\nabla^2\phi\right)^{-1}e_k.\end{aligned}$$

Hence, we have

$$\begin{aligned}\frac{1}{2}e_i^T\left(\nabla^2\phi\right)^{-1}\nabla\log\det\left(\nabla^2\phi\right)^{-1} &= -\frac{1}{2}\sum_{jk}\left(\nabla^2\phi\right)^{-1}_{ij}e_k^T\left(\frac{\partial}{\partial x_j}\nabla^2\phi\right)\left(\nabla^2\phi\right)^{-1}e_k\\ &= -\frac{1}{2}\sum_{jk}e_i^T\left(\nabla^2\phi\right)^{-1}\left(\frac{\partial}{\partial x_k}\nabla^2\phi\right)\left(\nabla^2\phi\right)^{-1}e_k\end{aligned}$$

On the other hand, we have

$$\begin{aligned}\mu_i &= \frac{1}{2}\sum_k\frac{\partial}{\partial x_k}\left(\left(\nabla^2\phi\right)^{-1}\right)_{ik}\\ &= -\frac{1}{2}\sum_k e_i^T\left(\nabla^2\phi\right)^{-1}\left(\frac{\partial}{\partial x_k}\nabla^2\phi\right)\left(\nabla^2\phi\right)^{-1}e_k.\end{aligned}$$

□

To have a uniform stationary distribution, the geodesic walk uses a Metropolis filter. The transition probability before applying the filter is given as follows.

Lemma 14.4.4. *For any $x\in M$ and $h>0$, the probability density of the 1-step distribution from x (before applying the Metropolis filter) is given by*

$$p_x(y)=\sum_{v_x:\exp_x(v_x)=y}\det(d\exp_x(v_x))^{-1}\sqrt{\frac{\det\left(g(y)\right)}{(2\pi h)^n}}\exp\left(-\frac{1}{2}\|\frac{v_x-\frac{h}{2}\mu(x)}{\sqrt{h}}\|_x^2\right)\tag{14.9}$$

where $y=\exp_x(v_x)$ and $d\exp_x$ is the differential of the exponential map at x.

Proof. We prove the formula by separately considering each $v_x\in T_xM$ s.t. $exp_x(v_x)=y$, then summing up. In the tangent space T_xM, the point v_x is a Gaussian step. Therefore, the probability density of v_x in T_xM as follows.

$$p_x^{T_xM}(v_x)=\frac{1}{(2\pi h)^{n/2}}\exp\left(-\frac{1}{2}\|\frac{v_x-\frac{h}{2}\mu(x)}{\sqrt{h}}\|_x^2\right).$$

Note that $v_x,\mu(x)\in T_xM$. Let $y=\exp_x(v_x)$. In the tangent space T_yM, we have that y maps to 0. Let $\phi:T_xM\to K$ defined by $F(v)=\mathrm{id}_{M\to K}\circ\exp_x(v)$. Here K is the same set as M but endowed with the Euclidean metric (we use F instead of $\exp_y^{-1}(\exp_x(v))$ since the exponential map might not be 1-1). Hence, we have

$$dF(v_x)=d\mathrm{id}_{M\to K}(y)d\exp_x(v_x).$$

The result follows from $p_x(y) = \det(dF(v_x))^{-1} p_x^{T_x M}(v_x)$ and

$$\begin{aligned} \det dF(v_x) &= \det\left(d\mathrm{id}_{M\to K}(y)\right) \det\left(d\exp_x(v_x)\right) \\ &= \det(g(y))^{-1/2} \det\left(d\exp_x(v_x)\right). \end{aligned}$$

□

In Section 14.4.5, we bound the acceptance probability of the Metropolis filter.

Lemma 14.4.5. *Under the condition on h of Theorem 14.4.2,*

$$\left|\log\left(\frac{p_x(y)}{p_y(x)}\right)\right| = O\left(\sqrt{n}h^{3/2}D_1 + (nh)^{3/2}G_1 + (nhR_1)^2\right) < \frac{1}{4}.$$

In Section 14.4.7, we bound the overlap of one-step distributions from nearby points.

Lemma 14.4.6. *Let $x, y \in M$ be such that $d(x,y) \le 0.1\sqrt{h}$. Then, under the condition on h of Theorem 14.4.2, the one-step distributions P_x, P_y from x, y satisfy*

$$d_{TV}(P_x, P_y) = O(nh^{3/2}R_2 + hD_2 + nh^{3/2}R_1D_0) < 0.3.$$

■ 14.4.3 Isoperimetry

For a convex body K, the *cross-ratio distance* of x and y is

$$d_K(x,y) = \frac{|x-y||p-q|}{|p-x||y-q|}$$

where p and q are on the boundary of K such that p, x, y, q are on the straight line $\overline{xy}$ and are in order. In this section, we show that if the distance $d(x,y)$ induced by the Riemannian metric is upper bounded by the cross-ratio distance, then the body has good isoperimetric constant in terms of d. We note that although the cross-ratio distance is not a metric, the closely-related Hilbert distance is a metric:

$$d_H(x,y) = \log\left(1 + \frac{|x-y||p-q|}{|p-x||y-q|}\right).$$

Theorem 14.4.7. *For a Hessian manifold M with smoothness parameters G_2, for any partition of M into three measurable subsets S_1, S_2, S_3, we have that*

$$\mathrm{vol}(S_3) \ge \frac{d(S_1,S_2)}{G_2} \min\{\mathrm{vol}(S_1), \mathrm{vol}(S_2)\}.$$

The theorem follows from the following isoperimetric inequality from [173], the definition of G_2 and the fact $d_H \le d_K$.

Theorem 14.4.8 ([173])**.** *For any convex body K and any partition of K into disjoint measurable subsets S_1, S_2, S_3*

$$\mathrm{vol}(S_3) \ge d_K(S_1,S_2)\mathrm{vol}(S_1)\mathrm{vol}(S_2).$$

14.4.4 Conductance and Mixing

Proof of Theorem 14.4.2. The proof follows the standard outline for geometric random walks (see e.g., [261]). Let Q be the uniform distribution over M and S be any measurable subset of M. Then our goal is to show that

$$\frac{\int_S P_x(M \setminus S)\, dQ(x)}{\min\{Q(S), Q(M \setminus S)\}} = \Omega\left(\frac{\sqrt{h}}{G_2}\right).$$

Since the Markov chain is time-reversible (For any two subsets A, B, $\int_A P_x(B)\, dx = \int_B P_x(A)\, dx$), we can write the numerator of the LHS above as

$$\frac{1}{2}\left(\int_S P_x(M \setminus S)\, dQ(x) + \int_{M \setminus S} P_x(S)\, dQ(x)\right).$$

Define

$$\begin{aligned} S_1 &= \{x \in S \,:\, P_x(M \setminus S) < 0.05\} \\ S_2 &= \{x \in M \setminus S \,:\, P_x(S) < 0.05\} \\ S_3 &= M \setminus S_1 \setminus S_2. \end{aligned}$$

We can assume wlog that $Q(S_1) \geq (1/2)Q(S)$ and $Q(S_2) \geq (1/2)Q(M \setminus S)$ (if not, the conductance is $\Omega(1)$).

Next, we note that for any two points $x \in S_1$ and $y \in S_2$, $d_{TV}(P_x, P_y) > 0.9$. Therefore, by Lemma 14.4.6, we have that $d(x, y) \geq 0.1\sqrt{h}$ and hence $d(S_1, S_2) \geq 0.1\sqrt{h}$. Therefore, using Theorem 14.4.7,

$$\mathrm{vol}(S_3) \geq \frac{0.1\sqrt{h}}{G_2} \min\{\mathrm{vol}(S_1), \mathrm{vol}(S_2)\}.$$

Going back to the conductance,

$$\begin{aligned} \frac{1}{2}\left(\int_S P_x(M \setminus S)\, dQ(x) + \int_{M \setminus S} P_x(S)\, dQ(x)\right) &\geq \frac{1}{2}\int_{S_3} (0.05) dQ(x) \\ &\geq \frac{\sqrt{h}}{800 G_2} \min\{\mathrm{vol}(S_1), \mathrm{vol}(S_2)\} \frac{1}{\mathrm{vol}(K)} \\ &= \frac{\sqrt{h}}{1600 G_2} \min\left\{\frac{\mathrm{vol}(S)}{\mathrm{vol}(M)}, \frac{\mathrm{vol}(M \setminus S)}{\mathrm{vol}(M)}\right\} \\ &= \frac{\sqrt{h}}{1600 G_2} \min\{Q(S), Q(M \setminus S)\} \end{aligned}$$

Therefore, $\phi(S) \geq (1/1600)\sqrt{h}/G_2$. □

Corollary 14.4.9. *Let K be a polytope with m facets. Let Q be the uniform distribution over K and Q_t be the distribution obtained after t steps of the geodesic walk started from a distribution Q_0 with $d_0 = \sup_K \frac{dQ_0}{dQ}$. Then after $t > C(G_2^2/h) \log\left(\frac{d_0}{\varepsilon}\right)$ steps, with probability at least $1 - \delta$, we have $d_{TV}(Q_t, Q) \leq \varepsilon$.*

14.4.5 Rejection Probability

The goal of this section is to prove Lemma 14.4.5, i.e., the rejection probability of the Metropolis filter is small. For a transition from x to y on the manifold, the filter is applied with respect to a randomly chosen v_x, i.e., for one geodesic from x to y. We will bound the ratio of the transition probabilities (without the filter) as follows:

$$\log\left(\frac{p(x \overset{v_x}{\to} y)}{p(y \overset{v_y}{\to} x)}\right) = \log\left(\frac{\det\left(d\exp_x(v_x)\right)^{-1}}{\det\left(d\exp_y(v_y)\right)^{-1}}\right) + \frac{1}{2}\log\det\left(g(y)\right) - \frac{1}{2}\log\det\left(g(x)\right) - \frac{1}{2}\|\frac{v_x - \frac{h}{2}\mu(x)}{\sqrt{h}}\|_2^2 + \|\frac{v_y - \frac{h}{2}\mu(y)}{\sqrt{h}}\|_2^2.$$

Since a geodesic has constant speed, we have $\|v_x\| = \|v_y\|$. Therefore, we have that

$$\log\left(\frac{p(x \overset{v_x}{\to} y)}{p(y \overset{v_y}{\to} x)}\right) = \log\left(\frac{\det\left(d\exp_y(v_y)\right)}{\det\left(d\exp_x(v_x)\right)}\right) + \frac{1}{2}\log\det\left(g(y)\right) - \frac{1}{2}\log\det\left(g(x)\right) + \frac{1}{2}\langle v_x, \mu(x)\rangle - \frac{h}{8}\|\mu(x)\|^2 - \frac{1}{2}\langle v_y, \mu(y)\rangle + \frac{h}{8}\|\mu(y)\|^2. \tag{14.10}$$

We separate the proof into three parts:

- $\left|\|\mu(x)\|^2 - \|\mu(y)\|^2\right| \le O\left(\sqrt{nh}D_1\right)$, immediate from the definition of D_1 and the fact that the $\|x - y\| = O\left(\sqrt{nh}\right)$ whp.

- Sec 14.4.5.1: $|\log\det\left(g(y)\right) - \log\det\left(g(x)\right) + \langle v_x, \mu(x)\rangle - \langle v_y\mu(y)\rangle| = O((nh)^{3/2}G_1)$,

- Sec 14.4.5.2: Assuming that $h < \frac{1}{2nR_1}$, we have $\left|\log\left(\frac{\det\left(d\exp_y(v_y)\right)}{\det\left(d\exp_x(v_x)\right)}\right)\right| = O((nhR_1)^2)$.

Together, these facts imply the lemma.

14.4.5.1 Trapezoidal Rule for Volumetric Barrier

Recall that the Trapezoidal rule is to approximate $\int_0^h f(t)dt$ by $\frac{h}{2}\left(f(0) + f(h)\right)$. The nice thing about this rule is that the error is $O(h^3)$ instead of $O(h^2)$ because the second order term cancels by symmetry. Our main observation here is that the geodesic walk is implicitly following a trapezoidal rule on the metric and hence it has a small error.

Lemma 14.4.10. *We have that*

$$\left|\int_0^\ell f'(t)dt - \frac{\ell}{2}\left(f'(0) + f'(\ell)\right)\right| \le \frac{\ell^3}{12}\max_{0\le t\le\ell}\left|f'''(t)\right|.$$

Proof. Note that

$$\begin{aligned}
\int_0^\ell f'(t)dt - \frac{\ell}{2}\left(f'(0)+f'(\ell)\right) &= \int_0^\ell \left(f'(0)+\int_0^t f''(s)ds\right)dt - \ell f'(0) - \frac{\ell}{2}\int_0^\ell f''(s)ds \\
&= \int_0^\ell \int_0^t f''(s)dsdt - \frac{\ell}{2}\int_0^\ell f''(s)ds \\
&= \int_0^\ell (\frac{\ell}{2}-s)f''(s)ds \\
&= \int_0^\ell (\frac{\ell}{2}-s)\left(f''(0)+\int_0^s f'''(t)dt\right)ds \\
&= \int_0^\ell (\frac{\ell}{2}-s)\int_0^s f'''(t)dtds \\
&\le \frac{\ell^3}{12}\max_{0\le t\le \ell}\left|f'''(t)\right|.
\end{aligned}$$

□

We apply this to the logdet function.

Lemma 14.4.11. *Let* $f(t) = \log\det(g(\gamma(t))$ *where* $\gamma(t) = \exp_x(\frac{t}{\ell}v_x) \in \Pi$ *(See Definition of* Π *in Section 14.4.1). Then,*

$$\left|\log\det(g(y)) - \log\det(g(x)) + \langle v_x, \mu(x)\rangle_x - \langle v_y, \mu(y)\rangle_y\right| = O((nh)^{3/2}G_1).$$

Proof. Let $f(t) = \log\det g(\gamma(t))$ and $z = \gamma(t)$. By Lemma 14.4.3, $\mu(z) = -\frac{1}{2}g(z)^{-1}\nabla f(z)$. Using this,

$$\begin{aligned}
f'(t) &= \langle \nabla_z \log\det g(z), \gamma'(t)\rangle_2 \\
&= \langle g(z)^{-1}\nabla_z \log\det g(z), \gamma'(t)\rangle_z \\
&= -\langle 2\mu(z), \gamma'(t)\rangle_z.
\end{aligned}$$

Noting that $v_x = \ell\gamma'(0)$ and $v_y = -\ell\gamma'(\ell)$, and using Lemma 14.4.10, we have

$$\begin{aligned}
&\left|\log\det(g(y)) - \log\det(g(x)) + \langle v_x, \mu(x)\rangle_x - \langle v_y, \mu(y)\rangle_y\right| \\
=\ &\left|\log\det(g(y)) - \log\det(g(x)) + \ell\left(\langle\gamma'(0), \mu(x)\rangle_x + \langle\gamma'(\ell), \mu(y)\rangle_y\right)\right| \\
=\ &\left|\int_0^\ell f'(t)dt - \frac{\ell}{2}\left(f'(0)+f'(\ell)\right)\right| \\
\le\ &\frac{\ell^3}{12}\max_{0\le t\le\ell}\left|f'''(t)\right| = O\left((nh)^{3/2}G_1\right).
\end{aligned}$$

□

14.4.5.2 Smoothness of exponential map

First, we show the relation between the differential of exponential map $d\exp_x(v_x)$ and the Jacobi field along the geodesic $\exp_x(tv_x)$.

Lemma 14.4.12. *Given a geodesic $\gamma(t) = \exp_x(\frac{t}{\ell} v_x) \in \Pi$, let $\{X_i(t)\}_{i=1}^n$ be the parallel transport of some orthonormal frame along $\gamma(t)$. Then, for any $w \in \mathbb{R}^n$, we have that*

$$d\exp_x(v_x)\left(\sum w_i X_i(0)\right) = \sum_i \psi_w(\ell)_i X_i(\ell)$$

where ψ_w satisfies the Jacobi field along $\gamma(t)$, i.e.,

$$\begin{aligned} \frac{d^2}{dt^2}\psi_w(t) + R(t)\psi_w(t) &= 0, \\ \frac{d}{dt}\psi_w(0) &= w/\ell, \\ \psi_w(t) &= 0, \end{aligned} \tag{14.11}$$

and $R(t)$ is a matrix given by $R_{ij}(t) = \langle R(X_i(t), \gamma'(t))\gamma'(t), X_j(t)\rangle$.

Proof. We want to compute $d\exp_x(v_x)(w)$ for some $w \in T_y M$. By definition, we have that

$$d\exp_x(v_x)(w) = \frac{d}{ds}\gamma(t,s)|_{t=\ell, s=0}$$

where $\gamma(t,s) = \exp_x(tv_x/\ell + sw)$. Let $\eta_w(t) = \frac{d}{ds}\gamma(t,s)|_{s=0}$ be a Jacobi field given by the formula

$$\begin{aligned} D_t D_t \eta_w + R(\eta_w, \gamma'(t))\gamma'(t) &= 0, \text{ for } 0 \le t \le \ell \\ D_t \eta_w(0) &= w/\ell, \\ \dot{\eta}_w(0) &= 0. \end{aligned}$$

Recall that the parallel transport of an orthonormal frame remains orthonormal because $\frac{d}{dt}\langle X_i, X_j\rangle = \langle D_t X_i, X_j\rangle + \langle X_i, D_t X_j\rangle = 0$. Since $X_i(t)$ is an orthonormal basis at $T_{\gamma(t)}M$, we can write

$$\eta_w(t) = \sum_i \psi_i(t) X_i(t).$$

Since η_w satisfies the ODE above, we have

$$D_t D_t \sum_i \psi_i(t) X_i(t) + R(\sum_i \psi_i(t) X_i(t), \gamma'(t))\gamma'(t) = 0.$$

Since $X_i(t)$ is a parallel transport, we have that $D_t D_t\left(\psi_i(t) X_i(t)\right) = \frac{d^2\psi_i(t)}{dt^2} X_i(t)$ and hence

$$\sum_i \frac{d^2}{dt^2}\psi_i(t) X_i(t) + \sum_i \psi_i(t) R(X_i(t), \gamma'(t))\gamma'(t) = 0$$

Let $R_{ij}(t) = \langle R(X_i(t), \gamma'(t))\gamma'(t), X_j(t)\rangle$. Since $R(t)$ is a symmetric matrix (Fact 14.2.3), we have

$$\frac{d^2}{dt^2}\psi(t) + R(t)\psi(t) = 0.$$

Hence, we have

$$d\exp_x(v_x)(w) = \psi(\ell).$$

□

Next, we have a lemma about determinant.

Lemma 14.4.13. *Suppose that E is a matrix (not necessarily symmetric) with $\|E\|_F \leq \frac{1}{4}$, we have*

$$|\log\det(I+E) - \mathrm{Tr}E| \leq \|E\|_F^2.$$

Proof. Let $f(t) = \log\det(I+tE)$. Then, by Jacobi's formula, we have

$$\begin{aligned} f'(t) &= \mathrm{Tr}\left((I+tE)^{-1}E\right), \\ f''(t) &= -\mathrm{Tr}((I+tE)^{-1}E(I+tE)^{-1}E). \end{aligned}$$

Therefore, we have

$$\begin{aligned} f(1) &= f(0) + f'(0) + \int_0^1 (1-s)f''(s)ds \\ &= \mathrm{Tr}(E) + \int_0^1 (1-s)f''(s)ds. \end{aligned}$$

We note that

$$\begin{aligned} \left|f''(s)\right| &= \left|\mathrm{Tr}((I+tE)^{-1}E(I+tE)^{-1}E)\right| \\ &\leq \left|\mathrm{Tr}(E^T\left((I+tE)^{-1}\right)^T(I+tE)^{-1}E)\right|. \end{aligned}$$

Since $\|E\|_F \leq \frac{1}{4}$, we have $\|E\|_2 \leq \frac{1}{4}$ and hence $\|(I+tE)^{-1}\|_2 \leq \frac{4}{3}$ for all $0 \leq t \leq 1$. Therefore, we have

$$\left|f''(s)\right| \leq 2\left|\mathrm{Tr}(E^TE)\right| = 2\|E\|_F^2.$$

Therefore, we have

$$|f(1) - \mathrm{Tr}E| \leq \|E\|_F^2.$$

□

Using lemma 14.4.13 and lemma 14.4.21, we have the following:

Lemma 14.4.14. *Given a geodesic walk $\gamma(t) = \exp_x(\frac{t}{\ell}v_x) \in \Pi$ with step size h satisfying $0 < h \leq \frac{1}{nR_1}$. We have that $d\exp_x(v_x)$ is invertible and*

$$\left|\log\det\left(d\exp_x(v_x)\right) - \int_0^\ell \frac{s(\ell-s)}{\ell}Ric(\gamma'(s))ds\right| \leq \frac{(nhR_1)^2}{6}. \tag{14.12}$$

Therefore, we have that

$$\left|\log\det\left(d\exp_x(v_x)\right) - \log\det\left(d\exp_y(v_y)\right)\right| \leq \frac{(nhR_1)^2}{3}.$$

Proof. Let Ψ be the solution of the ODE $\Psi''(t) + R(t)\Psi(t) = 0$, $\Psi'(0) = I/\ell$ and $\Psi(0) = 0$. We know that $\|R(t)\|_2 \leq \|R(t)\|_F \leq R_1$ for all $0 \leq t \leq \ell$. Hence, Lemma 14.4.21 shows that

$$\begin{aligned} \|\Psi(t) - \frac{t}{\ell}I\|_F &\leq \max_{0\leq s\leq\ell}\|R(s)\|_F\left(\frac{t^3}{5}\|I/\ell\|_2\right) \\ &\leq \frac{1}{5}R_1nh \leq \frac{1}{4} \end{aligned} \tag{14.13}$$

By the Lemma 14.4.13, we have that

$$|\log\det(\Psi(\ell)) - \mathrm{Tr}\left(\Psi(\ell) - I\right)| \leq \left(\frac{1}{5}R_1nh\right)^2.$$

Now, we need to estimate $\mathrm{Tr}(\Psi(\ell) - I)$. Note that

$$\begin{aligned}\Psi(\ell) &= \Psi(0) + \Psi'(0)\ell + \int_0^\ell (\ell - s)R(s)\Psi(s)ds \\ &= I + \int_0^\ell (\ell - s)R(s)\Psi(s)ds.\end{aligned}$$

Hence, we have

$$\Psi(\ell) - I - \int_0^\ell \frac{s(\ell - s)}{\ell} R(s)ds = \int_0^\ell (\ell - s)R(s)\left(\Psi(s) - \frac{s}{\ell}I\right)ds.$$

Using Lemma (14.13), we have

$$\begin{aligned}\left|\mathrm{Tr}\left(\Psi(\ell) - I - \int_0^\ell \frac{s(\ell - s)}{\ell} R(s)ds\right)\right| &\leq \int_0^\ell (\ell - s)\left|\mathrm{Tr}R(s)\left(\Psi(s) - \frac{s}{\ell}I\right)\right| ds \\ &\leq \int_0^\ell (\ell - s)\|R(s)\|_F \|\Psi(s) - \frac{s}{\ell}I\| ds \\ &\leq \frac{\ell^2}{2} \cdot R_1 \cdot \frac{1}{5}R_1 nh \leq \frac{(nhR_1)^2}{10}.\end{aligned}$$

Lemma 14.4.12 shows that $d\exp_x(v_x)\left(\sum w_i X_i(0)\right) = \sum_i \psi_w(\ell)_i X(\ell) = \sum_i (\Psi(\ell)w)_i X_i(\ell)$. Since $\{X_i(t)\}_{i=1}^n$ are orthonormal, this shows that

$$d\exp_x(v_x) = X(\ell)\Psi(\ell)X(0)^T$$

where X is the matrix $[X_1, X_2, \cdots, X_n]$. Since $\|\Psi(\ell) - I\|_F \leq \frac{1}{5}$ (14.13), we have that $\Psi(\ell)$ is invertible and so is $d\exp_x(v_x)$.

Since $X(\ell)$ and $X(0)$ are orthonormal, we have that

$$\log\det\left(d\exp_x(v_x)\right) = \log\det\Psi(\ell).$$

Therefore, this gives

$$\left|\log\det\left(d\exp_x(v_x)\right) - \int_0^\ell \frac{s(\ell - s)}{\ell}\mathrm{Tr}R(s)ds\right| \leq \frac{(nhR_1)^2}{6}.$$

By the definition of Ricci curvature and Fact 14.2.3, we have that

$$\begin{aligned}\mathrm{Tr}R(s) &= \sum_i \left\langle R(X_i(s), \gamma'(s))\gamma'(s), X_j(s)\right\rangle \\ &= \mathrm{Ric}(\gamma'(s)).\end{aligned}$$

This gives the result (14.12).

Since the geodesic $\exp_x(tv_x)$ is the same as $\exp_y(tv_y)$ except for swapping the parameterization, and since $\int_0^\ell \frac{s(\ell-s)}{\ell}\mathrm{Tr}R(s)ds$ is invariant under this swap, (14.12) implies that $\log\det\left(d\exp_x(v_x)\right)$ is close to $\log\det\left(d\exp_y(v_y)\right)$. □

■ 14.4.6 Implementation

Here, we explain in high level how to implement the geodesic walk in general via an efficient algorithm for approximately solving ODEs. Note that to implement the step, we need to compute the geodesic and compute the probability $p(x \xrightarrow{w} y)$ and $p(y \xrightarrow{w'} x)$. From the formula (14.10), we see that they

involves the term $d\exp_x(w)$. Lemma 14.4.12 shows that $d\exp_x(w)$ can be computed by Jacobi field. Also, Lemma 14.4.12 shows that Jacobi field can be computed by a ODE if it is written in the orthonormal frame systems. Therefore, to implement the geodesic walk, we need to compute geodesic, parallel transport and Jacobi field (See Algo 49). All of these are ODEs and can be solved using the collocation method. In the later sections, we will see how to collocation method can indeed solves these ODEs in matrix multiplication time.

Algorithm 49: Geodesic Walk (Detailed)

Pick a Gaussian random vector $w \sim N_x(0,1)$, i.e. $\mathbf{E}_w\|w\|_x^2 = 1$.
`/* Compute` $y = \exp_x(\sqrt{h}w + \frac{h}{2}\mu(x))$ `where` $\mu(x)$ `is given by (14.2). */`
`/* Compute a corresponding` w' `s.t.` $x = exp_y(\sqrt{h}w' + \frac{h}{2}\mu(y))$. `*/`
Generate a random direction $d = \sqrt{h}w + \frac{h}{2}\mu(x)$ where $\mu(x)$ is given by (14.2).
Solve the geodesic equation $D_{\gamma'}\gamma' = 0$ with $\gamma(0) = x$ and $\gamma'(0) = d$ using collocation method (Sec 14.6).
Set $y = \gamma(1)$ and $w' = -\gamma'(1)$.

`/* Compute the probability` $p(x \xrightarrow{w} y)$ `of going from` x `to` y `using the step` w. `*/`
Pick an orthonormal frame $\{X_i\}_{i=1}^n$ at x.
Compute the parallel transport of $\{X_i\}_{i=1}^n$ along $\gamma(t)$ using collocation method.
Compute $d\exp_x(w)$ via solve the Jacobi field equation (14.11) (in the coordinate systems $\{X_i(t)\}$).
Set $p(x \xrightarrow{w} y) = \det(d\exp_x(w))^{-1}\sqrt{\frac{\det(g(y))}{(2\pi h)^n}}\exp\left(-\frac{1}{2}\|\frac{w-\frac{h}{2}\mu(x)}{\sqrt{h}}\|_x^2\right)$.
Compute $p(y \xrightarrow{w'} x)$ similarly.

With probability $\min\left(1, \frac{p(y\xrightarrow{w'}x)}{p(x\xrightarrow{w}y)}\right)$, go to y; otherwise, stay at x.

14.4.7 Smoothness of p_x

Here we prove Lemma 14.4.6. Recall that the probability density of going from x to y is given by the following formula:

$$p_x(y) = \sum_{v_x : exp_x(v_x)=y} \det\left(d\exp_x(v_x)\right)^{-1}\sqrt{\frac{\det\left(g(y)\right)}{(2\pi h)^n}}\exp\left(-\frac{1}{2}\|\frac{v_x - \frac{h}{2}\mu(x)}{\sqrt{h}}\|_x^2\right)$$

To simplify the calculation, we apply Lemma 14.4.14 and consider the following estimate of $p_x(y)$ instead

$$\tilde{p}_x(y) = \sum_{v_x : exp_x(v_x)=y} \sqrt{\frac{\det\left(g(y)\right)}{(2\pi h)^n}}\exp\left(-\int_0^\ell \frac{t(\ell-t)}{\ell}\mathrm{Ric}(\gamma'(t))dt - \frac{1}{2}\|\frac{v_x - \frac{h}{2}\mu(x)}{\sqrt{h}}\|_x^2\right)$$

where $\gamma(t)$ be the geodesic from $\gamma(0) = x$ to $\gamma(\ell) = y$. Lemma 14.4.14 shows that $|\log\left(\tilde{p}_x(y)/p_x(y)\right)| \leq \frac{(nhR_1)^2}{6}$ and hence it suffices to prove the smoothness of $\tilde{p}_x(y)$.

Let $c(s)$ be an unit speed geodesic going from x to some point z infinitesimally close to x. For (almost) any v_x such that $\exp_x(v_x) = y$, Lemma 14.4.24 shows that there is an unique vector field $v(s)$

on $c(s)$ such that $\exp_{c(s)}(v(s)) = y$ and $v(0) = v_x$. Now, we define

$$\zeta(s) = \sqrt{\frac{\det(g(y))}{(2\pi h)^n}} \exp\left(- \int_0^\ell \frac{t(\ell - t)}{\ell} \mathrm{Ric}(\gamma_s'(t))dt - \frac{1}{2} \| \frac{v(s) - \frac{h}{2}\mu(c(s))}{\sqrt{h}} \|_{c(s)}^2 \right)$$

where $\exp_{c(s)}(v(s)) = y$ and $\gamma_s(t) = \exp_{c(s)}(\frac{t}{\ell} v(s))$. Then, we have that

$$\tilde{p}_{c(s)}(y) = \sum_{v(s):\exp_{c(s)}(v(s))=y} \zeta(s)$$

and hence

$$\frac{d}{ds}\tilde{p}_{c(s)}(y) = \sum_{v(s):\exp_{c(s)}(v(s))=y} \left(- \int_0^\ell \frac{t(\ell - t)}{\ell} \frac{d}{ds}\mathrm{Ric}(\gamma_s'(t))dt - \frac{1}{2}\frac{d}{ds} \| \frac{v(s) - \frac{h}{2}\mu(c(s))}{\sqrt{h}} \|_{c(s)}^2 \right) \zeta(s).$$

Hence, it suffices to bound the terms in the bracket.

In Section 14.4.7.1, we analyze $\frac{d}{ds}\mathrm{Ric}(\gamma_s'(t))$ and prove that

$$\left| \int_0^\ell \frac{t(\ell - t)}{\ell} \frac{d}{ds}\mathrm{Ric}(\gamma_s'(t))dt \right| \leq 2nhR_2.$$

In Section 14.4.7.2, we analyze $\frac{d}{ds} \| \frac{v(s) - \frac{h}{2}\mu(c(s))}{\sqrt{h}} \|_{c(s)}^2$ and prove that

$$\mathbf{E}_\gamma \left| \frac{d}{ds} \| \frac{v(s) - \frac{h}{2}\mu(c(s))}{\sqrt{h}} \|_{c(s)}^2 \right|_{s=0} \Bigg| = O\left(D_2\sqrt{h} + \frac{1}{\sqrt{h}} + D_0 \right).$$

Therefore, we have that

$$\left| \frac{d}{ds}\tilde{p}_c(y) \right| = O\left(nhR_2 + \sqrt{h}D_2 + \frac{1}{\sqrt{h}} + nhR_1D_0 \right) \tilde{p}_c(y).$$

Using $|\log(\tilde{p}_x(y)/p_x(y))| \leq \frac{(nhR_1)^2}{6}$, we have our result.

Theorem 14.4.15. *We have that*

$$d_{TV}(p_x, p_y) = O\left(nhR_2 + \sqrt{h}D_2 + \frac{1}{\sqrt{h}} + nhR_1D_0 \right) d(x, y) + O\left(nhR_1\right)^2.$$

14.4.7.1 Smoothness of exponential map

Definition 14.4.16. Given a manifold M and a geodesic walk $\gamma(t)$ on M_L with step size h. Let R_2 be a constant depending on the manifold M and the step size h such that for any t such that $0 \leq t \leq \ell$, any curve $\alpha(s)$ starting from $\gamma(t)$ and any vector field $u(s)$ on $\alpha(s)$, we have that

$$\left| \frac{d}{ds}\, Ric(u(s))|_{s=0} \right| \leq \left(\|\alpha'(0)\| + \|D_s u(0)\| \right) R_2.$$

Lemma 14.4.17. *For* $\frac{1}{n} < h \leq \frac{1}{2nR_1}$, *we have*

$$\left| \int_0^\ell \frac{t(\ell - t)}{\ell} \frac{d}{ds} Ric(\gamma_s'(t))dt \right| \leq 2nhR_2$$

where γ_s is as defined in the beginning of Section 14.4.7.

Proof. Fix any t such that $0 \le t \le \ell$ and let $\alpha(s) = \gamma_s(t)$ and $u(s) = \gamma_s'(t)$ be a vector field on α. By Definition 14.4.16, we have that

$$\left| \frac{d}{ds} Ric(u(s))|_{s=0} \right| \le \left(\|\alpha'(0)\| + \|D_s u(0)\| \right) R_2.$$

Recall that $\gamma_s = \exp_{c(s)}(\frac{t}{\ell} v(s))$ and hence

$$\begin{aligned} \frac{d\alpha}{ds}(0) &= \frac{\partial}{\partial s} \exp_{c(s)}(\frac{t}{\ell} v(s))|_{s=0}, \\ D_s u(0) &= D_s \frac{\partial}{\partial t} \exp_{c(s)}(\frac{t}{\ell} v(s))|_{s=0} \\ &= D_t \frac{\partial}{\partial s} \exp_{c(s)}(\frac{t}{\ell} v(s))|_{s=0}. \end{aligned}$$

Let $\psi(t) = \frac{\partial}{\partial s} \exp_{c(s)}(\frac{t}{\ell} v(s))|_{s=0}$. Since $\exp_{c(s)}(\frac{t}{\ell} v(s))$ is a geodesic for each s, $\psi(t)$ satisfies the Jacobi field equation. It is easier to calculate using orthogonal frame. So, we pick an arbitrary orthogonal frame along the curve $\gamma(t)$ and let $\phi(t)$ be $\psi(t)$ represented in that orthogonal frame. Then, we know that

$$\begin{aligned} \frac{d^2}{dt^2}\phi(t) + R(\phi(t), \gamma'(t))\gamma'(t) &= 0, \\ \phi(0) &= \alpha, \\ \phi'(0) &= \frac{\beta}{\ell} \end{aligned}$$

for some α and β. Since c is a unit speed geodesic, we have that $\alpha = \phi(0) = \frac{\partial c}{\partial s}(0)$ has ℓ_2 norm 1. Next, we note that $\beta = \ell\phi'(0) = D_s v(s)$. By lemma 14.4.22, we have that $\|\beta\|_2 \le \frac{3}{2}$.

Therefore, Lemma 14.4.20 shows that

$$\|\phi(t) - \alpha - \frac{\beta}{\ell} t\|_2 \le 2R_1 \ell^2$$

for all $0 \le t \le \frac{1}{\sqrt{R_1}}$. Hence, we have $\|\phi(t)\|_2 \le 4$ for all $0 \le t \le \frac{1}{\sqrt{R_1}}$. By Lemma 14.4.20 again, we have

$$\|\phi'(t) - \beta\|_2 \le 3R_1 \ell$$

for all $0 \le t \le \frac{1}{\sqrt{R_1}}$. Hence, we have $\|\phi'(t)\|_2 \le 4$ for all $0 \le t \le \frac{1}{\sqrt{R_1}}$.

Therefore, we have

$$\begin{aligned} \left| \frac{d}{ds} Ric(u(s))|_{s=0} \right| &\le \left(\|\alpha'(0)\| + \|D_s u(0)\| \right) R_2 \\ &= \left(\|\phi(t)\|_2 + \|\phi'(t)\|_2 \right) R_2 \\ &\le 8R_2. \end{aligned}$$

This gives the result. □

14.4.7.2 One-step overlap

We parametrize the geodesic from x to y as $\gamma(s)$. For a fixed z, we have $z = \exp_x(d_x) = \exp_{\gamma(s)}(d_{\gamma(s)}) = \exp_{\gamma(s)}(d(s))$.

Lemma 14.4.18. *We have that*

$$\mathbf{E}_\gamma \left| \frac{d}{ds} \| \frac{v(s) - \frac{h}{2}\mu(c(s))}{\sqrt{h}} \|^2_{c(s)} \Big|_{s=0} \right| = O\left(D_2\sqrt{h} + \frac{1}{\sqrt{h}} + D_0 \right).$$

where $c(s)$ is as defined in the beginning of Section 14.4.7.

Proof. Note that

$$\frac{d}{ds} \| \frac{v(s) - \frac{h}{2}\mu(c(s))}{\sqrt{h}} \|^2_{c(s)} \Big|_{s=0} = \frac{2}{h} \left\langle D_s v|_{s=0} - \frac{h}{2} D_s\mu|_{s=0}, v - \frac{h}{2}\mu \right\rangle. \tag{14.14}$$

Since $v - \frac{h}{2}\mu$ is a random Gaussian vector $N(0, nhI)$ in T_cM independent of $D_s\mu$, we have that

$$\mathbf{E} \left| \left\langle \frac{h}{2} D_s\mu|_{s=0}, v - \frac{h}{2}\mu \right\rangle \right| \leq O\left(\frac{\| \frac{h}{2} D_s\mu|_{s=0} \| \sqrt{nh}}{\sqrt{n}} \right) \leq O\left(\frac{hD_2\sqrt{nh}}{2\sqrt{n}} \right). \tag{14.15}$$

By Lemma 14.4.22, we know that $D_s v|_{s=0} = -c' + \zeta$ where $\zeta \perp v(0)$ and $\|\zeta\| \leq \frac{3}{2}nhR_1\|c'\|$. Since $v - \frac{h}{2}\mu$ is a random Gaussian vector independent of c', we have that

$$\begin{aligned} \mathbf{E} \left| \left\langle D_s v, v - \frac{h}{2}\mu \right\rangle \right| &\leq \mathbf{E} \left| \left\langle c', v - \frac{h}{2}\mu \right\rangle \right| + \mathbf{E}\frac{h}{2} |\langle \zeta, \mu \rangle| \\ &\leq O\left(\frac{\|c'\|\sqrt{nh}}{\sqrt{n}} \right) + nh^2 R_1 \|c'\| \|\mu\| \\ &\leq O\left(\sqrt{h} + nh^2 R_1 D_0 \right). \end{aligned} \tag{14.16}$$

Combining the bounds (14.14), (14.15) and (14.16), we have that

$$\mathbf{E} \left| \frac{d}{ds} \| \frac{v(s) - \frac{h}{2}\mu(c(s))}{\sqrt{h}} \|^2_{c(s)} \Big|_{s=0} \right| \leq O\left(D_2\sqrt{h} + \frac{1}{\sqrt{h}} + nhR_1 D_0 \right).$$

□

14.4.8 Approximate Solution of Jacobi Field

Let $\gamma(t)$ be a geodesic and $\{X_i(t)\}_{i=1}^n$ be the parallel transport of some orthonormal frame along $\gamma(t)$. As we demonstrated in the proof of Lemma 14.4.12, Jacobi fields are solutions the following matrix ODE:

$$\begin{aligned} \frac{d^2}{dt^2}\psi(t) + R(t)\psi(t) &= 0, \\ \frac{d}{dt}\psi(0) &= \beta, \\ \psi(0) &= \alpha \end{aligned} \tag{14.17}$$

where $\psi(t), \alpha, \beta \in \mathbb{R}^n$ and $R(t)$ is a matrix given by $R_{ji}(t) = \langle R(X_i(t), \gamma'(t))\gamma'(t), X_j(t) \rangle$. Recall that $R(t)$ is a symmetric matrix (Fact 14.2.3).

In this section, we show give an approximate solution of the the Jacobi equation (14.17). First, we give a basic bound on the solution in terms of hyperbolic sine and cosine functions.

Lemma 14.4.19. *Let ψ be the solution of (14.17). Suppose that $\|R(t)\|_2 \leq \lambda$. Then, we have that*

$$\|\psi(t)\|_2 \leq \|\alpha\|_2 \cosh(\sqrt{\lambda}t) + \frac{\|\beta\|_2}{\sqrt{\lambda}} \sinh(\sqrt{\lambda}t)$$

for all $t \geq 0$.

Proof. Let $a(t) = \|\psi(t)\|_2$. Using $\|R(t)\|_2 \leq \lambda$ and $\frac{d^2}{dt^2}\psi(t) + R(t)\psi(t) = 0$, we have that

$$a''(t) \leq \lambda a(t).$$

Since $a(0) = \|\alpha\|_2$ and $a'(0) \leq \|\beta\|_2$, for all $t \geq 0$, we have that $a(t) \leq b(t)$ where $b(t)$ is the solution of the $b''(t) = \lambda b(t)$, $b(0) = \|\alpha\|_2$, $b'(0) = \|\beta\|_2$. Solving these equations, we have

$$a(t) \leq b(t) = \|\alpha\|_2 \cosh(\sqrt{\lambda}t) + \frac{\|\beta\|_2}{\sqrt{\lambda}} \sinh(\sqrt{\lambda}t)$$

for all $t \geq 0$. □

Next, we give an approximate solution of (14.17).

Lemma 14.4.20. *Let ψ be the solution of (14.17). Suppose that $\|R(t)\|_2 \leq \lambda$. For any $0 \leq t \leq \frac{1}{\sqrt{\lambda}}$, we have that*

$$\|\psi(t) - \alpha - \beta t\|_2 \leq \lambda t^2 \|\alpha\|_2 + \frac{\lambda t^3}{5}\|\beta\|_2 \text{ and } \|\psi'(t) - \beta\|_2 \leq 2\lambda t\|\alpha\|_2 + \frac{3\lambda t^2}{5}\|\beta\|_2.$$

Proof. Note that

$$\begin{aligned} \psi(t) &= \psi(0) + \psi'(0)t + \int_0^t (t-s)\psi''(s)ds \\ &= \alpha + \beta t - \int_0^t (t-s)R(s)\psi(s)ds. \end{aligned}$$

Using Lemma 14.4.19 and $\|R(t)\|_2 \leq \lambda$, we have that

$$\begin{aligned} \|\psi(t) - \alpha - \beta t\|_2 &\leq \lambda \int_0^t (t-s)\|\psi(s)\|_2 ds \\ &\leq \lambda \int_0^t (t-s)\left(\|\alpha\|_2 \cosh(\sqrt{\lambda}s) + \frac{\|\beta\|_2}{\sqrt{\lambda}} \sinh(\sqrt{\lambda}s)\right) ds \\ &= \|\alpha\|_2 \left(\cosh(\sqrt{\lambda}t) - 1\right) + \frac{\|\beta\|_2}{\sqrt{\lambda}} \left(\sinh(\sqrt{\lambda}t) - \sqrt{\lambda}t\right). \end{aligned}$$

Since $0 \leq t \leq \frac{1}{\sqrt{\lambda}}$, we have that $\left|\cosh(\sqrt{\lambda}t) - 1\right| \leq \lambda t^2$ and $\left|\sinh(\sqrt{\lambda}t) - \sqrt{\lambda}t\right| \leq \frac{\lambda^{3/2}t^3}{5}$. This gives the result.

Similarly, we have that $\psi'(t) = \psi'(0) + \int_0^t \psi''(s)ds = \psi'(0) + \int_0^t R(s)\psi(s)ds$ and hence

$$\begin{aligned} \|\psi'(t) - \beta\|_2 &\leq \lambda \int_0^t \left(\|\alpha\|_2 \cosh(\sqrt{\lambda}s) + \frac{\|\beta\|_2}{\sqrt{\lambda}} \sinh(\sqrt{\lambda}s)\right) ds \\ &\leq \sqrt{\lambda}\|\alpha\|_2 \sinh(\sqrt{\lambda}t) + \|\beta\|_2 \left(\cos(\sqrt{\lambda}t) - 1\right) \\ &\leq 2\lambda t\|\alpha\|_2 + \frac{3}{5}\lambda t^2 \|\beta\|_2 \end{aligned}$$

□

The following is a matrix version of the above result.

Lemma 14.4.21. *Let ψ be the solution of*

$$\begin{aligned} \frac{d^2}{dt^2}\Psi(t) + R(t)\Psi(t) &= 0, \\ \frac{d}{dt}\Psi(0) &= B, \\ \Psi(0) &= A \end{aligned}$$

where Ψ, A and B are matrices with a compatible size. Suppose that $\|R(t)\|_2 \leq \lambda$. For any $0 \leq t \leq \frac{1}{\sqrt{\lambda}}$, we have that

$$\|\Psi(t) - A - Bt\|_F \leq \max_{0\leq s\leq t} \|R(s)\|_F \left(t^2\|A\|_2 + \frac{t^3}{5}\|B\|_2\right).$$

Proof. Note that $\Psi(t)x$ is the solution of (14.17) with $\beta = Bx$ and $\alpha = Ax$. Therefore, Lemma 14.4.19 shows that

$$\|\Psi(t)x\|_2 \leq \left(\|A\|_2 \cosh(\sqrt{\lambda}t) + \frac{\|B\|_2}{\sqrt{\lambda}} \sinh(\sqrt{\lambda}t)\right) \|x\|_2$$

for all x. Therefore, we have that

$$\Psi(t)^T\Psi(t) \preceq \left(\|A\|_2 \cosh(\sqrt{\lambda}t) + \frac{\|B\|_2}{\sqrt{\lambda}} \sinh(\sqrt{\lambda}t)\right)^2 I.$$

Hence, we have that

$$\Psi(t)\Psi(t)^T \preceq \left(\|A\|_2 \cosh(\sqrt{\lambda}t) + \frac{\|B\|_2}{\sqrt{\lambda}} \sinh(\sqrt{\lambda}t)\right)^2 I.$$

Using this, we have that

$$\begin{aligned} \|\Psi(t) - A - Bt\|_F &= \|\int_0^t (t-s)R(s)\Psi(s)ds\|_F \\ &\leq \int_0^t (t-s)\sqrt{\mathrm{Tr}\Psi^T(s)R^T(s)R(s)\Psi(s)}ds \\ &= \int_0^t (t-s)\sqrt{\mathrm{Tr}R(s)\Psi(s)\Psi^T(s)R^T(s)}ds \\ &\leq \int_0^t (t-s)\left(\|A\|_2 \cosh(\sqrt{\lambda}s) + \frac{\|B\|_2}{\sqrt{\lambda}} \sinh(\sqrt{\lambda}s)\right)\|R(s)\|_F ds. \\ &\leq \max_{0\leq s\leq t}\|R(s)\|_F \int_0^t (t-s)\left(\|A\|_2 \cosh(\sqrt{\lambda}s) + \frac{\|B\|_2}{\sqrt{\lambda}} \sinh(\sqrt{\lambda}s)\right) ds \\ &= \max_{0\leq s\leq t}\|R(s)\|_F \left(\|A\|_2\left(\cosh(\sqrt{\lambda}t) - 1\right) + \frac{\|B\|_2}{\sqrt{\lambda}}\left(\sinh(\sqrt{\lambda}t) - \sqrt{\lambda}t\right)\right). \end{aligned}$$

Since $0 \leq t \leq \frac{1}{\sqrt{\lambda}}$, we have that $\left|\cosh(\sqrt{\lambda}t) - 1\right| \leq \lambda t^2$ and $\left|\sinh(\sqrt{\lambda}t) - \sqrt{\lambda}t\right| \leq \frac{\lambda^{3/2}t^3}{5}$. This gives the result. □

14.4.9 Almost One-to-One Correspondence of Geodesics

We do not know if every pair $x, y \in M$ has a unique geodesic connecting x and y. Due to this, the probability density p_x at y on $\mathcal{M}$ can be the sum over all possible geodesics connect x and y. The goal

of this section is to show there is a 1-1 map between geodesics paths connecting x to y as we move x.

Lemma 14.4.22. *Given a geodesic walk $\gamma(t) = \exp_x(\frac{t}{\ell}v_x) \in \Pi$ with step size h satisfying $\frac{1}{n} \le h \le \frac{1}{nR_1}$. Let the end points $x = \gamma(0)$ and $y = \gamma(\ell)$. There is an unique smooth invertible function $v : U \subset M \to V \subset T_xM$ such that*

$$y = \exp_z(v(z))$$

for any $z \in U$ where U is a neighborhood of x and V is a neighborhood of $v_x = v(x)$. Furthermore, for any $\eta = \eta_1 + \alpha v_x$ with $\eta_1 \perp v_x$ and scale α, we have that

$$\nabla_\eta v(x) = -\eta + \zeta$$

where $\|\zeta\| \le \frac{3}{2}nhR_1\|\eta_1\| \le \frac{3}{2}\|\eta_1\|$ and $\zeta \perp v(x)$. In particular, we have that $\|\nabla_\eta v(x)\| \le \frac{5}{2}\|\eta\|$.

Proof. Consider the function $f(z, w) = \exp_z(w)$. It is continuously differentiable. From Lemma 14.4.14, the differential of w at (x, v_x) on the w variables, i.e. $d\exp_x(v_x)$, is invertible. Hence, the implicit function theorem shows that there is a open neighborhood U of x and a unique function v on U such that $f(z, v(z)) = f(z, v_z)$, i.e. $y = \exp_x(v_x) = \exp_z(v(z))$.

To compute $\nabla_\eta v$, let $c(t, s) = \exp_{\gamma(s)}(\frac{t}{\ell}v(\gamma(s)))$ be a family of geodesics with the end points $c(\ell, s) = \exp_{\gamma(s)}(v(\gamma(s))) = y$. Note that $\psi(t) \stackrel{\text{def}}{=} \frac{\partial c(t,s)}{\partial s}\,|_{s=0}$ satisfies the Jacobi field equation

$$D_t^2\psi(t) + R(\psi, \frac{\partial c}{\partial t})\frac{\partial c}{\partial t} = 0.$$

Moreover, we know that

$$\begin{aligned}
\psi(0) &= \frac{\partial c(0,s)}{\partial s}\,|_{s=0} = \gamma'(0) = \eta, \\
D_t\psi(0) &= D_t\frac{\partial c(t,s)}{\partial s}\,|_{t,s=0} = D_s\frac{\partial c(t,s)}{\partial t}\,|_{t,s=0} = \frac{1}{\ell}\nabla_\eta v(x) \stackrel{\text{def}}{=} \frac{\chi}{\ell}, \\
\psi(\ell) &= \frac{\partial c(\ell,s)}{\partial s}\,|_{s=0} = 0
\end{aligned}$$

where we used $D_t\frac{\partial}{\partial s} = D_s\frac{\partial}{\partial t}$ (torsion free-ness, Fact 14.2.1).

From Fact 14.2.4, we know that ψ can be split into the tangential part ψ_1 and the normal part ψ_2. For the tangential part, we know that

$$\begin{aligned}
\psi_1(t) &= \left(\left\langle \psi(0), \frac{dc}{dt}(0,0)\right\rangle_{c(0)} + \left\langle D_t\psi(0), \frac{dc}{dt}(0,0)\right\rangle_{c(0)} t\right)\frac{dc}{dt}(t) \\
&= \left(\langle \eta, v_x\rangle_x + \left\langle \frac{\chi}{\ell}, v_x\right\rangle_x t\right)\frac{dc}{dt}(t) \\
&= \left(\alpha\|v_x\|^2 + \left\langle \frac{\chi}{\ell}, v_x\right\rangle_x t\right)\frac{dc}{dt}(t).
\end{aligned}$$

Since $\psi_1(\ell) = 0$, we have that

$$\langle \chi, v_x\rangle = -\alpha\|v_x\|^2. \tag{14.18}$$

For the normal part ψ_2, it is easier to calculate using orthogonal frames. So, we pick an arbitrary orthogonal frame parallel transported along the curve $c(t, 0)$ and let $\bar{\psi}(t)$ be $\psi_2(t)$ represented in that

orthogonal frame. Hence, we have that

$$\begin{aligned} \frac{d^2}{dt^2}\bar{\psi}(t) + R(\bar{\psi}(t), \frac{\partial c}{\partial t})\frac{\partial c}{\partial t} &= 0, \\ \bar{\psi}(0) &= \eta_1, \\ \bar{\psi}'(0) &= \frac{\chi_1}{\ell}, \\ \bar{\psi}(\ell) &= 0 \end{aligned}$$

where $\chi_1 = \chi - \left\langle \chi, \frac{v_x}{\|v_x\|} \right\rangle \frac{v_x}{\|v_x\|} = \chi + \alpha v_x$ (14.18). Using $h \leq \frac{1}{nR_1}$, Lemma 14.4.20 shows that

$$\begin{aligned} \|\eta_1 + \chi_1\|_2 = \|\bar{\psi}(\ell) - \eta_1 - \chi_1\|_2 &\leq R_1\ell^2\|\eta_1\|_2 + \frac{R_1\ell^2}{5}\|\chi_1\|_2 \\ &\leq \frac{6}{5}R_1\ell^2\|\eta_1\|_2 + \frac{1}{5}\|\chi_1\|_2. \end{aligned}$$

Hence, we have that $\|\eta_1 + \chi_1\|_2 \leq \frac{3}{2}R_1\ell^2\|\eta_1\|_2$. Therefore, we have that

$$\begin{aligned} \nabla_\eta v(x) &= \chi \\ &= -\alpha v_x + \chi_1 \\ &= -\eta + (\eta_1 + \chi_1) \end{aligned}$$

where $\|\chi_1 + \eta_1\|_2 \leq \frac{3}{2}\|\eta_1\|_2$. Furthermore, we have that both η_1 and χ_1 are orthogonal to η. □

Remark. If the above lemma holds without the assumption $\gamma \in \Pi$, this would imply uniqueness of geodesics.

The following lemma shows there is a 1-1 map between geodesics paths connecting x to y as we move x. When we move x, the geodesic γ from x to y no longer insides Ξ. To avoid this issue, we need to assume $\langle R(X_i, \gamma')\gamma', X_j\rangle$ is mildly smooth in the following sense.

Assumption 14.4.23. *Given a manifold M. There is some constant C such that for any a curve $\alpha(s) \in M$ and a vector field $u(s)$ on $\alpha(s)$, let $A(s)_{ij} = \langle R(X_i, u(s))u(s), X_j\rangle$ where $\{X_i\}$ is any orthonormal frame at $\alpha(s)$, we have*

$$\left|\frac{d}{ds}\|A(s)\|_F\right| \leq \left(\|\alpha'(0)\| + \|D_s u(0)\|\right) n^C.$$

Note that this assumption implies $R_1 = n^{\Theta(1)}$ by picking the curve is a singleton and the vector field $u(s) = s \cdot u$.

Lemma 14.4.24. *Given a geodesic $\gamma(t) = \exp_x(\frac{t}{\ell}v_x)$ on M with step size h satisfying $\frac{1}{n} \leq h \leq \frac{1}{2nR_1}$, let $c(s)$ be any geodesic starting at $\gamma(0)$. Let $x = c(0) = \gamma(0)$ and $z = c(1)$. Suppose that the length of the geodesic is less than $n^{-\Theta(1)}$. Then, there is a unique vector field v on c such that*

$$y = \exp_{c(s)}(v(s)).$$

This vector field is uniquely determined by the geodesic and any $v(s)$ on this vector field.

Proof. Let $s_{\max}$ be the supremum of s such that $v(s)$ can be defined continuously and $y = \exp_{c(s)}(v(s))$. Note that $s_{\max} > 0$ as Lemma 14.4.22 shows that there is a neighborhood N at x and a vector field u on N such that for any $z \in N$, we have that

$$y = \exp_z(u(z)).$$

Note that the the vector field u is unique and hence the extension we found for v is unique. Also, we know that $\|D_t v(t)\| = \|\nabla_{c'} u(x)\| \le \frac{5}{2}L$ where L is the length of c. Therefore, by continuity, $v(s_{\max})$ is well-defined.

If $s_{\max} > 1$, then we are done. Otherwise, we want to apply Lemma 14.4.22 at $s_{\max}$. However, the curve $\exp_{c(s_{\max})}(\frac{t}{\ell} v(s_{\max}))$ may not lie in Π. Note that the only place Lemma 14.4.22 uses the assumption $\gamma = \exp_x(\frac{t}{\ell} v_x) \in \Pi$ is to apply the bound $R_1 = \sup_{\gamma \in \Pi, 0 \le t \le \ell} \|R(t)\|_F$.

To bound $\|R(t)\|_F$, we note that $\|D_t v(t)\| = \|\nabla_{c'} u(x)\| \le \Theta(L)$ where L is the length of c. Hence, both the starting point and the direction of the geodesic move by $\Theta(L)$. Now, we can apply Lemma 14.4.19 to prove that along the geodesic, all the points move by $\Theta(L)$ for $0 \le t \le \ell$. Then, we can apply Assumption 14.4.23 to conclude that $\|R(t)\|_F$ only changes by at most $\Theta(n^C L)$. Therefore, if $L \le \frac{R_1}{n^C} = n^{-C}$, then $\|R(t)\|_F$ is still bounded by $\Theta(R_1)$ and Lemma 14.4.22 can be applied at $s_{\max} < 1$. This is a contradiction. Hence, $s_{\max} \ge 1$. Its uniqueness follows from Lemma 14.4.22. □

14.5 Logarithmic Barrier

For any polytope $M = \{Ax > b\}$, the logarithmic barrier function $\phi(x)$ is defined as

$$\phi(x) = -\sum_{i=1}^{m} \log(a_i^T x - b_i).$$

We denote the Hessian manifold induced by the logarithmic barrier on M by M_L. The goal of this section is to analyze the geodisic walk on M_L.

In section 14.5.1, we give explicit formulas of various Riemannian geometry concepts on M_L. In Section 14.5.2, we explain the geodesic walk on M_L. In Sections 14.5.3 to 14.5.7, we bound the necessary constant required by Theorem 14.4.2. This shows that

Theorem 14.5.1. *The geodesic walk on M_L with step size $h \le \frac{c}{n^{3/4}}$ has mixing time is $O(m/h)$ for some universal constant c.*

In the sections afterward, we show how to implement geodesic walk and calculate the rejection probability. To implement these, we apply the techniques developed in Section 14.6 to solve the corresponding ODEs. In this section, we first show that geodesic, parallel transport and Jacobi Field are complex analytic (Section 14.5.9), then we bound their radius of convergence (Section 14.5.9.1). This will help us bound the approximation error of the collocation method developed in Section 14.6.

Theorem 14.5.2. *We can implement one step of geodesic walk with step size $h \le \frac{c}{\sqrt{n}}$ in time $O(mn^{\omega-1}\log^2(n))$ for a universal constant c.*

14.5.1 Riemannian geometry on M_L

We use the following definitions throughout this section:

- $s_x = Ax - b$, $S_x = \mathrm{Diag}(s_x)$.
- $A_x = S_x^{-1} A$.
- $s_{x,v} = A_x v$, $S_{x,v} = \mathrm{Diag}(A_x v)$.
- $P_x = A_x (A_x^T A_x)^{-1} A_x^T$, $\sigma_x = \mathrm{Diag}(P_x)$.
- $\left(P_x^{(2)}\right)_{ij} = (P_x)_{ij}^2$.

- For brevity (overloading notation), we define $s_{\gamma'} = s_{\gamma,\gamma'}$, $s_{\gamma''} = s_{\gamma,\gamma''}$, $S_{\gamma'} = S_{\gamma,\gamma'}$ and $S_{\gamma''} = S_{\gamma,\gamma''}$ for a curve $\gamma(t)$.

Since the Hessian manifold M_L is naturally embedded in $\mathbb{R}^n$, we identify $T_x M_L$ by Euclidean coordinates unless otherwise stated. Therefore, we have that

$$\begin{aligned} \langle u, v \rangle_x &= u^T \nabla^2 \phi(x) v \\ &= u^T A_x^T A_x v. \end{aligned}$$

Lemma 14.5.3. *Let $u(t)$ be a vector field defined on a curve $\gamma(t)$ in M_L. Then, we have that*

$$\nabla_{\gamma'} u = \frac{du}{dt} - \left(A_\gamma^T A_\gamma\right)^{-1} A_\gamma^T S_{\gamma'} s_{\gamma,u}.$$

In particular, the geodesic equation on M_L is given by

$$\gamma'' = \left(A_\gamma^T A_\gamma\right)^{-1} A_\gamma^T s_{\gamma'}^2 \tag{14.19}$$

and the equation for parallel transport on a curve $\gamma(t)$ is given by

$$\frac{d}{dt} v(t) = \left(A_\gamma^T A_\gamma\right)^{-1} A_\gamma^T S_{\gamma'} A_\gamma v. \tag{14.20}$$

Proof. By Lemma 14.2.5, the Christoffel symbols respect to the Euclidean coordinates is given by

$$\begin{aligned} \Gamma_{ij}^k &= \frac{1}{2} \sum_l g^{kl} \phi_{ijl} \\ &= -\sum_z \sum_l e_k^T \left(A_x^T A_x\right)^{-1} e_l \left(A_x e_i\right)_z \left(A_x e_j\right)_z \left(A_x e_l\right)_z \\ &= -e_k^T \left(A_x^T A_x\right)^{-1} A_x^T \left(\left(A_x e_i\right) \left(A_x e_j\right)\right). \end{aligned}$$

Recall that the Levi-Civita connection is given by

$$\nabla_v u = \sum_{ik} v_i \frac{\partial u_k}{\partial x^i} e_k + \sum_{ijk} v_i u_j \Gamma_{ij}^k e_k.$$

Therefore, we have

$$\nabla_v u = \sum_{ik} v^i \frac{\partial u^k}{\partial x^i} e_k - \left(A_x^T A_x\right)^{-1} A_x^T S_{x,v} s_{x,u}.$$

Since U is a vector field defined on a curve $\gamma(t)$, we have that $\sum_{ik} \gamma_i' \frac{\partial u_k}{\partial x^i} e_k = \frac{du}{dt}$.

The geodesic equation follows from $\nabla_{\gamma'} \gamma' = 0$ and the parallel transport equation follows from $\nabla_{\gamma'} v = 0$. □

Lemma 14.5.4. *Given $u, v, w, z \in T_x M_L$, the Riemann Curvature Tensor is given by*

$$\langle R(u,v)w, z \rangle = \left(s_{x,u} s_{x,w}\right)^T A_x \left(A_x^T A_x\right)^{-1} A_x^T \left(s_{x,v} s_{x,z}\right) - \left(s_{x,u} s_{x,z}\right)^T A_x \left(A_x^T A_x\right)^{-1} A_x^T \left(s_{x,v} s_{x,w}\right)$$

and the Ricci curvature is given by

$$Ric(u) = s_{x,u}^T P_x^{(2)} s_{x,u} - \sigma_x^T P_x s_{x,u}^2.$$

Proof. By Lemma 14.2.5, we have that

$$\begin{aligned}\langle R(u,v)w,z\rangle &= \frac{1}{4}\sum_{pqijlk} g^{pq}\left(\phi_{jkp}\phi_{ilq}-\phi_{ikp}\phi_{jlq}\right)u^i v^j w^l z^k \\ &= \sum_{pq} g^{pq}\left(e_p^T A_x^T\left(s_{x,v}s_{x,z}\right)e_q^T A_x^T\left(s_{x,u}s_{x,w}\right)-e_p^T A_x^T\left(s_{x,u}s_{x,z}\right)e_q^T A_x^T\left(s_{x,v}s_{x,w}\right)\right) \\ &= \left(s_{x,u}s_{x,w}\right)^T A_x\left(A_x^T A_x\right)^{-1}A_x^T\left(s_{x,v}s_{x,z}\right)-\left(s_{x,u}s_{x,z}\right)^T A_x\left(A_x^T A_x\right)^{-1}A_x^T\left(s_{x,v}s_{x,w}\right)\end{aligned}$$

and

$$\begin{aligned}Ric(u) &= \sum_{jl} g^{jl}\left\langle R(u,\frac{\partial}{\partial x^j}),\frac{\partial}{\partial x^l},u\right\rangle \\ &= \sum_{jl} g^{jl}\left(s_{x,u}s_{x,e_l}\right)^T A_x\left(A_x^T A_x\right)^{-1}A_x^T\left(s_{x,e_j}s_{x,u}\right)-\left(s_{x,u}^2\right)^T A_x\left(A_x^T A_x\right)^{-1}A_x^T\left(s_{x,e_j}s_{x,e_l}\right) \\ &= \mathrm{Tr}S_{x,u}A_x\left(A_x^T A_x\right)^{-1}A_x^T S_{x,u}A_x\left(A_x^T A_x\right)^{-1}A_x-\mathrm{Tr}\left(A_x^T\mathrm{Diag}(P_x s_{x,u}^2)A_x\left(A_x^T A_x\right)^{-1}\right) \\ &= s_{x,u}^T P_x^{(2)}s_{x,u}-\sigma_x^T P_x s_{x,u}^2.\end{aligned}$$

□

Lemma 14.5.5. *Given a geodesic $\gamma(t)$ on M_L and an orthogonal frame $\{x_i\}_{i=1}^n$ on $\gamma(t)$. The Jacobi field equation (in the orthogonal frame coordinate) is given by*

$$\frac{d^2u}{dt^2}+X^{-1}(A_\gamma^T A_\gamma)^{-1}\left(A_\gamma^T S_{\gamma'}P_\gamma S_{\gamma'}A_\gamma-A_\gamma^T\mathrm{Diag}(P_\gamma s_{\gamma'}^2)A_\gamma\right)Xu=0$$

where $X(t)=[x_1(t),x_2(t),\cdots,x_n(t)]$.

Proof. The equation for Jacobi field along is

$$D_t^2 v+R(v,\gamma')\gamma'=0.$$

By Lemma 14.5.4, under Euclidean coordinates, we have

$$D_t^2 v+(A_\gamma^T A_\gamma)^{-1}\left(A_\gamma^T S_{\gamma'}P_\gamma S_{\gamma'}A_\gamma-A_\gamma^T diag(P_\gamma s_{\gamma'}^2)A_\gamma\right)v=0$$

We write v in terms of the orthogonal frame, namely, $v(t)=X(t)u(t)$ where $u(t)\in\mathbb{R}^n$. Then, we have that $D_t^2 v=X(t)\frac{d^2u(t)}{dt^2}$. Hence, under the orthogonal frame coordinate, we have that

$$\frac{d^2u}{dt^2}+X^{-1}(A_\gamma^T A_\gamma)^{-1}\left(A_\gamma^T S_{\gamma'}P_\gamma S_{\gamma'}A_\gamma-A_\gamma^T\mathrm{Diag}(P_\gamma s_{\gamma'}^2)A_\gamma\right)Xu=0.$$

□

Now, we show the relation between the distance induced by the metric and the Hilbert metric.

Lemma 14.5.6. *For any $x,y\in M_L$, we have that*

$$\frac{d(x,y)}{d_H(x,y)}\leq\sqrt{m}.$$

Hence, we have that $G_2\leq\sqrt{m}$.

Proof. First, we note that it suffices to prove that $\frac{d(x,y)}{d_H(x,y)}\leq(1+O(\varepsilon))\sqrt{m}$ for any $x,y\in M_L$ with $d(x,y)\leq\varepsilon$. Then, one can run a limiting argument as follows. Let $x_t=t\cdot x+(1-t)\cdot y$, then we have

that

$$d_H(x,y) = \lim_{n\to\infty} \sum_{i=0}^{n-1} d_H(x_{k/n}, x_{(k+1)/n}).$$

Since $d_H(x_{k/n}, x_{(k+1)/n}) = O_{x,y}(\frac{1}{n})$, we have that

$$\begin{aligned} d_H(x,y) &= \lim_{n\to\infty} \sum_{i=0}^{n-1} d_H(x_{k/n}, x_{(k+1)/n}) \\ &\geq \lim_{n\to\infty} \frac{(1 - O_{x,y}(\frac{1}{n}))}{\sqrt{m}} \sum_{i=0}^{n-1} d(x_{k/n}, x_{(k+1)/n}) \\ &= \frac{1}{\sqrt{m}} \lim_{n\to\infty} \sum_{i=0}^{n-1} d(x_{k/n}, x_{(k+1)/n}) \\ &\geq \frac{1}{\sqrt{m}} \lim_{n\to\infty} d(x,y) = \frac{d(x,y)}{\sqrt{m}}. \end{aligned}$$

Now, we can assume $d(x,y) \leq \varepsilon$.Let p and q are on the boundary of M_L such that p, x, y, q are on the straight line $\overline{xy}$ and are in order. Without loss of generality, we assume p is closer to x. Then, we have that $p \in M_L \cap (x - M_L)$, equivalently, we have that $\left|a_i^T p - a_i^T x\right| \leq a_i^T x - b_i$ for all i and hence $\|A_x(p-x)\|_\infty \leq 1$. Therefore, we have that

$$\|A_x(p-x)\|_2 \leq \sqrt{m}\|A_x(p-x)\|_\infty \leq \sqrt{m}.$$

Since p, x, y are on the same line, we have that

$$\frac{\|x-y\|_2\|p-q\|_2}{\|p-x\|_2\|y-q\|_2} \geq \frac{\|x-y\|_2}{\|p-x\|_2} = \frac{\|A_x(x-y)\|_2}{\|A_x(p-x)\|_2} \geq \frac{\|A_x(x-y)\|_2}{\sqrt{m}}.$$

Since $d(x,y) < \varepsilon$, Lemma 3.1 in [214] shows that

$$d(x,y) \leq -\log\left(1 - \|A_x(x-y)\|_2\right).$$

Hence, we have that $\|A_x(x-y)\|_2 \geq (1 - O(\varepsilon))d(x,y)$ and hence

$$d_H(x,y) = \frac{\|x-y\|_2\|p-q\|_2}{\|p-x\|_2\|y-q\|_2} \geq (1 - O(\varepsilon))\frac{d(x,y)}{\sqrt{m}}.$$

□

■ 14.5.2 Geodesic Walk on M_L

Recall that the geodesic walk is given by

$$x^{(j+1)} = \exp_{x^{(j)}}(\sqrt{h}w + \frac{h}{2}\mu(x^{(j)}))$$

where $w \sim N(0, I)$ (in the local coordinates at $x^{(j)}$). In many proofs in this section, we consider the geodesic γ from $x^{(j)}$ to $x^{(j+1)}$ with the initial velocity

$$\gamma'(0) = \frac{w}{\sqrt{n}} + \frac{1}{2}\sqrt{\frac{h}{n}}\mu(x).$$

The scaling is to make the speed of geodesic γ close to one. Since w is a Gaussian vector, we have that $0.9 \leq \|\gamma'(0)\|_{\gamma(0)} \leq 1.1$ with high probability. Due to this rescaling, the geodesic is defined from 0 to $\ell = \sqrt{nh}$.

We often work on the Euclidean coordinates. In this case, the geodesic walk is given by the following formula:

Lemma 14.5.7. *Given $x \in M_L$ and step size $h > 0$, one step of the geodesic walk starting at x is given by the solution $\gamma(\ell)$ of the following geodesic equation*

$$\begin{aligned} \gamma''(t) &= \left(A_\gamma^T A_\gamma\right)^{-1} A_\gamma^T s_{\gamma'}^2 \text{ for } 0 \leq t \leq \ell \\ \gamma'(0) &= \frac{w}{\sqrt{n}} + \frac{1}{2}\sqrt{\frac{h}{n}}\mu(x) \\ \gamma(0) &= x \end{aligned}$$

where $\ell = \sqrt{nh}$, $w \sim N(0, (A_x^T A_x)^{-1})$ and $\mu(x) = \left(A_x^T A_x\right)^{-1} A_x^T \sigma_x$.

Proof. The geodesic equation is given by Lemma 14.5.3. From (14.2), the drift term is given by

$$\begin{aligned} \mu_i(x) &= \frac{1}{2}\sum_{j=1}^n \frac{\partial}{\partial x_j}\left(\left(\nabla^2\phi(X_t)\right)^{-1}\right)_{ij} = \frac{1}{2}\sum_{j=1}^n \frac{\partial}{\partial x_j}\left(\left(A_x^T A_x\right)^{-1}\right)_{ij} \\ &= \sum_j e_i^T (A_x^T A_x)^{-1} A_x^T S_{x,e_j} A_x (A_x^T A_x)^{-1} e_j \\ &= \sum_j \sum_k V_{ik} e_k^T A_x e_j V_{jk} \quad \text{where } V = (A_x^T A_x)^{-1} A_x^T \\ &= \sum_k V_{ik} e_k^T A_x \sum_j e_j (e_k^T A_x (A_x^T A_x)^{-1} e_j) \\ &= \sum_k V_{ik} e_k^T A_x \sum_j e_j e_j^T (A_x^T A_x)^{-1} A_x^T e_k \\ &= \sum_k V_{ik} e_k^T A_x (A_x^T A_x)^{-1} A_x^T e_k \\ &= V_i \sigma_x = e_i^T (A_x^T A_x)^{-1} A_x^T \sigma_x. \end{aligned}$$

□

14.5.3 Randomness and Stability of the Geodesic

In this and subsequent sections, we will frequently use the following elementary calculus facts (using only the chain/product rules and formula for derivative of inverse of a matrix):

$$\begin{aligned} \frac{dA_x}{dt} &= -S_{\gamma'} A_x \\ \frac{dP_x}{dt} &= -S_{\gamma'} P_x - P_x S_{\gamma'} + 2P_x S_{\gamma'} P_x \\ \frac{dS_{\gamma'}}{dt} &= \text{Diag}(-S_{\gamma'} A_x \gamma' + A_x \gamma'') = -S_{\gamma'}^2 + \text{Diag}(P_x) S_{\gamma'}^2 \\ \frac{d\sigma_x}{dt} &= \text{Diag}(\frac{dP_x}{dt}) \end{aligned}$$

We also use these matrix inequalities: $\mathrm{Tr}(AB) = \mathrm{Tr}(BA)$, $\mathrm{Tr}(PAP) \leq \mathrm{Tr}(A)$ for any psd matrix A; $\mathrm{Tr}(ABA^T) \leq \mathrm{Tr}(AZA^T)$ for any $B \preceq Z$; the Cauchy-Schwartz, namely, $\mathrm{Tr}(AB) \leq \mathrm{Tr}(AA^T)^{\frac{1}{2}}\mathrm{Tr}(B^TB)^{\frac{1}{2}}$. We also use $P^2 = P$ since P is a projection matrix.

We note that $\|\gamma'(0)\|_4$ is small because it is the sum of a random vector plus a small drift term. Here, we would like to prove that the geodesic is stable in ℓ^4 norm. This fact is very important in the proof and hence we first establish this here.

Lemma 14.5.8. *Let γ be a geodesic in M_L starting at x. Let $v_4 = \|s_{\gamma'(0)}\|_4$. Then, for $0 \leq t \leq \frac{1}{10v_4}$, we have that*

1. $\|s_{\gamma'(t)}\|_4 \leq 1.25v_4$.
2. $\|\gamma''\|_\gamma^2 \leq 3v_4^4$.

Proof. Let $u(t) = \|s_{\gamma'(t)}\|_4$. Then, we have

$$\begin{aligned} \frac{du}{dt} &\leq \|\frac{d}{dt}\left(A_\gamma \gamma'\right)\|_4 \\ &= \|A_\gamma \gamma'' - \left(A_\gamma \gamma'\right)^2\|_4 \\ &\leq \|A_\gamma \gamma''\|_4 + u^2(t). \end{aligned} \tag{14.21}$$

Under the Euclidean coordinates, the geodesic equation is given by

$$\gamma'' = \left(A_\gamma^T A_\gamma\right)^{-1} A_\gamma^T s_{\gamma'}^2.$$

Hence, we have that

$$\begin{aligned} \|\gamma''\|_\gamma^2 &= \left(s_{\gamma'}^2\right)^T A_\gamma \left(A_\gamma^T A_\gamma\right)^{-1}\left(A_\gamma^T A_\gamma\right)\left(A_\gamma^T A_\gamma\right)^{-1} A_\gamma^T s_{\gamma'}^2 \\ &\leq \sum_i (s_{\gamma'}^4)_i = u^4(t). \end{aligned} \tag{14.22}$$

Therefore, we have

$$\|A_\gamma \gamma''\|_4 \leq \|A_\gamma \gamma''\|_2 \leq u^2(t).$$

Plugging it into (14.21), we have that

$$\frac{du}{dt} \leq u^2(t) + u^2(t) \leq 2u^2(t).$$

Since $u(0) = v_4$, for all $t \geq 0$, we have

$$u(t) \leq \frac{v_4}{1 - 2v_4 t}. \tag{14.23}$$

For $0 \leq t \leq \frac{1}{10v_4}$, we have $u(t) \leq 1.25v_4$ and this gives the first inequality. Using (14.22), we get the second inequality. □

Lemma 14.5.9. *For $p \geq 1$, we have*

$$P_{x\sim N(0,I)}\left(\|Ax\|_p^p \leq \left(\left(\frac{2^{p/2}\Gamma(\frac{p+1}{2})}{\sqrt{\pi}}\sum_i \|a_i\|_2^p\right)^{1/p} + \|A\|_{2\to p}t\right)^p\right) \leq 1 - \exp\left(-\frac{t^2}{2}\right).$$

In particular, we have

$$P_{x\sim N(0,I)}\left(\|Ax\|_2^2 \leq \left(\|A\|_F + \|A\|_2 t\right)^2\right) \leq 1 - \exp\left(-\frac{t^2}{2}\right)$$

and

$$P_{x\sim N(0,I)}\left(\|Ax\|_4^4 \le \left(\left(3\sum_i \|a_i\|_2^4\right)^{1/4} + \|A\|_{2\to 4}t\right)^4\right) \le 1-\exp\left(-\frac{t^2}{2}\right).$$

Proof. Let $F(x) = \|Ax\|_p$. Since $|F(x)-F(y)| \le \|A\|_{2\to p}\|x-y\|_2$, Gaussian concentration shows that

$$P\left(F(x) \le \mathbf{E}F(x) + \|A\|_{2\to p}t\right) \le 1-\exp\left(-\frac{t^2}{2}\right).$$

Since x^p is convex, we have that

$$\begin{aligned}\mathbf{E}\|Ax\|_p &\le \left(\mathbf{E}\|Ax\|_p^p\right)^{1/p} = \left(\sum_i \mathbf{E}\left|a_i^T x\right|^p\right)^{1/p}\\ &= \left(\mathbf{E}_{t\sim N(0,1)}|t|^p \sum_i \|a_i\|_2^p\right)^{1/p}\\ &= \left(\frac{2^{p/2}\Gamma(\frac{p+1}{2})}{\sqrt{\pi}}\sum_i \|a_i\|_2^p\right)^{1/p}.\end{aligned}$$

Hence, we have that

$$P\left(\|Ax\|_p^p \le \left(\left(\frac{2^{p/2}\Gamma(\frac{p+1}{2})}{\sqrt{\pi}}\sum_i \|a_i\|_2^p\right)^{1/p} + \|A\|_{2\to p}t\right)^p\right) \le 1-\exp\left(-\frac{t^2}{2}\right).$$

□

Using the lemma above, we prove that within small time, the direction of geodesic is a random vector plus a small vector.

Lemma 14.5.10. *Let γ given by a geodesic walk on M_L with step size h satisfying $h \le \frac{1}{256\sqrt{n}}$. Let u be the Gaussian part of the initial velocity $\gamma'(0)$. If $\|A_{\gamma(0)}u\|_2 \le \frac{3}{2}$ and $\|A_\gamma u\|_4 \le \frac{1.55}{n^{1/4}}$, then for any $0 \le t \le \ell$,*

1. $\|A_\gamma \gamma'\|_2 \le 2$.
2. $\|A_\gamma \gamma'\|_4 \le 2n^{-1/4}$.
3. $\|\gamma''\|_\gamma^2 \le 15n^{-1}$.
4. $\|A_\gamma \gamma'\|_\infty \le 6\sqrt{h}$.

Furthermore, $\|A_{\gamma(0)}u\|_2 \le \frac{3}{2}$ and $\|A_\gamma u\|_4 \le \frac{1.55}{n^{1/4}}$ holds with probability at least $1-\exp(-\sqrt{n}/100)$.

Proof. From the definition of the geodesic walk (Lemma 14.5.7), we have that

$$\gamma'(0) = u + v$$

where $u \sim N\left(0, \frac{1}{n}\left(A_\gamma^T A_\gamma\right)^{-1}\right)$ and $v = \frac{1}{2}\sqrt{\frac{h}{n}}\left(A_\gamma^T A_\gamma\right)^{-1} A_\gamma^T \sigma_\gamma$. Note that

$$\begin{aligned}\|A_\gamma \gamma'(0)\|_2 &\leq \|A_\gamma u\|_2 + \|\frac{1}{2}\sqrt{\frac{h}{n}} A_\gamma \left(A_\gamma^T A_\gamma\right)^{-1} A_\gamma^T \sigma_\gamma\|_2 \\ &= \|A_\gamma u\|_2 + \frac{1}{2}\sqrt{\frac{h}{n}}\sqrt{\sigma_\gamma^T A_\gamma \left(A_\gamma^T A_\gamma\right)^{-1} A_\gamma^T \sigma_\gamma}.\end{aligned}$$

Note that $A_\gamma u = Bx$ where $B = \frac{1}{\sqrt{n}} A_\gamma \left(A_\gamma^T A_\gamma\right)^{-1/2}$ and $x \sim N(0, I)$. Note that $\|Bx\|_2^2 = \frac{1}{n}\|x\|_2^2$ for all x and hence $\|B\|_F^2 = 1$ and $\|B\|_{2\to 2} = \frac{1}{\sqrt{n}}$. Hence, Lemma 14.5.9 shows that

$$P\left(\|A_\gamma u\|_2^2 \leq \left(1 + \frac{t}{\sqrt{n}}\right)^2\right) \leq 1 - \exp\left(-\frac{t^2}{2}\right)$$

Hence, we have

$$\|A_\gamma \gamma'\|_2 \leq \frac{3}{2} + \frac{1}{2}\sqrt{h} \leq 2$$

with probability $1 - \exp(-n/8)$. Since geodesic preserves the speed, this gives the first inequality.

Next, we note that $\sum_i \|e_i^T B\|_2^4 = \frac{1}{n^2}\sum_i (\sigma_\gamma)_i^2 \leq \frac{1}{n}$ and $\|B\|_{2\to 4} \leq \|B\|_{2\to 2} = \frac{1}{\sqrt{n}}$. Hence, Lemma 14.5.9 shows that

$$P\left(\|A_\gamma u\|_4^4 \leq \left(\left(\frac{3}{n}\right)^{1/4} + \frac{t}{\sqrt{n}}\right)^4\right) \leq 1 - \exp\left(-\frac{t^2}{2}\right).$$

Hence, we have that

$$v_4 \stackrel{\text{def}}{=} \|A_\gamma \gamma'(0)\|_4 \leq \frac{1.55}{n^{1/4}} + \frac{1}{2}\sqrt{h} \leq \frac{1.6}{n^{1/4}}$$

with probability at least $1 - \exp(-\sqrt{n}/100)$. Since $\ell = \sqrt{nh} \leq \frac{n^{1/4}}{16} = \frac{1}{10 v_4}$, Lemma 14.5.8 shows the middle two inequalities.

Lemma 14.5.8 shows that $\|A_\gamma \gamma''\|_\infty \leq \|\gamma''\|_{\gamma(t)} \leq \sqrt{3} v_4^2 \leq 5 n^{-1/2}$ and hence

$$\begin{aligned}\|A_{\gamma(t)}\gamma'(t)\|_\infty &\leq \|A_\gamma \gamma'(0)\|_\infty + \int_0^\ell \|A_{\gamma(t)}\gamma''(t)\|_\infty dt \\ &\leq \sqrt{h} + 5\ell n^{-1/2} \\ &= 6\sqrt{h}.\end{aligned}$$

□

■ 14.5.4 Stability of Drift

We begin with D_0 and D_1.

Lemma 14.5.11. *For any $x \in M_L$, we have that $\|\mu(x)\|_x^2 \leq n$. Hence, $D_0 \leq \sqrt{n}$.*

Proof. We have

$$\|\mu(x)\|_x^2 = \sigma_x^T A_x \left(A_x^T A_x\right)^{-1} A_x \sigma_x \leq \sum_i (\sigma_x)_i^2 \leq n.$$

□

Lemma 14.5.12. *Let $\gamma(t)$ be a geodesic walk on M_L with step size h satisfying $\frac{1}{n} \le h \le \frac{1}{256\sqrt{n}}$. Then, we have that*

$$D_1 = \sup_{0\le t\le \ell} \left| \frac{d}{dt} \|\mu(\gamma(t))\|^2 \right| \le 32n\sqrt{h}.$$

with probability at least $1 - \exp(-\sqrt{n}/100)$ over the choice of the next step $\gamma \in \widetilde{\Xi}_x$ for any $x \in M_L$.

Proof. We want to bound $\left| \|\mu(x)\|_x^2 - \|\mu(y)\|_y^2 \right|$. Note that

$$\begin{aligned} \|\mu(\gamma(t))\|_\gamma^2 &= \sigma_\gamma^T A_\gamma \left(A_\gamma^T A_\gamma\right)^{-1} A_\gamma \sigma_\gamma \\ &= 1^T \mathrm{Diag}(P_\gamma) P_\gamma \mathrm{Diag}(P_\gamma) 1. \end{aligned}$$

Using $\frac{d}{dt} P_\gamma = -S_{\gamma'} P_\gamma - P_\gamma S_{\gamma'} + 2P_\gamma S_{\gamma'} P_\gamma$, we have

$$\begin{aligned} \frac{d}{dt}\|\mu(\gamma(t))\|_\gamma^2 &= -2 \cdot 1^T \mathrm{Diag}(S_{\gamma'} P_\gamma) P_\gamma \mathrm{Diag}(P_\gamma) 1 \\ &\quad -2 \cdot 1^T \mathrm{Diag}(P_\gamma S_{\gamma'}) P_\gamma \mathrm{Diag}(P_\gamma) 1 \\ &\quad +4 \cdot 1^T \mathrm{Diag}(P_\gamma S_{\gamma'} P_\gamma) P_\gamma \mathrm{Diag}(P_\gamma) 1 \\ &\quad -1^T \mathrm{Diag}(P_\gamma) S_{\gamma'} P_\gamma \mathrm{Diag}(P_\gamma) 1 \\ &\quad -1^T \mathrm{Diag}(P_\gamma) P_\gamma S_{\gamma'} \mathrm{Diag}(P_\gamma) 1 \\ &\quad +2 \cdot 1^T \mathrm{Diag}(P_\gamma) P_\gamma S_{\gamma'} P_\gamma \mathrm{Diag}(P_\gamma) 1 \\ &= -2 \cdot 1^T \mathrm{Diag}(S_{\gamma'} P_\gamma) P_\gamma \mathrm{Diag}(P_\gamma) 1 \\ &\quad -4 \cdot 1^T \mathrm{Diag}(P_\gamma) S_{\gamma'} P_\gamma \mathrm{Diag}(P_\gamma) 1 \\ &\quad +4 \cdot 1^T \mathrm{Diag}(P_\gamma S_{\gamma'} P_\gamma) P_\gamma \mathrm{Diag}(P_\gamma) 1 \\ &\quad +2 \cdot 1^T \mathrm{Diag}(P_\gamma) P_\gamma S_{\gamma'} P_\gamma \mathrm{Diag}(P_\gamma) 1. \end{aligned}$$

Now, we note that

$$\begin{aligned} \left|1^T \mathrm{Diag}(S_{\gamma'} P_\gamma) P_\gamma \mathrm{Diag}(P_\gamma) 1\right| &\le \|\gamma'\|_\gamma \sqrt{n}, \\ \left|1^T \mathrm{Diag}(P_\gamma) S_{\gamma'} P_\gamma \mathrm{Diag}(P_\gamma) 1\right| &\le \|\gamma'\|_\gamma \sqrt{n}, \\ \left|1^T \mathrm{Diag}(P_\gamma S_{\gamma'} P_\gamma) P_\gamma \mathrm{Diag}(P_\gamma) 1\right| &\le \|\gamma'\|_\gamma \sqrt{n}, \\ \left|1^T \mathrm{Diag}(P_\gamma) P_\gamma S_{\gamma'} P_\gamma \mathrm{Diag}(P_\gamma) 1\right| &\le \|S_{\gamma'}\|_\infty n. \end{aligned}$$

Since $\|\gamma'\|_\gamma \le 2$ and $\|A_\gamma \gamma'\|_\infty \le 6\sqrt{h}$ (Lemma 14.5.10), we have

$$D_1 = \sup \left| \frac{d}{dt} \|\mu(\gamma(t))\|^2 \right| \le 20\sqrt{n} + 12n\sqrt{h} \le 32n\sqrt{h}.$$

□

■ 14.5.5 Smoothness of the metric

Lemma 14.5.13. *Let γ given by a geodesic walk on M_L with step size h satisfying $\frac{1}{n} \le h \le \frac{1}{256\sqrt{n}}$. Let $f(t) = \log\det(A_{\gamma(t)}^T A_{\gamma(t)})$. Then, we have*

$$G_1 = \sup_{0\le t\le \ell} \left| f'''(t) \right| \le 1000\sqrt{h}$$

with probability at least $1 - \exp(-\sqrt{n}/100)$.

Proof. Note that

$$f'(t) = -2\mathrm{Tr}(A_\gamma^T A_\gamma)^{-1} A_\gamma^T S_{\gamma'} A_\gamma.$$

Hence, we have

$$\begin{aligned} f''(t) &= -4\mathrm{Tr}(A_\gamma^T A_\gamma)^{-1} A_\gamma^T S_{\gamma'} A_\gamma (A_\gamma^T A_\gamma)^{-1} A_\gamma^T S_{\gamma'} A_\gamma \\ &\quad +6\mathrm{Tr}(A_\gamma^T A_\gamma)^{-1} A_\gamma^T S_{\gamma'}^2 A_\gamma \\ &\quad -2\mathrm{Tr}(A_\gamma^T A_\gamma)^{-1} A_\gamma^T S_{\gamma''} A_\gamma. \end{aligned}$$

So, we have

$$\begin{aligned} f'''(t) &= -16\mathrm{Tr}(A_\gamma^T A_\gamma)^{-1} A_\gamma^T S_{\gamma'} A_\gamma (A_\gamma^T A_\gamma)^{-1} A_\gamma^T S_{\gamma'} A_\gamma (A_\gamma^T A_\gamma)^{-1} A_\gamma^T S_{\gamma'} A_\gamma \\ &\quad +24\mathrm{Tr}(A_\gamma^T A_\gamma)^{-1} A_\gamma^T S_{\gamma'}^2 A_\gamma (A_\gamma^T A_\gamma)^{-1} A_\gamma^T S_{\gamma'} A_\gamma \\ &\quad -8\mathrm{Tr}(A_\gamma^T A_\gamma)^{-1} A_\gamma^T S_{\gamma''} A_\gamma (A_\gamma^T A_\gamma)^{-1} A_\gamma^T S_{\gamma'} A_\gamma \\ &\quad +12\mathrm{Tr}(A_\gamma^T A_\gamma)^{-1} A_\gamma^T S_{\gamma'} A_\gamma (A_\gamma^T A_\gamma)^{-1} A_\gamma^T S_{\gamma'}^2 A_\gamma \\ &\quad -24\mathrm{Tr}(A_\gamma^T A_\gamma)^{-1} A_\gamma^T S_{\gamma'}^3 A_\gamma \\ &\quad +12\mathrm{Tr}(A_\gamma^T A_\gamma)^{-1} A_\gamma^T S_{\gamma'} S_{\gamma''} A_\gamma \\ &\quad -4\mathrm{Tr}(A_\gamma^T A_\gamma)^{-1} A_\gamma^T S_{\gamma'} A_\gamma (A_\gamma^T A_\gamma)^{-1} A_\gamma^T S_{\gamma''} A_\gamma \\ &\quad +4\mathrm{Tr}(A_\gamma^T A_\gamma)^{-1} A_\gamma^T S_{\gamma''} S_{\gamma'} A_\gamma \\ &\quad -2\mathrm{Tr}(A_\gamma^T A_\gamma)^{-1} A_\gamma^T \frac{d}{dt} S_{\gamma''} A_\gamma \\ &= -16\mathrm{Tr} P_\gamma S_{\gamma'} P_\gamma S_{\gamma'} P_\gamma S_{\gamma'} + 36\mathrm{Tr} P_\gamma S_{\gamma'}^2 P_\gamma S_{\gamma'} \\ &\quad -12\mathrm{Tr} P_\gamma S_{\gamma''} P_\gamma S_{\gamma'} - 24\mathrm{Tr} P_\gamma S_{\gamma'}^3 \\ &\quad +16\mathrm{Tr} P_\gamma S_{\gamma'} S_{\gamma''} - 2\mathrm{Tr} P_\gamma \frac{d}{dt} S_{\gamma''}. \end{aligned}$$

Hence, we have

$$\left|f'''(t)\right| \le (16+36+24)\sqrt{\mathrm{Tr} S_{\gamma'}^4 \mathrm{Tr} S_{\gamma'}^2} + (12+16)\sqrt{\mathrm{Tr} S_{\gamma''}^2 \mathrm{Tr} S_{\gamma'}^2} + 2\left|\mathrm{Tr} P_\gamma \frac{d}{dt} S_{\gamma''}\right|.$$

Since $\mathrm{Tr} S_{\gamma'}^2 \le 2$, $\mathrm{Tr} S_{\gamma'}^4 \le 16n^{-1}$, $\mathrm{Tr} S_{\gamma''}^2 \le 15n^{-1}$ (Lemma 14.5.10), we have

$$\begin{aligned} \left|f'''(t)\right| &\le 76\sqrt{2\cdot 16 n^{-1}} + 28\sqrt{2\cdot 15 n^{-1}} + 2\left|\mathrm{Tr} P_\gamma \frac{d}{dt} S_{\gamma''}\right| \\ &\le 600 n^{-1/2} + 2\left|\mathrm{Tr} P_\gamma \frac{d}{dt} S_{\gamma''}\right|. \end{aligned}$$

To bound the last term, we start with the geodesic equation: $s_{\gamma''} = P_\gamma s_{\gamma'}^2$. Since $\frac{d}{dt} P_\gamma = -S_{\gamma'} P_\gamma - P_\gamma S_{\gamma'} + 2P_\gamma S_{\gamma'} P_\gamma$, we have that

$$\begin{aligned} \frac{d}{dt} s_{\gamma''} &= -S_{\gamma'} P_\gamma s_{\gamma'}^2 - P_\gamma S_{\gamma'} s_{\gamma'}^2 + 2P_\gamma S_{\gamma'} P_\gamma s_{\gamma'}^2 \\ &\quad +2P_\gamma s_{\gamma'} s_{\gamma''} - 2P_\gamma s_{\gamma'}^3 \\ &= -S_{\gamma'} P_\gamma s_{\gamma'}^2 - 3P_\gamma s_{\gamma'}^3 + 2P_\gamma S_{\gamma'} P_\gamma s_{\gamma'}^2 + 2P_\gamma s_{\gamma'} s_{\gamma''} \\ &= -S_{\gamma'} P_\gamma s_{\gamma'}^2 - 3P_\gamma s_{\gamma'}^3 + 4P_\gamma S_{\gamma'} P_\gamma s_{\gamma'}^2, \end{aligned}$$

Hence, we have that

$$\begin{aligned}\left|\mathrm{Tr}P_\gamma \frac{d}{dt}S_{\gamma''}\right| &\leq \left|\sigma_\gamma^T S_{\gamma'}P_\gamma s_{\gamma'}^2\right| + 3\left|\sigma_\gamma^T P_\gamma s_{\gamma'}^3\right| + 4\left|\sigma_\gamma^T P_\gamma S_{\gamma'}P_\gamma s_{\gamma'}^2\right| \\ &\leq \left|(S_{\gamma'}\sigma_\gamma)^T P_\gamma s_{\gamma'}^2\right| + 3\left|(S_{\gamma'}P_\gamma\sigma_\gamma)^T s_{\gamma'}^2\right| + 4(S_{\gamma'}P_\gamma\sigma_\gamma)^T(P_\gamma s_{\gamma'}^2) \\ &\leq \sqrt{\sum s_{\gamma'}^2 \sum s_{\gamma'}^4} + 28n^{-1/2}\sqrt{\sigma_\gamma^T P_\gamma S_{\gamma'}^2 P_\gamma \sigma_\gamma} \\ &\leq 8n^{-1/2} + 168\sqrt{h} \leq 180\sqrt{h}.\end{aligned}$$

Hence, we have that $|f'''(t)| \leq 600n^{-1/2} + 360\sqrt{h} \leq 1000\sqrt{h}$. □

■ 14.5.6 Stability of Jacobi Field

Now, we give a bound on R_1.

Lemma 14.5.14. *Let γ given by a geodesic walk on M_L with step size h satisfying $\frac{1}{n} \leq h \leq \frac{1}{256\sqrt{n}}$. With probability at least $1 - \exp(-\sqrt{n}/100)$ over the choice of the next step $\gamma \in \widetilde{\Xi}_x$, we have that*

$$R_1 = \sup_{0\leq t\leq \ell} \|R(t)\|_F \leq \frac{8}{\sqrt{n}}$$

where $R(t)_{ij} = \langle R(X_i(t), \gamma'(t))\gamma'(t), X_j(t)\rangle$ and $\{X_i\}_{i=1}^n$ is any orthogonal basis at $\gamma(t)$.

Proof. Let $\overline{R(t)}$ such that $\langle R(u, \gamma'(t))\gamma'(t), v\rangle = u^T\overline{R(t)}v$. Lemma 14.5.4 shows that

$$u^T\overline{R(t)}v = s_u^T S_{\gamma'}A_\gamma(A_\gamma^TA_\gamma)^{-1}A_\gamma^T S_{\gamma'}s_v - (s_v s_u)^T A_\gamma(A_\gamma^TA_\gamma)^{-1}A_\gamma^T s_{\gamma'}^2.$$

Hence, we have

$$\overline{R(t)} = A_\gamma^T S_{\gamma'}P_\gamma S_{\gamma'}A_\gamma - A_\gamma^T\mathrm{Diag}(P_\gamma s_{\gamma'}^2)A_\gamma.$$

Pick $X_i = (A^TA_\gamma)^{-1/2}e_i$. Then, we have

$$R(t) = (A_\gamma^TA_\gamma)^{-1/2}\left(A_\gamma^T S_{\gamma'}P_\gamma S_{\gamma'}A_\gamma - A_\gamma^T\mathrm{Diag}(P_\gamma s_{\gamma'}^2)A_\gamma\right)(A_\gamma^TA_\gamma)^{-1/2}.$$

The claim follows from Lemma 14.5.10 and the calculation

$$\begin{aligned}&\|R(t)\|_F^2 \\ \leq\ & 2\|(A_\gamma^TA_\gamma)^{-1/2}A_\gamma^T S_{\gamma'}P_\gamma S_{\gamma'}A_\gamma(A_\gamma^TA_\gamma)^{-1/2}\|_F^2 + 2\|(A_\gamma^TA_\gamma)^{-1/2}A_\gamma^T\mathrm{Diag}(P_\gamma s_{\gamma'}^2)A_\gamma(A_\gamma^TA_\gamma)^{-1/2}\|_F^2 \\ =\ & 2\mathrm{Tr}P_\gamma S_{\gamma'}P_\gamma S_{\gamma'}P_\gamma S_{\gamma'}P_\gamma S_{\gamma'} + 2\mathrm{Tr}P_\gamma\mathrm{Diag}(P_\gamma s_{\gamma'}^2)P_\gamma\mathrm{Diag}(P_\gamma s_{\gamma'}^2) \leq 4\|s_{\gamma'}\|_4^4.\end{aligned}$$

□

■ 14.5.7 Smoothness of One-Step Distribution

We begin with R_2.

Lemma 14.5.15. *Given a geodesic $\gamma(t)$ on M_L with step size h satisfying $\frac{1}{n} \leq h \leq \frac{1}{256\sqrt{n}}$, with probability at least $1 - \exp(-\sqrt{n}/100)$ over the choice of the next step $\gamma \in \widetilde{\Xi}_x$, for any t such that $0 \leq t \leq \ell$, any curve $c(s)$ starting from $\gamma(t)$ and any vector field $v(s)$ on $c(s)$, we have that*

$$\left|\frac{d}{ds}\,Ric(v(s))|_{s=0}\right| \leq 64\sqrt{nh}\|c'(0)\| + 6\sqrt{nh}\|D_s v(0)\|.$$

Therefore, $R_2 = O(\sqrt{nh})$.

Proof. By Lemma 14.5.4, we know that

$$\begin{aligned} Ric(v(s)) &= s^T_{c(s),v(s)} P^{(2)}_{c(s)} s_{c(s),v(s)} - \sigma^T_{c(s)} P_{c(s)} s^2_{c(s),v(s)} \\ &= \mathrm{Tr}(S_{c(s),v(s)} P_{c(s)} S_{c(s),v(s)} P_{c(s)}) - \mathrm{Tr}(\mathrm{Diag}(P_{c(s)} s^2_{c(s),v(s)}) P_{c(s)}). \end{aligned}$$

For simplicity, we suppress the parameter s and hence, we have

$$Ric(v) = \mathrm{Tr}(S_{c,v} P_c S_{c,v} P_c) - \mathrm{Tr}(\mathrm{Diag}(P_c S^2_{c,v}) P_c)$$

Now, we use that

$$\begin{aligned} \frac{d}{ds} P_c &= -S_{c'} P_c - P_c S_{c'} + 2 P_c S_{c'} P_c, \\ \frac{d}{ds} S_{c,v} &= -S_{c'} S_{c,v} + S_{c,v'} \end{aligned}$$

and get

$$\begin{aligned} & \frac{d}{ds} Ric(v) \\ = \; & -2\mathrm{Tr}(S_{c,v} S_{c'} P_c S_{c,v} P_c) - 2\mathrm{Tr}(S_{c,v} P_c S_{c'} S_{c,v} P_c) + 2\mathrm{Tr}(S_{c,v} P_c S_{c'} P_c S_{c,v} P_c) \\ & -2\mathrm{Tr}(S_{c'} S_{c,v} P_c S_{c,v} P_c) + 2\mathrm{Tr}(S_{c,v'} P_c S_{c,v} P_c) \\ & +\mathrm{Tr}(\mathrm{Diag}(P_c s^2_{c,v}) S_{c'} P_c) + \mathrm{Tr}(\mathrm{Diag}(P_c s^2_{c,v}) P_c S_{c'}) - 2\mathrm{Tr}(\mathrm{Diag}(P_c s^2_{c,v}) P_c S_{c'} P_c) \\ & +\mathrm{Tr}(\mathrm{Diag}(P_c S_{c'} s^2_{c,v}) P_c) + \mathrm{Tr}(\mathrm{Diag}(S_{c'} P_c s^2_{c,v}) P_c) - 2\mathrm{Tr}(\mathrm{Diag}(P_c S_{c'} P_c s^2_{c,v}) P_c) \\ & +2\mathrm{Tr}(\mathrm{Diag}(P_c S_{c,v} S_{c'} s_{c,v}) P_c) - 2\mathrm{Tr}(\mathrm{Diag}(P_c S_{c,v} s_{c,v'}) P_c) \\ = \; & -6\mathrm{Tr}(S_{c,v} S_{c'} P_c S_{c,v} P_c) + 2\mathrm{Tr}(S_{c,v} P_c S_{c'} P_c S_{c,v} P_c) + 2\mathrm{Tr}(S_{c,v'} P_c S_{c,v} P_c) \\ & +2\mathrm{Tr}(\mathrm{Diag}(P_c s^2_{c,v}) S_{c'} P_c) - 2\mathrm{Tr}(\mathrm{Diag}(P_c s^2_{c,v}) P_c S_{c'} P_c) \\ & +3\mathrm{Tr}(\mathrm{Diag}(P_c S_{c'} s^2_{c,v}) P_c) + \mathrm{Tr}(\mathrm{Diag}(S_{c'} P_c s^2_{c,v}) P_c) - 2\mathrm{Tr}(\mathrm{Diag}(P_c S_{c'} P_c s^2_{c,v}) P_c) \\ & -2\mathrm{Tr}(\mathrm{Diag}(P_c S_{c,v} s_{c,v'}) P_c). \end{aligned}$$

Let $\frac{d}{ds} Ric(v) = (1) + (2)$ where (1) is the sum of all terms not involving v' and (2) is the sum of other terms.

For the first term (1), let $L = \|s_{c'(s)}\|_2$. Then, we have that

$$\begin{aligned} |(1)| \leq \; & 6\left|\mathrm{Tr}(S_{c,v} S_{c'} P_c S_{c,v} P_c)\right| + 2\left|\mathrm{Tr}(S_{c,v} P_c S_{c'} P_c S_{c,v} P_c)\right| \\ & +2\left|\mathrm{Tr}(\mathrm{Diag}(P_c s^2_{c,v}) S_{c'} P_c)\right| + 2\left|\mathrm{Tr}(\mathrm{Diag}(P_c s^2_{c,v}) P_c S_{c'} P_c)\right| \\ & +3\left|\mathrm{Tr}(\mathrm{Diag}(P_c S_{c'} s^2_{c,v}) P_c)\right| + \left|\mathrm{Tr}(\mathrm{Diag}(S_{c'} P_c s^2_{c,v}) P_c)\right| + 2\left|\mathrm{Tr}(\mathrm{Diag}(P_c S_{c'} P_c s^2_{c,v}) P_c)\right| \\ \leq \; & 6L\sqrt{\sum_i (s_{c,v})_i^2}\sqrt{\sum_i (s_{c,v})_i^2} + 2L\left|\mathrm{Tr}(S_{c,v} P_c S_{c,v} P_c)\right| \\ & +2L\sqrt{\sum_i (s_{c,v})_i^4}\sqrt{\sum_i (P_c)_{ii}^2} + 2L\sqrt{\sum_i (s_{c,v})_i^4}\sqrt{\sum_i (P_c S_{c'} P_c)_{ii}^2} \\ & +3L\sqrt{\sum_i (s_{c,v})_i^4}\sqrt{\sum_i (P_c)_{ii}^2} + L\sqrt{\sum_i (s_{c,v})_i^4}\sqrt{\sum_i (P_c)_{ii}^2} + 2L\sqrt{\sum_i (s_{c,v})_i^4}\sqrt{\sum_i (P_c)_{ii}^2} \\ \leq \; & 8L\sqrt{\sum_i (s_{c,v})_i^2}\sqrt{\sum_i (s_{c,v})_i^2} + 10\sqrt{n} L\sqrt{\sum_i (s_{c,v})_i^4}. \end{aligned}$$

Since $v(0) = \gamma'(t)$, Lemma 14.5.10 shows that $\|s_{c,v}\|_2 \leq 2$ and $\|s_{c,v}\|_4 \leq 2n^{-1/4}$. Hence, we have

$$|(1)| \leq 52L.$$

For the second term (2), we note that

$$D_s v = \frac{dv}{ds} - \left(A_c^T A_c\right)^{-1} A_c^T S_{c'} s_{c,v}.$$

Therefore, we have

$$s_{c,v'} = A_c \left(D_s v\right) - A_c \left(A_c^T A_c\right)^{-1} A_c^T S_{c'} s_{c,v}$$

and hence

$$\begin{aligned} \|s_{c,v'}\|_2 &\leq \|D_s v\| + \|A_c \left(A_c^T A_c\right)^{-1} A_c^T S_{c'} s_{c,v}\|_2 \\ &\leq \|D_s v\| + L\|s_{c,v}\|_2 \\ &\leq 2L + \|D_s v\|. \end{aligned}$$

Also, note that Lemma 14.5.10 shows that $\|s_{c,v}\|_\infty \leq 6\sqrt{h}$. Hence, we have that

$$\begin{aligned} |(2)| &\leq 2\left|\mathrm{Tr}(S_{c,v'} P_c S_{c,v} P_c)\right| + 2\left|\mathrm{Tr}(\mathrm{Diag}(P_c S_{c,v} s_{c,v'}) P_c)\right| \\ &\leq 2\|s_{c,v'}\|_2 \|s_{c,v}\|_2 + 2\sqrt{n}\sqrt{\sum_i (s_{c,v'} s_{c,v})_i^2} \\ &\leq 4\left(2 + \|D_s v\|\right) + 2\sqrt{nh}\left(2 + \|D_s v\|\right) \\ &\leq 8 + 4\|D_s v\| + 4\sqrt{nh} + 2\sqrt{nh}\|D_s v\| \\ &\leq 12\sqrt{nh} + 6\sqrt{nh}\|D_s v\|. \end{aligned}$$

Therefore, we have

$$\begin{aligned} \left|\frac{d}{ds} Ric(v)\right| &\leq 52L + 12\sqrt{nh}L + 6\sqrt{nh}\|D_s v\| \\ &\leq 64\sqrt{nh}L + 6\sqrt{nh}\|D_s v\|. \end{aligned}$$

□

Finally, we bound D_2.

Lemma 14.5.16. *Given $x \in M_L$ and any curve $c(t)$ starting at x with unit speed. We have that*

$$\|\frac{D}{dt}\mu(c(t))\| \leq 9\sqrt{n}.$$

Hence, $D_2 \leq 9\sqrt{n}$.

Proof. Recall that $\frac{D}{dt}\mu(c(t)) = \frac{d\mu(c)}{dt} - (A_c^T A_c)^{-1} A_c^T S_{c,\mu} s_{c'}$. We bound the terms separately. For the second term, since c is an unit speed curve, we have that $\|s_{c'}\|_2 = 1$ and

$$\begin{aligned} \|(A_c^T A_c)^{-1} A_c^T S_{c'} A_c \mu(c)\|_c &\leq \|S_{c,\mu} s_{c'}\|_2 \leq \|S_{c,\mu}\|_\infty \\ &= \|A_c (A_c^T A_c)^{-1} A_c^T \sigma_c\|_\infty \leq \sqrt{n}. \end{aligned}$$

For the first term,

$$\begin{aligned}\frac{d}{dt}\left(A_c\mu(c)\right) &= \frac{d}{dt}\left(P_c\boldsymbol{diag}(P_c)\right)\\ &= -S_{c'}P_c\boldsymbol{diag}(P_c) - P_cS_{c'}\boldsymbol{diag}(P_c) + 2P_cS_{c'}P_c\boldsymbol{diag}(P_c)\\ &\quad -P_c\boldsymbol{diag}(S_{c'}P_c) - P_c\boldsymbol{diag}(P_cS_{c'}) + 2P_c\boldsymbol{diag}(P_cS_{c'}P_c)\\ &= -S_{c'}P_c\boldsymbol{diag}(P_c) - 2P_cS_{c'}\boldsymbol{diag}(P_c) + 2P_cS_{c'}P_c\boldsymbol{diag}(P_c)\\ &\quad -P_c\boldsymbol{diag}(P_cS_{c'}) + 2P_c\boldsymbol{diag}(P_cS_{c'}P_c).\end{aligned}$$

Using $\|s_{c'}\|_2 = 1$, we have

$$\begin{aligned}\|\frac{d}{dt}\mu(c)\|_c &\le \|S_{c'}A_c\mu(c)\|_2\\ &\quad +\|S_{c'}P_c\boldsymbol{diag}(P_c)\|_2 + 2\|P_cS_{c'}\boldsymbol{diag}(P_c)\|_2 + 2\|P_cS_{c'}P_c\boldsymbol{diag}(P_c)\|_2\\ &\quad +\|P_c\boldsymbol{diag}(P_cS_{c'})\|_2 + \|P_c\boldsymbol{diag}(P_cS_{c'}P_c)\|_2\\ &\le 8\sqrt{n}.\end{aligned}$$

Combining both terms, we have the result. □

14.5.8 Mixing time

Proof of Theorem 14.5.1. In the last previous sections, we proved that if $\frac{1}{n} \le h \le \frac{1}{256\sqrt{n}}$

1. $D_0 \le O(\sqrt{n})$ (Lemma 14.5.11)
2. $D_1 = O(n\sqrt{h})$ (Lemma 14.5.12)
3. $D_2 = O(\sqrt{n})$ (Lemma 14.5.16)
4. $G_1 = O(\sqrt{h})$ (Lemma 14.5.13)
5. $G_2 = O(\sqrt{m})$ (Lemma 14.5.6)
6. $R_1 = O(1/\sqrt{n})$ (Lemma 14.5.14)
7. $R_2 = O(\sqrt{nh})$ (Lemma 14.5.15)

It is easy to see that all these parameters decrease if h decreases. So, the condition $\frac{1}{n} \le h$ can be ignored. We only use randomness to prove that $\|A_{\gamma(0)}u\|_2 \le \frac{3}{2}$ and $\|A_\gamma u\|_4 \le \frac{1.55}{n^{1/4}}$ (Lemma 14.5.10) and they can be checked. These bounds hold with exponentially small failure probability. Hence, the measure of Ξ is large as claimed and the result follows. Theorem 14.4.2 implies that the walk has mixing time $O(G_2^2/h)$ as long as

$$h \le \min\left\{\frac{1}{(nD_0R_1)^{2/3}}, \frac{1}{D_2}, \frac{1}{nR_1}, \frac{1}{n^{1/3}D_1^{2/3}}, \frac{1}{nG_1^{2/3}}, \frac{1}{(nR_2)^{2/3}}\right\} \le \frac{1}{\Theta(n^{3/4})}.$$

□

14.5.9 Complex Analyticity of geodesics, parallel transport and Jacobi fields

To show geodesics, parallel transport and Jacobi fields on M_L are complex analytic, we note that the Cauchy–Kowalevski theorem shows that if a differential equation is complex analytic, then the equation has a unique complex analytic solution.

Theorem 14.5.17 (Simplified Version of Cauchy–Kowalevski theorem). *If f is a complex analytic function defined on a neighborhood of $(z_0, \alpha) \in \mathbb{C}^{n+1}$, then the problem*

$$\frac{dw}{dz} = f(z, w), \quad w(z_0) = \alpha,$$

has a unique complex analytic solution w defined on a neighborhood around z_0.

Similarly, for a complex analytic function f defined in a neighborhood of $(z_0, \alpha, \beta) \in \mathbb{C}^{2n+1}$, the ODE

$$\frac{d^2w}{dz^2} = f(z, w, \frac{dw}{dz}), \quad w(z_0) = \alpha, \quad \frac{dw}{dz}(z_0) = \beta$$

has a unique complex analytic solution w defined in a neighborhood around z_0.

As we see in section 14.5.1, the equations for geodesic, parallel transport and Jacobi field involve only rational functions. Since rational functions are complex analytic, the Cauchy-Kowalevski theorem shows that geodesic, orthogonal frame and Jacobi field are complex analytic.

Lemma 14.5.18. *Geodesics, parallel transport and Jacobi fields are complex analytic for the Hessian manifold induced by the logarithmic barrier.*

14.5.9.1 Radius of Convergence

Next we bound the higher-order derivatives of geodesic, parallel transport and Jacobi field, using techniques developed in Section 14.7 to. Since these are complex analytic, we get a bound for their radius of convergence.

The purpose of these derivative bounds is to show that the solutions of the corresponding ODEs are well-approximated by low-degree polynomials, where the degree of the polynomial grows as $\log\left(\frac{1}{\varepsilon}\right)$ for desired accuracy ε. The bound on the degree also implies that the Collocation method for solving ODEs is efficient (roughly matrix multiplication time).

14.5.9.2 Geodesic

Motivated from the geodesic equation under Euclidean coordinate (14.19), we define the following auxiliary function

$$\overline{F}_\eta(y, x, t) = \left(A_{x+t\eta}^T A_{x+t\eta}\right)^{-1} A_{x+t\eta}^T s_{x+t\eta, y+\eta}^2. \tag{14.24}$$

The derivative bounds on geodesic rely on the smoothness of this auxiliary function.

Lemma 14.5.19. *Under the normalization $A^T A = I$, $S_x = I$, we have that*

$$\overline{F}_\eta(y, x, t) \leq_{(0,x,0)} \frac{4\left(\|A\eta\|_4 + 1\right)^2}{1 - \max\left(8 + 8\|A\eta\|_\infty, 1\right) t}$$

where $\overline{F}_\eta$ is defined in (14.24).

Proof. By the assumption that $S_x = I$, we have that $\|S_x\|_2 = 1$. Using $A^T A = I$, we have that

$$\begin{aligned} \|DS_{x+t\eta}(x,t)[(d_x, d_t)]\|_2 &= \|\mathrm{Diag}(Ad_x + d_t A\eta)\|_2 \\ &\leq \|Ad_x\|_\infty + \|d_t A\eta\|_\infty \\ &\leq \|d_x\|_2 + |d_t|\,\|A\eta\|_\infty \\ &\leq (1 + \|A\eta\|_\infty)\|(d_x, d_t)\|_2. \end{aligned}$$

Let $\beta = 1 + \|A\eta\|_\infty$. Then, we have that $S_{x+t\eta} \leq_{(x,t=0)} 1 + \beta t$.

By using the inverse formula (Lemma 14.7.3),

$$S_{x+t\eta}^{-1} \leq_{(x,0)} \frac{1}{1-\beta t}.$$

Using $A^TA = I$, we have that $A \leq_{(x,0)} 1$ and hence product formula (Lemma 14.7.3) shows that

$$A_{x+t\eta} = S_{x+t\eta}^{-1} A \leq_{(x,0)} \frac{1}{1-\beta t}.$$

Since $A^TA = I$, we have that $AA^T \preceq I$ and hence $A_{x+t\eta}^T \leq_{(x,0)} \frac{1}{1-\beta t}$. Therefore, we have that

$$A_{x+t\eta}^T A_{x+t\eta} \leq_{(x,0)} \frac{1}{(1-\beta t)^2}.$$

By using the inverse formula again,

$$\left(A_{x+t\eta}^T A_{x+t\eta}\right)^{-1} \leq_{(x,0)} \frac{1}{2 - \frac{1}{(1-\beta t)^2}} = \frac{(1-\beta t)^2}{2(1-\beta t)^2 - 1}.$$

Hence, we have that

$$\left(A_{x+t\eta}^T A_{x+t\eta}\right)^{-1} A_{x+t\eta}^T \leq_{(x,0)} \frac{(1-\beta t)^2}{2(1-\beta t)^2 - 1} \frac{1}{1-\beta t} = \frac{1-\beta t}{2(1-\beta t)^2 - 1}.$$

Now, we consider the function $H(y) = (A(y+\eta))^2$. Note that

$$\begin{aligned} \|H(y)\|_2 &\leq \|A(y+\eta)\|_4^2, \\ \|DH(y)[d]\|_2 &= 2\|(A(y+\eta))Ad\|_2 \leq 2\|A(y+\eta)\|_4 \|d\|_2, \\ \|D^2H(y)[d,d]\|_2 &\leq 2\|(Ad)^2\|_2 \leq 2\|d\|_2^2. \end{aligned}$$

Therefore, we have that

$$H \leq_{y=0} \|A\eta\|_4^2 + 2\|A\eta\|_4 t + t^2 = (\|A\eta\|_4 + t)^2.$$

Hence, we have that

$$\overline{F}_\eta(y,x,t) = \left(A_{x+t\eta}^T A_{x+t\eta}\right)^{-1} A_{x+t\eta}^T s_{x+t\eta,y+\eta}^2 \leq_{(0,x,0)} \frac{(1-\beta t)\left(\|A\eta\|_4 + t\right)^2}{2(1-\beta t)^2 - 1} \frac{1}{(1-\beta t)^2}.$$

Let $\phi(t) = \frac{(\|A\eta\|_4 + t)^2}{(2(1-\beta t)^2 - 1)(1-\beta t)}$ and we write $\phi(t) = \sum_{k=0}^{\infty} a_k t^k$. For any $|z| = \frac{1}{8}\min\left(\frac{1}{\beta}, \|A\eta\|_4 + 8\right)$, we have that

$$\begin{aligned} |\phi(z)| &\leq 3\left(\|A\eta\|_4 + \frac{\|A\eta\|_4}{8} + 1\right)^2 \\ &\leq 4\left(\|A\eta\|_4 + 1\right)^2. \end{aligned}$$

Theorem 14.2.11 shows that

$$\begin{aligned} |a_k| &\leq 4\left(\|A\eta\|_4 + 1\right)^2 \left(8\max\left(\beta, \left(\|A\eta\|_4 + 8\right)^{-1}\right)\right)^k \\ &\leq 4\left(\|A\eta\|_4 + 1\right)^2 \left(\max\left(8\beta, 1\right)\right)^k \end{aligned}$$

Hence, we can instead bound $\overline{F}_\eta$ by

$$\overline{F}_\eta \leq_{(0,x,0)} \frac{4\left(\|A\eta\|_4+1\right)^2}{1-\max\left(8+8\|A\eta\|_\infty,1\right)t}.$$

□

Now, we prove the geodesic has large radius of convergence.

Lemma 14.5.20. *Under the normalization $A^TA=I$, $S_x=I$ and Euclidean coordinate, any geodesic starting at x satisfies the bound*

$$\|\gamma^{(k)}(0)\|_2 \leq k!c^k$$

for all $k\geq 2$ where $c=512\|A\gamma'(0)\|_4$.

Proof. Recall that under Euclidean coordinate, the geodesic equation is given by

$$\gamma''=\left(A_\gamma^TA_\gamma\right)^{-1}A_\gamma^Ts_{\gamma'}^2 \stackrel{\text{def}}{=} F(\gamma',\gamma).$$

So, we have that $\overline{F}_\eta(y,x,t)=\alpha^2F(\alpha^{-1}y(t)+\gamma'(0),x(t)+\alpha t\gamma'(0))$ with $\eta=\alpha\gamma'(0)$.

Now, we estimate $\overline{F}_\eta$. Lemma 14.5.19 shows that

$$\overline{F}_\eta \leq_{(0,x,0)} \frac{4\left(\|A\eta\|_4+1\right)^2}{1-\max\left(8+8\|A\eta\|_\infty,1\right)t}=\frac{4\left(\alpha\|A\gamma'(0)\|_4+1\right)^2}{1-\max\left(8+8\alpha\|A\gamma'(0)\|_\infty,1\right)t}.$$

Setting $\alpha=\|A\gamma'(0)\|_4^{-1}$ and using $\|A\gamma'(0)\|_\infty\leq\|A\gamma'(0)\|_4$, we have that

$$\overline{F}_\eta \leq_{(0,x,0)} \frac{16}{1-16t}.$$

Lemma 14.7.5 shows that

$$\|\gamma^{(k)}(0)\|_2 \leq \frac{\psi^{(k)}(0)}{\alpha^k}$$

for all $k\geq 2$ where $\psi(t)$ is the solution of

$$\psi'(t)=\frac{16}{1-16\psi(t)} \text{ with } \psi(0)=0.$$

Solving it, we get that

$$\psi(t)=\frac{1}{16}(1-\sqrt{1-512t}).$$

By Theorem 14.2.11, we have that

$$\left|\psi^{(k)}(0)\right| \leq k!(512)^k.$$

Hence, we have that

$$\|\gamma^{(k)}(0)\|_2 \leq \frac{k!(512)^k}{\alpha^k}=k!(512)^k\|A\gamma'(0)\|_4^k$$

for all $k\geq 2$. □

14.5.9.3 Parallel Transport

Motivated from the equation for parallel transport under Euclidean coordinate 14.20, we define the following auxiliary function

$$F(t)=(A_{\gamma(t)}^TA_{\gamma(t)})^{-1}A_{\gamma(t)}^TS_{\gamma'(t)}A_{\gamma(t)}. \tag{14.25}$$

The derivative bounds on parallel transport rely on the smoothness of this auxiliary function.

Lemma 14.5.21. *Given a geodesic $\gamma(t)$. Under the normalization that $A^TA = I$, $S_{\gamma(0)} = I$, we have that*

$$F(t) \leq_0 \frac{c}{2(1-2ct)^2 - (1-ct)^2} \frac{1-ct}{1-2ct}$$

where F is defined in (14.25) and $c = 512\|A\gamma'(0)\|_4$.

Proof. Lemma 14.5.20 shows that $\|\gamma^{(k)}(0)\|_2 \leq k!c^k$ for all $k \geq 2$. Therefore, we have that

$$S_{\gamma(t)} \leq_0 1 + t\|A\gamma'(0)\|_\infty + \sum_{k\geq 2}(ct)^k \leq_0 \frac{1}{1-ct}.$$

Using Lemma 14.7.3, we have that

$$A_{\gamma(t)} \leq_0 \frac{1}{2 - \frac{1}{1-ct}} = \frac{1-ct}{1-2ct}$$

and hence

$$(A_{\gamma(t)}^T A_{\gamma(t)})^{-1} A_{\gamma(t)}^T \leq_0 \frac{(1-2ct)(1-ct)}{2(1-2ct)^2 - (1-ct)^2}.$$

Now, we note that

$$\mathrm{Diag}(A\gamma'(t)) \leq_0 \|A\gamma'(0)\|_\infty + \sum_{k\geq 1} \frac{(k+1)!c^{k+1}t^k}{k!} \leq_0 \frac{c}{(1-ct)^2} \tag{14.26}$$

and hence

$$S_{\gamma'} = S_\gamma^{-1}\mathrm{Diag}(A\gamma'(t)) \leq_0 \frac{1-ct}{1-2ct}\frac{c}{(1-ct)^2} = \frac{c}{(1-2ct)(1-ct)}.$$

This gives the result. □

Now, we prove parallel transport has large radius of convergence.

Lemma 14.5.22. *Given a geodesic with $\gamma(0) = x$. Let $v(t)$ be the parallel transport of a unit vector along $\gamma(t)$. Under the normalization that $A^TA = I$, $S_{\gamma(0)} = I$, we have that*

$$\|v^{(k)}(0)\|_2 \leq k!\,(16c)^k$$

for all $k \geq 1$ where $c = 512\|A\gamma'(0)\|_4$.

Proof. From 14.20, we have that

$$\frac{d}{dt}v(t) = (A_{\gamma(t)}^T A_{\gamma(t)})^{-1} A_{\gamma(t)}^T S_{\gamma'(t)} A_{\gamma(t)} v(t).$$

Let $u(t) = v(\alpha t)$, then we have that

$$\begin{aligned} u'(t) &= \alpha v'(\alpha t) \\ &= \alpha(A_{\gamma(\alpha t)}^T A_{\gamma(\alpha t)})^{-1} A_{\gamma(\alpha t)}^T S_{\gamma'(\alpha t)} A_{\gamma(\alpha t)} u(t). \end{aligned}$$

Let $F(x,t) = \alpha(A_{\gamma(\alpha t)}^T A_{\gamma(\alpha t)})^{-1} A_{\gamma(\alpha t)}^T S_{\gamma'(\alpha t)} A_{\gamma(\alpha t)} x$. Then, Lemma 14.5.21 shows that

$$F(x,t) \leq_{(X(0),0)} \frac{\alpha c}{2(1-2\alpha ct)^2 - (1-\alpha ct)^2} \frac{1-\alpha ct}{1-2\alpha ct}(1+t).$$

Setting $\alpha = \frac{1}{8c}$, we have that

$$F(x,t) \leq_{(X(0),0)} \frac{\frac{1}{8}}{2(1-\frac{t}{4})^2 - \left(1-\frac{t}{8}\right)^2} \frac{1-\frac{t}{8}}{1-\frac{t}{4}}(1+t) \leq_0 \frac{1}{1-t}.$$

Lemma 14.7.4 shows that

$$\|u^{(k)}(0)\|_2 \leq \psi^{(k)}(0)$$

for all $k \geq 1$ where $\psi(t)$ is the solution of

$$\psi'(t) = \frac{1}{1-\psi(t)} \text{ with } \psi(0) = 0.$$

Solving it, we get that

$$\psi(t) = 1 - \sqrt{1-2t}.$$

By Theorem 14.2.11, we have that for any $0 \leq t \leq \frac{1}{2}$, we have that

$$\|u^{(k)}(0)\|_2 \leq \left|\psi^{(k)}(0)\right| \leq k! 2^k$$

for all $k \geq 1$. For $k \geq 1$, we have that

$$\|v^{(k)}(0)\|_2 \leq k! \left(16c\right)^k.$$

□

14.5.9.4 Jacobi field

Motivated from the equation for Jacobi field under orthogonal frame basis (Lemma 14.5.5), we define the following auxiliary function

$$F(t) = X^{-1}(A_\gamma^T A_\gamma)^{-1}\left(A_\gamma^T S_{\gamma'} P_\gamma S_{\gamma'} A_\gamma - A_\gamma^T diag(P_\gamma s_{\gamma'}^2) A_\gamma\right) X. \tag{14.27}$$

The derivative bounds on Jacobi field rely on the smoothness of this auxiliary function.

Lemma 14.5.23. *Given a geodesic $\gamma(t)$ and an orthogonal frame $\{X_i\}_{i=1}^n$. Under the normalization that $A^T A = I$, $S_{\gamma(0)} = I$, we have that*

$$F(t) \leq_0 \frac{12c^2}{1-64ct}$$

where F is defined in (14.27) and $c = 512\|A\gamma'(0)\|_4$.

Proof. We first bound the derivatives of $s_{\gamma'}^2$. Using $\|\gamma^{(k)}(0)\|_2 \leq k! c^k$ for all $k \geq 2$ (Lemma 14.5.20), we have that

$$\gamma'(t) - \gamma'(0) \leq_0 \frac{d}{dt}\frac{1}{1-ct} = \frac{c}{(1-ct)^2}.$$

Using $A^T A = I$ and $\mathrm{Diag}(A\gamma'(t)) \leq_0 \frac{c}{(1-ct)^2}$ (14.26), we have that

$$\mathrm{Diag}(A\gamma'(t))A(\gamma'(t) - \gamma'(0)) \leq_0 \left(\frac{c}{(1-ct)^2}\right)^2. \tag{14.28}$$

Next, we note that

$$\mathrm{Diag}(A\gamma'(t))A\gamma'(0) = \mathrm{Diag}(A\gamma'(0))A(\gamma'(t) - \gamma'(0)) + \mathrm{Diag}(A\gamma'(0))^2.$$

and hence

$$\mathrm{Diag}(A\gamma'(t))A\gamma'(0) \leq_0 \frac{c^2}{(1-ct)^2} + c^2. \tag{14.29}$$

Combining (14.28) and (14.28), we get

$$\mathrm{Diag}(A\gamma'(t))A(\gamma'(t)) \leq_0 \left(\frac{c}{(1-ct)^2}\right)^2 + \frac{c^2}{(1-ct)^2} + c^2 \leq_0 \frac{3c^2}{(1-ct)^4}$$

In the proof of Lemma 14.5.21, we showed that $S_\gamma \leq_0 \frac{1}{1-ct}$ and hence

$$\begin{aligned} s^2_{\gamma'} &= S_\gamma^{-2}\mathrm{Diag}(A\gamma'(t))A(\gamma'(t)). \\ &\leq_0 \left(\frac{1}{2-\frac{1}{1-ct}}\right)^2 \frac{3c^2}{(1-ct)^4} \\ &= \frac{3c^2}{(1-ct)^2(2(1-ct)-1)^2}. \end{aligned}$$

In the proof of Lemma 14.5.21, we showed that $A_\gamma \leq_0 \frac{1-ct}{1-2ct}$ and $(A_\gamma^T A_\gamma)^{-1} \leq_0 \frac{(1-2ct)^2}{2(1-2ct)^2-(1-ct)^2}$. Therefore, we have that

$$P_\gamma = A_\gamma(A_\gamma^T A_\gamma)^{-1}A_\gamma^T \leq_0 \frac{(1-ct)^2}{2(1-2ct)^2-(1-ct)^2}.$$

Hence, we have

$$\begin{aligned} P_\gamma s^2_{\gamma'} &\leq_0 \frac{(1-ct)^2}{2(1-2ct)^2-(1-ct)^2}\frac{3c^2}{(1-ct)^2(1-2ct)^2} \\ &= \frac{3c^2}{(1-2ct)^2(2(1-2ct)^2-(1-ct)^2)}. \end{aligned}$$

Let $Y = (A_\gamma^T A_\gamma)^{-1}\left(A_\gamma^T S_{\gamma'} P_\gamma S_{\gamma'} A_\gamma - A_\gamma^T diag(P_\gamma s^2_{\gamma'})A_\gamma\right)$. By a similar proof, we have that

$$\begin{aligned} Y_0 \leq_0 &\frac{(1-2ct)^2}{2(1-2ct)^2-(1-ct)^2}\left(\frac{1-ct}{1-2ct}\right)^2 \\ &\left(\left(\frac{c}{(1-2ct)(1-ct)}\right)^2\frac{(1-ct)^2}{2(1-2ct)^2-(1-ct)^2} + \frac{3c^2}{(1-2ct)^2(2(1-2ct)^2-(1-ct)^2)}\right) \\ = &\frac{4c^2}{(2(1-2ct)^2-(1-ct)^2)^2}\left(\frac{1-ct}{1-2ct}\right)^2 \end{aligned}$$

Next, we let $Z(t) = X(t)v$ for some unit vector v. Since $X(t)$ is a parallel transport of $X(0)$, $Z(t)$ is a parallel transport of $X(0)v$ and hence Lemma 14.5.22 shows that $\|Z^{(k)}(t)\|_2 \leq (16c)^k$ for all i. For any k, there is unit vector v_k such that $\|\frac{d}{dt^k}X(0)\|_2 = \|\frac{d}{dt^k}X(0)v_k\|_2$. Hence, we have that

$$\|\frac{d}{dt^k}X(0)\|_2 \leq (16c)^k.$$

Therefore, we have that $X \leq_0 \frac{1}{1-16ct}$. Since $X(0) = I$, we have that $X^{-1} \leq_0 \frac{1}{2-\frac{1}{1-16ct}} = \frac{1-16ct}{1-32ct}$. Thus,

we have that

$$\begin{aligned} F(t) &\leq_0 \frac{4c^2}{(2(1-2ct)^2-(1-ct)^2)^2}\left(\frac{1-ct}{1-2ct}\right)^2\frac{1}{1-32ct} \\ &\leq_0 \frac{12c^2}{1-64ct}. \end{aligned}$$

□

Now, we prove Jacobi field has large radius of convergence.

Lemma 14.5.24. *Given a geodesic $\gamma(t)$ and an orthogonal frame $\{X_i\}_{i=1}^n$ along γ. Let $V(t)$ be a Jacobi field along $\gamma(t)$ with $V(0)=0$ and let $U(t)$ be the Jacobi field under the X_i coordinates, namely, $V(t)=X(t)U(t)$. Under the normalization that $A^TA=I$, $S_{\gamma(0)}=I$, we have that*

$$\|U^{(k)}(0)\|_2 \leq k!\,(256c)^k$$

for all $k\geq 2$ where $c=512\|A\gamma'(0)\|_4$.

Proof. Note that $\frac{d^2U(t)}{dt^2}+F(t)U(t)=0$ where $F(t)$ defined in (14.27). Since the equation is linear, we can rescale U and assume that $\|U'(0)\|_2=\frac{1}{\alpha}$.

Let $\overline{F}(U(t),t)=\alpha^2F(\alpha t)\,(U(t)+\alpha tU'(0))$ with $\alpha=\frac{1}{64c}$. Using $\alpha^2F(\alpha t)\leq_0 \frac{12c^2\alpha^2}{1-64c\alpha t}$ (Lemma 14.5.23), $U\leq_{U=0} t$ and $\alpha tU'(0)\leq_{t=0}\|\alpha U'(0)\|_2 t=t$, we have that

$$\begin{aligned} \overline{F} &\leq_{(0,0)} \frac{24c^2\alpha^2 t}{1-64c\alpha t} \\ &\leq_0 \frac{t}{1-t}. \end{aligned}$$

Lemma 14.7.5 shows that that

$$\|U^{(k)}(t)\|_2 \leq \frac{\psi^{(k)}(\alpha^{-1}t)}{\alpha^k}$$

for all $k\geq 2$ where $\psi(t)$ is the solution of

$$\psi'(t)=\frac{2}{1-\psi(t)} \text{ with } \psi(0)=0.$$

Solving it, we get that

$$\psi(t)=1-\sqrt{1-4t}.$$

By Theorem 14.2.11, we have that for any $0\leq t\leq\frac{1}{4}$, we have that

$$\left|\psi^{(k)}(0)\right| \leq k!(4)^k.$$

Hence, we have that

$$\|U^{(k)}(0)\|_2 \leq \frac{k!4^k}{\alpha^k} \leq k!\,(256c)^k$$

for all $k\geq 2$. □

14.5.10 Implementation

14.5.10.1 Computing Geodesic Equation

To apply Theorem 14.6.4, we define

$$F(u,s) = A(A^T S^{-2} A)^{-1} A^T S^{-1} u^2 \tag{14.30}$$

The following lemma bounds the Lipschitz constant of F.

Lemma 14.5.25. *Assuming $\frac{1}{2} \leq s_i \leq 2$ for all i. Then, we have*

$$\|DF(u,s)[d_u,d_s]\|_2^2 \leq 48\|u\|_4^2\|d_u\|_4^2 + 240\|u\|_4^4\|d_s\|_\infty^2.$$

Proof. Let $A_s = S^{-1}A$ and $S_d = S^{-1}\mathrm{Diag}(d_s)$. Then, we have $F(u,s) = A(A_s^T A_s)^{-1} A_s^T u^2$. Hence, we have that

$$\begin{aligned} DF(u,s)[d_u,d_s] &= 2A(A_s^T A_s)^{-1} A_s^T S_d A_s (A_s^T A_s)^{-1} A_s^T u^2 \\ &\quad -A(A_s^T A_s)^{-1} A_s^T S_d u^2 \\ &\quad +2A(A_s^T A_s)^{-1} A_s^T U d_u \end{aligned}$$

Let $P = A_s \left(A_s^T A_s\right)^{-1} A_s^T$, then, we have that

$$\begin{aligned} \|DF(u,s)[d_u,d_s]\|_2^2 &\leq 12\left(u^2\right)^T PS_d PS^2 PS_d P u^2 \\ &\quad +3\left(u^2\right)^T S_d PS^2 PS_d u^2 \\ &\quad +12 d_u^T UPS^2 PU d_u. \end{aligned}$$

Using that $P \preceq I$, we have that

$$\begin{aligned} & \|DF(u,s)[d_u,d_s]\|_2^2 \\ \leq\ & 12\|S\|_\infty^2 \left(u^2\right)^T PS_d^2 P u^2 + 3\|S\|_\infty^2 \left(u^2\right)^T S_d^2 u^2 + 12\|S\|_\infty^2 \sum_i (d_u)_i^2 u_i^2 \\ \leq\ & 15\|S\|_\infty^2 \|S_d\|_\infty^2 \|u\|_4^4 + 12\|S\|_\infty^2 \|u\|_4^2 \|d_u\|_4^2 \\ \leq\ & 15\|S\|_\infty^2 \|S^{-1}\|_\infty^2 \|d_s\|_\infty^2 \|u\|_4^4 + 12\|S\|_\infty^2 \|u\|_4^2 \|d_u\|_4^2. \end{aligned}$$

Now, we use that $\frac{1}{2} \leq s_i \leq 2$ and get

$$\|DF(u,s)[d_u,d_s]\|_2^2 \leq 48\|u\|_4^2\|d_u\|_4^2 + 240\|d_s\|_\infty^2\|u\|_4^4.$$

□

Now, we can apply the collocation method to obtain a good approximation of the solution u^*.

Lemma 14.5.26. *Let γ be a random geodesic generated by the geodesic walk with step size $h \leq \frac{1}{10^7\sqrt{n}}$. In time $O(mn^{\omega-1}\log^2(1/\varepsilon))$, we can find $\overline{\gamma}$ such that $\max_{0\leq t\leq \ell} \|\gamma(t) - \overline{\gamma}(t)\|_\infty \leq \varepsilon$ and $\max_{0\leq t\leq \ell} \|\gamma'(t) - \overline{\gamma}'\|_\infty \leq \varepsilon$ with probability at least $1 - O(\exp(-\sqrt{n}/100))$. Furthermore, $\overline{\gamma}$ is a $O(\log(1/\varepsilon))$ degree polynomial.*

Proof. Let $s(t) = A\gamma(t) - b$. By rotating the space and rescaling the rows of A, we assume that

$s(0)_i = 1$ for all i and $A^T A = I$. We define F as (14.30). Then, we have that

$$\begin{aligned} s''(t) &= F(s', s), \\ s'(0) &= A\gamma'(0), \\ s(0) &= 1. \end{aligned}$$

We let $\alpha = 40\ell$ and

$$\begin{aligned} K &\stackrel{\text{def}}{=} \alpha\|F(A\gamma'(0), 1)\|_4 + \|A\gamma'(0)\|_4 \\ &\leq \alpha\|A(A^TA)^{-1}A^T\left(A\gamma'(0)\right)^2\|_2 + \|A\gamma'(0)\|_4 \\ &\leq \alpha\|A\gamma'(0)\|_4^2 + \|A\gamma'(0)\|_4. \end{aligned}$$

Using $\|A_x\gamma'\|_4 \leq 2n^{-1/4}$ (Lemma 14.5.10), we have that

$$\begin{aligned} K \leq 2\|A\gamma'(0)\|_4 \leq 4n^{-1/4}, \\ \alpha K \leq 160\ell n^{-1/4} \leq \frac{1}{2}. \end{aligned}$$

For any $\|u - s'(0)\|_4 \leq K \leq 4n^{-1/4}$, $\|s - 1\|_4 \leq \alpha K \leq \frac{1}{2}$, Lemma 14.5.25 shows that

$$\begin{aligned} \|DF(u,s)[d_u, d_s]\|_4 &\leq \|DF(u,s)[d_u, d_s]\|_2 \\ &\leq 8\|u\|_4\|d_u\|_4 + 16\|u\|_4^2\|d_s\|_\infty \\ &\leq 48n^{-1/4}\|d_u\|_4 + 576n^{-1/2}\|d_s\|_4. \end{aligned}$$

Therefore, for any $\|u_1 - s'(0)\|_4 \leq K$, $\|s_1 - 1\|_4 \leq \alpha K$, $\|u_2 - s'(0)\|_4 \leq K$, $\|s_2 - 1\|_4 \leq \alpha K$, we have that

$$\begin{aligned} \|F(u_1, s_1) - F(u_2, s_2)\|_4 &\leq 48n^{-1/4}\|u_1 - u_2\|_4 + 576n^{-1/2}\|s_1 - s_2\|_4 \\ &\leq \frac{1}{\alpha}\|u_1 - u_2\|_4 + \frac{1}{\alpha^2}\|s_1 - s_2\|_4. \end{aligned}$$

Since γ is analytic and $\|\gamma^{(k)}(0)\|_2 = O(k!n^{-k/4})$ (Lemma 14.5.22), $\gamma(t)$ is ε close to a polynomial with degree $O(\log(1/\varepsilon))$ for $0 \leq t \leq o(n^{1/4})$. Hence, we can apply Theorem 14.6.6 and find $\overline{\gamma}$ such that $\|\gamma - \overline{\gamma}\|_4 \leq \varepsilon$ and $\|\gamma' - \overline{\gamma}'\|_4 \leq \varepsilon$ in $O(n\log^3(K/\varepsilon))$ time plus $O(\log^2(K/\varepsilon))$ evaluations of F. Note that each evaluation of F involves solving a linear system and hence it takes $O(mn^{\omega-1})$. Therefore, the total running time is $O(mn^{\omega-1}\log^2(1/\varepsilon))$. □

14.5.10.2 Computing Parallel Transport

Lemma 14.5.27. *Given γ be a random geodesic generated by the geodesic walk with step size $h \leq \frac{1}{10^7\sqrt{n}}$ and an unit vector v. Let $v(t)$ be the parallel transport of a unit vector along $\gamma(t)$. In time $O(mn^{\omega-1}\log^2(1/\varepsilon))$, we can find $\overline{v}$ such that $\max_{0\leq t\leq\ell}\|v(t) - \overline{v}(t)\|_\infty \leq \varepsilon$ with probability at least $1 - O(\exp(-\sqrt{n}/100))$. Furthermore, $\overline{v}$ is a $O(\log(1/\varepsilon))$ degree polynomial.*

Similarly, given a basis $\{v_i\}_{i=1}^n$, in time $O(mn^{\omega-1}\log^2(1/\varepsilon))$, we can find an approximate parallel transport $\overline{v_i}(t)$ of $\{v_i\}_{i=1}^n$ along $\gamma(t)$ with probability at least $1 - O(\exp(-\sqrt{n}/100))$.

Proof. Recall that the equation for parallel transport (14.20) is given by

$$\frac{d}{dt}v(t) = \left(A_{\gamma(t)}^T A_{\gamma(t)}\right)^{-1} A_{\gamma(t)}^T S_{\gamma'(t)} A_{\gamma(t)} v.$$

By rotating the space and rescaling the rows of A, we assume that $s(\gamma(0))_i = 1$ for all i and $A^TA = I$.

In the proof of Lemma 14.5.26, we know that $\frac{1}{2} \le s(\gamma(t))_i \le 2$ for all $0 \le t \le \ell$. For any unit vector u, we have

$$\begin{aligned}\| \left(A_\gamma^T A_\gamma\right)^{-1} A_\gamma^T S_{\gamma'} A_\gamma u\|_2 &\le 2\| \left(A_\gamma^T A_\gamma\right)^{-1/2} A_\gamma^T S_{\gamma'} A_\gamma u\|_2 \\ &\le 2\|S_{\gamma'}\|_\infty \|A_\gamma u\|_2 \\ &\le 4\|S_{\gamma'}\|_\infty \le 24\sqrt{h}\end{aligned}$$

where we used Lemma 14.5.10 in the last line. Using $h \le \frac{1}{10^7\sqrt{n}}$, we have that

$$\| \left(A_\gamma^T A_\gamma\right)^{-1} A_\gamma^T S_{\gamma'} A_\gamma\|_2 \le \frac{1}{20\ell}.$$

Since v is analytic and $\|v^{(k)}(0)\|_2 = O(k! n^{-k/4})$ (Lemma 14.5.22), $v(t)$ is ε close to a polynomial with degree $O(\log(1/\varepsilon))$ for $0 \le t \le o(n^{1/4})$. Hence, we can apply Theorem 14.6.4 and find $\overline{v}$ such that $\|v - \overline{v}\|_2 \le \varepsilon$ in $O(n \log^3(1/\varepsilon))$ time plus $O(\log^2(1/\varepsilon))$ evaluations of F. Note that each evaluation of F involves solving a linear system and hence it takes $O(mn^{\omega-1})$. Therefore, the total running time is $O(mn^{\omega-1} \log^2(1/\varepsilon))$.

For the last result, we note that each evaluation of F becomes computing matrix inverse and performing matrix multiplication and they can be done in again $O(mn^{\omega-1})$ time. □

14.5.10.3 Computing Jacobi field

Lemma 14.5.28. *Given γ be a random geodesic generated by the geodesic walk with step size $h \le \frac{1}{10^7\sqrt{n}}$. Let $\{X_i\}_{i=1}^n$ be an orthogonal frame along γ. Let $v(t)$ be a Jacobi field along $\gamma(t)$ with $v(0) = 0$ and let $u(t)$ be the Jacobi field under the X_i coordinates, namely, $v(t) = X(t)u(t)$. In time $O(mn^{\omega-1} \log^2(1/\varepsilon))$, we can find $\overline{u}$ such that $\max_{0\le t\le \ell} \|u(t) - \overline{u}(t)\|_\infty \le \varepsilon$ with probability at least $1 - O(\exp(-\sqrt{n}/100))$. Furthermore, $\overline{u}$ is a $O(\log(1/\varepsilon))$ degree polynomial.*

Proof. Recall that the equation for Jacobi field 14.5.5 is given by

$$\frac{d^2u}{dt^2} + X^{-1}(A_\gamma^T A_\gamma)^{-1} \left(A_\gamma^T S_{\gamma'} P_\gamma S_{\gamma'} A_\gamma - A_\gamma^T \mathrm{Diag}(P_\gamma s_{\gamma'}^2) A_\gamma\right) Xu = 0.$$

By rotating the space and rescaling the rows of A, we assume that $s(\gamma(0))_i = 1$ for all i and $A^T A = I$. In the proof of Lemma 14.5.26, we know that $\frac{1}{2} \le s(\gamma(t))_i \le 2$ for all $0 \le t \le \ell$. Since X is an orthogonal frame, we have $XX^T = XX^T = I$. Hence, for any unit vector v, we have

$$\begin{aligned}&\|X^{-1}(A_\gamma^T A_\gamma)^{-1} \left(A_\gamma^T S_{\gamma'} P_\gamma S_{\gamma'} A_\gamma - A_\gamma^T \mathrm{Diag}(P_\gamma s_{\gamma'}^2) A_\gamma\right) Xv\|_2 \\ \le& 2\|(A_\gamma^T A_\gamma)^{-1/2} \left(A_\gamma^T S_{\gamma'} P_\gamma S_{\gamma'} A_\gamma - A_\gamma^T \mathrm{Diag}(P_\gamma s_{\gamma'}^2) A_\gamma\right) Xv\|_2 \\ \le& 2\|S_{\gamma'} P_\gamma S_{\gamma'} A_\gamma v\|_2 + 2\|\mathrm{Diag}(P_\gamma s_{\gamma'}^2) A_\gamma X v\|_2.\end{aligned}$$

Using $\|S_{\gamma'}\|_2 \le 6\sqrt{h}$ and $\|\mathrm{Diag}(P_\gamma s_{\gamma'}^2)\|_2 \le \|P_\gamma s_{\gamma'}^2\|_2 \le \|s_{\gamma'}\|_4^2 \le 4n^{-1/2}$ (Lemma 14.5.10), we have that

$$\begin{aligned}&\|X^{-1}(A_\gamma^T A_\gamma)^{-1} \left(A_\gamma^T S_{\gamma'} P_\gamma S_{\gamma'} A_\gamma - A_\gamma^T \mathrm{Diag}(P_\gamma s_{\gamma'}^2) A_\gamma\right) Xv\|_2 \\ \le& 12\sqrt{h}\|P_\gamma S_{\gamma'} A_\gamma v\|_2 + 8n^{-1/2}\|A_\gamma X v\|_2 \\ \le& 72h\|A_\gamma v\|_2 + 16n^{-1/2}\|Xv\|_2 \\ \le& 144h + 16n^{-1/2} \le \frac{1}{(40\ell)^2}\end{aligned}$$

where we used $h \leq \frac{1}{10^7\sqrt{n}}$ in the last line.

Since u is analytic and $\|u^{(k)}(0)\|_2 = O(k!n^{-k/4})$ (Lemma 14.5.24), $u(t)$ is ε close to a polynomial with degree $O(\log(1/\varepsilon))$ for $0 \leq t \leq o(n^{1/4})$. Hence, we can apply Theorem 14.6.4 and find $\overline{u}$ such that $\|u - \overline{u}\|_2 \leq \varepsilon$ in $O(n\log^3(1/\varepsilon))$ time plus $O(\log^2(1/\varepsilon))$ evaluations of F. Note that each evaluation of F involves computing matrix inversions and matrix multiplications and hence it takes $O(mn^{\omega-1})$. Therefore, the total running time is $O(mn^{\omega-1}\log^2(1/\varepsilon))$. □

14.5.10.4 Computing Geodesic Walk

Proof of Theorem 14.5.2. To implement the geodesic walk, we use Lemma 14.5.26 to compute the geodesic, Lemma 14.5.27 to compute an orthogonal frame along the geodesic and Lemma 14.5.28 to compute the Jacobi field along. Using the Jacobi field, we can use (14.9) and Lemma 14.4.12 to compute the probability from x to y and the probability from y to x. Using these probabilities, we can implement the rejection sampling. It suffices to compute the geodesic and the probability up to $1/n^{O(1)}$ accuracy and hence these operations can be done in time $O(mn^{\omega-1}\log^2(n))$.

Note that we only use randomness to prove that $\|A_{\gamma(0)}u\|_2 \leq \frac{3}{2}$ and $\|A_\gamma u\|_4 \leq \frac{1.55}{n^{1/4}}$ (Lemma 14.5.10) and they can be checked. When we condition our walk to that, we only change the distribution by exponential small amount. Hence, this result is stated without mentioning the success probability. □

14.6 Collocation Method for ODE

In this section, we study a collocation method for solving ordinary differential equation (ODE) and show how to solve a nice enough ODE in nearly constant number of iterations without explicitly computing higher derivatives.

14.6.1 First Order ODE

We first consider the following first order ODE

$$\begin{aligned} \frac{d}{dt}u(t) &= F(u(t),t), \text{ for } 0 \leq t \leq \ell \\ u(0) &= v \end{aligned} \tag{14.31}$$

where $F : \mathbb{R}^{n+1} \to \mathbb{R}^n$ and $u(t) \in \mathbb{R}^n$. The idea of collocation methods is to find a degree d polynomial p such that

$$\begin{aligned} \frac{d}{dt}p(t) &= F(p(t),t), \text{ for } t = c_1, c_2, \cdots, c_d \\ p(0) &= v \end{aligned} \tag{14.32}$$

where $c_1, c_2, \cdots, c_d$ are carefully chosen distinct points on $[0, \ell]$. Here, we call $p : \mathbb{R} \to \mathbb{R}^n$ is a degree d polynomial if $p(t) = [p_1(t); p_2(t); \cdots; p_n(t)]$ and each $p_i(t)$ is an univariate polynomial with degree at most d. The first part of the proof shows the existence of a solution for the systems (14.32). To describe the algorithm, it is easier to consider an equivalent integral equation.

Lemma 14.6.1. *Given distinct points $c_1, c_2, \cdots, c_d \in \mathbb{R}$ and $F : \mathbb{R}^{n+1} \to \mathbb{R}^n$, consider the nonlinear*

map $T : \mathbb{R}^{d\times n} \to \mathbb{R}^{d\times n}$ *defined by*

$$T(\zeta)_{(i,k)} = \int_0^{c_i} \sum_{j=1}^{d} F(\zeta_j, c_j)_k \phi_j(s) ds \text{ for } i \in [d], k \in [n]$$

where $\phi_i(s) = \prod_{j\neq i} \frac{s-c_j}{c_i-c_j}$ *are the Lagrange basis polynomials. Given any* $\zeta \in \mathbb{R}^{d\times n}$ *such that*

$$\zeta_i = v + T(\zeta)_i, \text{ for } i \in [d] \tag{14.33}$$

the polynomial

$$p(t) = v + \int_0^t \sum_{j=1}^{d} F(\zeta_j, c_j)\phi_j(s) ds$$

is a solution of the system (14.32).

Proof. Define the polynomials $\phi_i(s) = \prod_{j\neq i} \frac{s-c_j}{c_i-c_j}$. Note that $\phi_i(c_j) = \delta_{ij}$. Therefore, we have

$$\sum_{j=1}^{d} \alpha_j \phi_j(c_i) = \alpha_i.$$

Therefore, $p(0) = v$ and

$$\frac{d}{dt}p(c_i) = \sum_{j=1}^{d} F(\zeta_j, c_j)\phi_j(c_i) = F(\zeta_j, c_j).$$

Since $\zeta_i = v + T(\zeta)_i$, we have that

$$\zeta_i = v + \int_0^{c_i} \sum_{j=1}^{d} F(\zeta_j, c_j)\phi_j(s) ds = p(c_i).$$

Hence, we have $\frac{d}{dt}p(c_i) = F(p(c_i), c_i)$. Therefore, p is a solution to the system (14.32). □

From Lemma 14.6.1, we see that it suffices to solve the system (14.33). We solve it by a simple fix point iteration shown in Algorithm 50.

Algorithm 50: `CollocationMethod`

Input: An ordinary differential equation $\frac{d}{dt}u(t) = F(u(t), t)$ for $0 \leq t \leq \ell$ with initial condition $u(0) = v$.
Define $T(\zeta)_{(i,k)} = \int_0^{c_i} \sum_{j=1}^{d} F(\zeta_j, c_j)_k \phi_j(s) ds$ for $i \in [d], k \in [n]$ as defined in Lemma 14.6.1.
$c_i = \frac{\ell}{2} + \frac{\ell}{2}\cos(\frac{2i-1}{2d}\pi)$ for all $i \in [d]$, $K = 40\ell \max_{t\in[0,\ell]} \|F(v,t)\|_p$, $Z = \log_2(K/\varepsilon)$
$\zeta_i^{(0)} = v + T(\overline{v})_i$ for all $i \in [d]$ where $\overline{v} = (v, v, \cdots, v) \in \mathbb{R}^{d\times n}$.
for $z = 0, \cdots, Z$ **do**
  $\zeta_i^{(z+1)} = v + T(\zeta^{(z)})_i$ for $i \in [d]$.
end
Output: $p(t) = v + \int_0^{\ell} \sum_{j=1}^{d} F(\zeta_j^{(Z+1)}, c_j)\phi_j(s) ds$.

In the following lemma, we show that T is a contraction mapping if F is smooth enough and hence this algorithm converges linearly. We will use the following norm: $\|x\|_{\infty;p} = \max_{i\in[d]} \left(\sum_{k\in[n]} |x_{(i,k)}|^p\right)^{1/p}$.

Lemma 14.6.2. *Given $x, y \in \mathbb{R}^{d\times n}$ with $d \geq 2$, suppose that $\|F(x_i,t) - F(y_i,t)\|_p \leq L\|x_i - y_i\|_p$ for all $0 \leq t \leq \ell$ and all $i \in [d]$. For $c_k = \frac{\ell}{2} + \frac{\ell}{2}\cos(\frac{2k-1}{2d}\pi)$ and T defined Lemma 14.6.1, we have that*

$$\|T(x) - T(y)\|_{\infty;p} \leq 10\ell L\|x - y\|_{\infty;p}.$$

Also, we have that

$$\|T(x)\|_{\infty;p} \leq 10\ell \max_i \|F(x_i, c_i)\|_{\infty;p}$$

Proof. Using the definition of T and $\|\cdot\|_{\infty;p}$, we have

$$\begin{aligned}
\|T(x) - T(y)\|_{\infty;p} &= \max_i \| \int_0^{c_i} \sum_{j=1}^d F(x_j, c_j)\phi_j(s)ds - \int_0^{c_i} \sum_{j=1}^d F(y_j, c_j)\phi_j(s)ds\|_p \\
&\leq \left(\max_i \sum_{j=1}^d \left| \int_0^{c_i} \phi_j(s)ds \right| \right) \left(\max_i \|F(x_i, c_i) - F(y_i, c_i)\|_p \right) \\
&\leq L\|x - y\|_{\infty;p} \left(\max_i \sum_{j=1}^d \left| \int_0^{c_i} \phi_j(s)ds \right| \right).
\end{aligned}$$

To bound $\max_i \sum_{j=1}^d \left|\int_0^{c_i} \phi_j(s)ds\right|$, it is easier to work with the function $\psi_j(x) \stackrel{\text{def}}{=} \phi_j(\frac{2}{\ell}x - 1)$. Note that ψ_k is the Lagrange basis polynomials on the nodes $\{\cos(\frac{2k-1}{2d}\pi)\}_{k=1}^d$ and hence we have

$$\psi_k(x) = \frac{(-1)^{k-1}\sqrt{1 - x_k^2}\cos\left(d\cos^{-1}x\right)}{d(x - x_k)}$$

where $x_k = \cos(\frac{2k-1}{2d}\pi)$. Lemma 14.6.5 at the end of this section shows that $\left|\int_{-1}^t \psi_k(x)dx\right| \leq \frac{20}{d}$ for all t. Therefore, we have that $\max_i \sum_{j=1}^d \left|\int_0^{c_i} \phi_j(s)ds\right| \leq 10\ell$. This gives the first inequality. The second inequality is similar. □

In each iteration of the collation method, we need to compute $\int_0^{c_i} \sum \alpha_j\phi_j(s)ds$ for some α_j. The following theorem shows that this can be done in $O(d\log(d/\varepsilon))$ time using the multipole method.

Theorem 14.6.3 ([77, Sec 5])**.** *Let $\phi_i(s)$ be the Lagrange basis polynomials on the Chebyshev nodes on $[0,\ell]$, namely, $\phi_i(s) = \prod_{j\neq i} \frac{s-c_j}{c_i-c_j}$ with $c_i = \frac{\ell}{2} + \frac{\ell}{2}\cos(\frac{2i-1}{2d}\pi)$. Given a polynomial $p(s) = \sum_j \alpha_j\phi_j(s)$ and a point set $\{x_1, x_2, \cdots, x_d\}$, one can compute t_i such that*

$$\left| t_i - \int_0^{x_i} p(s)ds \right| \leq \varepsilon\ell\sqrt{\sum_{i\neq j}(\alpha_i - \alpha_j)^2} \text{ for } i \in [d]$$

in time $O(d\log(d/\varepsilon))$.

Now we have everything to state our main result in this subsection.

Theorem 14.6.4. *Let $u(t) \in \mathbb{R}^n$ be the solution of the ODE (14.31). Suppose we are given $\varepsilon, \ell > 0$, $d \geq 2$ and $1 \leq p \leq \infty$ such that*

1. *There is a degree d polynomial q from $\mathbb{R}$ to $\mathbb{R}^n$ such that $q(0) = v$ and $\|\frac{d}{dt}u(t) - \frac{d}{dt}q(t)\|_p \leq \frac{\varepsilon}{\ell}$ for all $0 \leq t \leq \ell$.*
2. *We have that $\|F(x,t) - F(y,t)\|_p \leq \frac{1}{20\ell}\|x - y\|_p$ for all $\|x - v\|_p \leq K$ and $\|y - v\|_p \leq K$ with $K = 40\ell \max_{t\in[0,\ell]} \|F(v,t)\|_p$.*

Then, for any t such that $0 \le t \le \ell$, Algorithm `CollocationMethod` *outputs a degree d polynomial $p(t)$ such that $\|u(t) - p(t)\|_p \le 43\varepsilon$ in $O(nd\log^2(dK/\varepsilon))$ time and $O(d\log(K/\varepsilon))$ evaluations of F.*

Proof. First, we estimate the initial error. Let $\zeta^{(\infty)}$ be the solution to (14.33), $\zeta^{(0)} = v + T(\overline{v})_i$ be the initial vector and $\overline{v} = (v, v, \cdots, v) \in \mathbb{R}^{d\times n}$. Then, we have that

$$\begin{aligned}\|\zeta^{(\infty)} - \zeta^{(0)}\|_{\infty;p} &= \max_i \|(v + T(\zeta^{(\infty)})_i) - (v + T(\overline{v})_i)\|_p \\ &= \|T(\zeta^{(\infty)}) - T(\overline{v})\|_{\infty;p} \\ &\le \frac{1}{2}\|\zeta^{(\infty)} - \overline{v}\|_{\infty;p} \\ &\le \frac{1}{2}\left(\|\zeta^{(\infty)} - \zeta^{(0)}\|_{\infty;p} + \|\zeta^{(0)} - \overline{v}\|_{\infty;p}\right).\end{aligned}$$

Therefore, we have that

$$\begin{aligned}\|\zeta^{(\infty)} - \zeta^{(0)}\|_{\infty;p} &\le \|\zeta^{(0)} - \overline{v}\|_{\infty;p} \\ &= \|T((v, v, \cdots, v))\|_{\infty;p} \\ &\le 10\ell \max_i \|F(v, c_i)\|_p \\ &\le \frac{K}{4}.\end{aligned}$$

Hence, we have that $\|\zeta^{(\infty)} - \overline{v}\|_{\infty;p} \le \frac{K}{2}$.

Using the assumption on F, Lemma 14.6.2 shows that $\|T(x) - T(y)\|_{\infty;p} \le \frac{1}{2}\|x - y\|_{\infty;p}$ and hence

$$\|\zeta^{(\infty)} - \zeta^{(k)}\|_{\infty;p} \le \frac{K}{2^{k+1}}.$$

Thus, it takes $\log_2(K/\varepsilon)$ iteration to get a point $\zeta^{(k)}$ with

$$\|\zeta^{(\infty)} - \zeta^{(k)}\|_{\infty;p} \le \varepsilon. \tag{14.34}$$

Also, this shows that $\|\zeta^{(\infty)} - \overline{v}\|_{\infty;p} \le K$. Hence, we only requires the assumption (2) to be satisfied in this region.

Now, we show that $\zeta^{(k)}$ is close to the solution by using the existence of q. By the assumption on q, we have that $\|\frac{d}{dt}u(t) - \frac{d}{dt}q(t)\|_p \le \frac{\varepsilon}{\ell}$ and hence $\|u(t) - q(t)\|_p \le \varepsilon$. Using the smoothness of F, we have that $\|F(u(t), t) - F(q(t), t)\|_p \le \frac{\varepsilon}{\ell}$ for all $0 \le t \le \ell$. Therefore, we have that

$$\begin{aligned}\|\frac{d}{dt}q(t) - F(q(t), t)\|_p &\le \|\frac{d}{dt}q(t) - \frac{d}{dt}u(t)\|_p + \|F(q(t), t) - F(u(t), t)\|_p \\ &\le 2\frac{\varepsilon}{\ell}\end{aligned}$$

for all $0 \le t \le 1$. Therefore, we have

$$\begin{aligned}q(t) &= v + \int_0^t \sum_{j=1}^d \frac{d}{dt}q(c_j)\phi_j(s)ds \\ &= v + \int_0^t \sum_{j=1}^d \left(F(q(c_j), c_j) + \delta_j\right)\phi_j(s)ds\end{aligned}$$

where $\|\delta_i\|_p \le 2\frac{\varepsilon}{\ell}$ for all $i \in [d]$. By Lemma 14.6.2, we have that

$$\|q(t) - v - \int_0^t \sum_{j=1}^d F(q(c_j), c_j)\phi_j(s)ds\|_p \le 20\varepsilon.$$

Now, we compare $\overline{q}_j \stackrel{\text{def}}{=} q(c_j)$ with the approximate solution constructed by the fixpoint algorithm

$$q^{(k)}(t) = v + \int_0^t \sum_{j=1}^d F(\zeta_j^{(k)}, c_j)\phi_j(s)ds.$$

For $0 \le t \le \ell$, we have

$$\begin{aligned} \|q(t) - q^{(k)}(t)\|_p &\le \|T(\overline{q}) - T(\zeta^{(k)})\|_{\infty;p} + 20\varepsilon \\ &\le \frac{1}{2}\|\overline{q} - \zeta^{(k)}\|_{\infty;p} + 20\varepsilon. \end{aligned} \tag{14.35}$$

Setting $t = c_i$, we have

$$\begin{aligned} \|\overline{q} - \zeta^{(k+1)}\|_{\infty;p} &= \max_i \|\overline{q}(c_i) - q^{(k)}(c_i)\|_p \\ &\le \frac{1}{2}\|q - \zeta^{(k)}\|_{\infty;p} + 20\varepsilon. \end{aligned}$$

Since $\|\zeta^{(k+1)} - \zeta^{(k)}\|_{\infty;p} \le \|\zeta^{(k+1)} - \zeta^{(\infty)}\|_{\infty;p} + \|\zeta^{(k)} - \zeta^{(\infty)}\|_{\infty;p} \le 2\varepsilon$, we have

$$\|\overline{q} - \zeta^{(k)}\|_{\infty;p} \le \frac{1}{2}\|\overline{q} - \zeta^{(k)}\|_{\infty;p} + 2\varepsilon + 20\varepsilon$$

Therefore, we have

$$\|\overline{q} - \zeta^{(k)}\|_{\infty;p} \le 44\varepsilon.$$

Putting it into (14.35), we have

$$\|q(t) - q^{(k)}(t)\|_p \le 42\varepsilon$$

for all $0 \le t \le \ell$. Using that $\|u(t) - q(t)\|_p \le \varepsilon$, we have

$$\|u(t) - q^{(k)}(t)\|_p \le 43\varepsilon.$$

Hence, it proves the guarantee.

Each iteration involves computing $v_k + \int_0^{c_i} \sum_{j=1}^d F(\zeta_j^{(z)}, c_j)_k \phi_j(s)ds$ for all $i \in [d], k \in [n]$. Note that $\sum_{j=1}^d F(\zeta_j^{(z)}, c_j)_k \phi_j(s)$ is a polynomial expressed by Lagrange polynomials. Theorem 14.6.3 shows they can be computed in $O(d\log(dK/\varepsilon)))$ with $\frac{\varepsilon}{Kd^{O(1)}}$ accuracy. Since there are n coordinates, it takes $O(nd\log(dK/\varepsilon)))$ time plus d evaluation per iteration. □

The theorem above essentially says that if the solution is well approximated by a polynomial and if the F has small enough Lipschitz constant, then we can reconstruct the solution efficiently. Note that this method is not useful for stochastic differential equation because Taylor expansion of the solution involves the high moments of probability distributions which is very expensive to store.

We conclude this subsection with a lemma used in the proof of Lemma 14.6.2.

Lemma 14.6.5. *Given $d \ge 2$. Let $\psi(x) = \frac{\sqrt{1-y^2}\cos(d\cos^{-1}x)}{d(x-y)}$ where $y = \cos(\frac{2k-1}{2d}\pi)$ for some integer $k \in [d]$. Then, we have that*

$$\left|\int_{-1}^t \psi(x)dx\right| \le \frac{20}{d}.$$

for any $-1 \le t \le 1$.

Proof. Let $z_j = \cos(j\pi/d)$. For $j < k$, we have that

$$\begin{aligned}
\left|\int_{z_{j-1}}^{z_j} \psi(x)dx\right| &\le \frac{\sqrt{1-y^2}}{d}\left|\int_{z_{j-1}}^{z_j} \frac{\cos\left(d\cos^{-1}x\right)}{z_j - y}dx\right| + \frac{\sqrt{1-y^2}}{d}\int_{z_{j-1}}^{z_j}\left|\frac{\cos\left(d\cos^{-1}x\right)(z_j-x)}{(z_j-y)(x-y)}\right|dx \\
&\le \frac{\sqrt{1-y^2}}{d}\left|\int_{z_{j-1}}^{z_j} \frac{\cos\left(d\cos^{-1}x\right)}{z_j - y}dx\right| + \frac{\sqrt{1-y^2}}{d}\frac{(z_j - z_{j-1})^2}{(z_j-y)^2} \\
&= \frac{\sqrt{1-y^2}}{d}\frac{z_j - z_{j-1}}{d^2-1} + \frac{\sqrt{1-y^2}}{d}\frac{(z_j-z_{j-1})^2}{(z_j-y)^2} \\
&\le \frac{\pi}{d^2(d^2-1)} + \frac{4}{d}\frac{1}{(k-\frac{1}{2}-j)^2}.
\end{aligned}$$

Hence, we have that

$$\begin{aligned}
\left|\int_{-1}^{t}\psi(x)dx\right| &\le \sum_{k\neq j}\left|\int_{z_{j-1}}^{z_j}\psi(x)dx\right| + \int_{z_{k-1}}^{z_k}|\psi(x)|\,dx \\
&\le \frac{6}{d} + \int_{z_{k-1}}^{z_k}|\psi(x)|\,dx.
\end{aligned}$$

By simple calculation, one can check that $|\psi(x)| \le 2$ and hence

$$\left|\int_{-1}^{t}\psi(x)dx\right| \le \frac{20}{d}.$$

□

■ 14.6.2 Second Order ODE

Now, we consider the following second order ODE

$$\begin{aligned}
\frac{d^2}{dt^2}u(t) &= F(\frac{d}{dt}u(t), u(t), t), \text{ for } 0 \le t \le \ell \qquad (14.36)\\
\frac{d}{dt}u(0) &= w, \\
u(0) &= v
\end{aligned}$$

where $F : \mathbb{R}^{2n+1} \to \mathbb{R}^n$ and $u(t) \in \mathbb{R}^n$. Using a standard reduction from second order ODE to first order ODE, we show how to apply our first order ODE method to second order ODE.

Theorem 14.6.6. *Let $x(t) \in \mathbb{R}^n$ be the solution of the ODE (14.36). Given some $\varepsilon, \ell > 0$, $d \ge 2$ and $1 \le p \le \infty$, let $\alpha = 40\ell$ and suppose that*

1. *There is a degree d polynomial q from $\mathbb{R}$ to $\mathbb{R}^n$ such that $q(0) = v$, $q'(0) = w$, $\|\frac{d}{dt^2}x(t) - \frac{d}{dt^2}q(t)\|_p \le \frac{\varepsilon}{\ell^2}$ for all $0 \le t \le \ell$.*
2. *We have that $\|F(x,\gamma,t) - F(y,\eta,t)\|_p \le \frac{1}{\alpha}\|x-y\|_p + \frac{1}{\alpha^2}\|\gamma-\eta\|_p$ for all $\|x-w\|_p \le K$, $\|y-w\|_p \le K$, $\|\gamma - v\|_p \le \alpha K$, $\|\eta - v\|_p \le \alpha K$ where $K = \alpha \max_{t\in[0,\ell]}\|F(w,v,t)\|_p + \|w\|_p$.*

Then, for any t such that $0 \le t \le \ell$, in $O(nd\log^2(dK/\varepsilon))$ time plus $O(d\log(K/\varepsilon))$ evaluations of F, we can find $p(t), p'(t)$ such that $\|u(t) - p(t)\|_p = O(\varepsilon)$ and $\|u'(t) - p'(t)\|_p \le O(\varepsilon/\ell)$.

Proof. Let $\alpha = 40\ell$. Let $x(t) = (\alpha u'(\alpha t), u(\alpha t)) \in \mathbb{R}^{2n}$. Note that $x(t)$ satisfies the following ODE

$$\begin{aligned} \frac{d}{dt}x(t) &= \overline{F}(x(t),t) \text{ for } 0 \le t \le \ell \\ x(0) &= (\alpha w, v). \end{aligned} \tag{14.37}$$

where $\overline{F}(x(t),t) = (\alpha^2 F(\alpha^{-1}x_{(1)}(t), x_{(2)}(t), \alpha t), x_{(1)}(t))$, $x_{(1)}(t)$ is the first n variables of $x(t)$ and $x_{(2)}(t)$ is the last n variables of $x(t)$. Next, we verify the conditions of Theorem 14.6.4 for this ODE. Let $\overline{\ell} = \frac{1}{60}$ and $\overline{K} = 40\overline{\ell}\max_{t\in[0,\ell]} \|\overline{F}(x(0),t)\|_p$. Note that

$$\overline{K} \le \alpha^2 \max_{t\in[0,\ell]} \|F(w,v,t)\|_p + \alpha\|w\|_p.$$

For any y, z such that $\|y - x(0)\|_p \le \overline{K}$ and $\|z - x(0)\|_p \le \overline{K}$, we apply the assumption on F and get that

$$\begin{aligned} & \|\overline{F}(y,t) - \overline{F}(z,t)\|_p \\ \le\ & \alpha^2\|F(\alpha^{-1}y_{(1)}, y_{(2)}, t) - \overline{F}(\alpha^{-1}z_{(1)}, z_{(2)}, t)\|_p + \|y_{(1)} - z_{(1)}\|_p \\ \le\ & \alpha^2\left(\frac{1}{\alpha}\|\alpha^{-1}y_{(1)} - \alpha^{-1}z_{(1)}\|_p + \frac{1}{\alpha^2}\|y_{(2)} - z_{(2)}\|_p\right) + \|y_{(1)} - z_{(1)}\|_p \\ \le\ & 3\|y - z\|_p. \end{aligned}$$

Also, by our assumption on x, we have a polynomial $\overline{q} = (\alpha q'(\alpha t), q(\alpha t))$ such that

$$\|\frac{d}{dt}x(t) - \frac{d}{dt}\overline{q}(t)\|_p \le \frac{28\varepsilon}{\overline{\ell}}$$

where x is the solution of the ODE (14.37). Therefore, Theorem 14.6.4 shows that we can compute $x\left(\overline{\ell}\right)$ with $O(\varepsilon)$ error in $O(nd\log^2(dK/\varepsilon))$ time plus $O(d\log(K/\varepsilon))$ evaluations of F. Now, we can read off $u(\ell)$ from $x_{(2)}$ because

$$x_{(2)}\left(\overline{\ell}\right) = u\left(\alpha\overline{\ell}\right) = u(\ell).$$

The result for $u'(\ell)$ is similar. □

■ 14.7 Derivative Estimations

For any smooth one dimension function f, we know by Tayloer's theorem that

$$f(x) = \sum_{k=0}^{N} \frac{f^{(k)}(a)}{k!}(x-a)^k + \frac{1}{N!}\int_a^x (x-t)^N f^{(N+1)}(t)dt.$$

This formula provides a polynomial estimate of f around a. To analyze the accuracy of this estimate, we need to bound $\left|f^{(N+1)}(t)\right|$. In one dimension, we could simply give explicit formulas for the derivatives of f and use it to estimate the remainder term. However, for functions in high dimension, it is usually too tedious. In this section, we describe some techniques for bounding the derivatives of higher dimensional functions.

The derivatives of one-variable functions can be bounded via Cauchy's estimate (Theorem (14.2.11)). In Subsection 14.7.1, we give calculus rules that reduces the problem of estimating derivatives of high-dimensional functions to derivatives of one-dimensional functions. In Subsection 14.7.2, we show how to reduce bounding the derivative for an arbitrary ODE to an ODE in one dimension.

14.7.1 Explicit Function

In this subsection, we show how to bound the derivatives of a complicated explicit function using the following object, generating upper bound. We reduce estimates of the derivatives of functions in high dimension to one variable rational functions. Since rational functions are holomorphic, one can apply Cauchy's estimates (Theorem (14.2.11)) to bound their derivatives.

Definition 14.7.1. Given a function F. We call that $F \leq_x f$ for some one variable function $f : \mathbb{R} \to \mathbb{R}$ if

$$\|D^{(k)}F(x)[\Delta_1, \Delta_2, \cdots, \Delta_k]\| \leq f^{(k)}(0) \prod_{i=1}^{k} \|\Delta_i\|^k \tag{14.38}$$

for any $k \geq 0$ and any Δ_i.

Remark. In general, $F \leq_x f$ and $f(t) \leq g(t)$ point-wise does NOT imply $F \leq_x g$. However, $F \leq_x f$ and $f \leq_0 g$ does imply $F \leq_x g$.

This concept is useful for us to reduce bounding derivatives of a high dimension function to bounding derivatives of 1 dimension function. First of all, we note that upper bounds are composable.

Lemma 14.7.2. *Given $F \leq_x f$ and $G \leq_{F(x)} g$, we have that*

$$G \circ F \leq_x g \circ \overline{f}$$

where $\overline{f}(s) = f(s) - f(0)$.

Proof. Fix any $\Delta_1, \Delta_2, \cdots$ be unit vectors in the domain of F. Let $H(x) = G \circ F(x)$. By chain rule, we have that

$$\begin{aligned}
DH(x)[\Delta_1] &= DG(F(x))[DF(x)[\Delta_1]], \\
DH(x)[\Delta_1, \Delta_2] &= DG(F(x))[D^2F(x)[\Delta_1, \Delta_2]] \\
&\quad + D^2G(F(x))[DF(x)[\Delta_1], DF(x)[\Delta_2]], \\
DH(x)[\Delta_1, \Delta_2, \Delta_3] &= DG(F(x))[D^2F(x)[\Delta_1, \Delta_2, \Delta_3]] \\
&\quad + D^2G(F(x))[D^2F(x)[\Delta_1, \Delta_2], DF(x)[\Delta_3]] \\
&\quad + D^2G(F(x))[D^2F(x)[\Delta_1, \Delta_3], DF(x)[\Delta_2]] \\
&\quad + D^2G(F(x))[D^2F(x)[\Delta_2, \Delta_3], DF(x)[\Delta_1]] \\
&\quad + D^3G(F(x))[DF(x)[\Delta_1], DF(x)[\Delta_2], DF(x)[\Delta_3]], \\
&\vdots
\end{aligned}$$

Since $G \leq_{F(x)} g$, equation (14.38) shows that

$$\begin{aligned}
\|DH(x)[\Delta_1]\| &\leq g^{(1)}(0)\|DF(x)[\Delta_1]\|, \\
\|D^2H(x)[\Delta_1, \Delta_2]\| &\leq g^{(1)}(0)\|D^2F(x)[\Delta_1, \Delta_2]\| \\
&\quad + g^{(2)}(0)\|DF(x)[\Delta_1]\|_2\|DF(x)[\Delta_2]\|, \\
\|D^3H(x)[\Delta_1, \Delta_2, \Delta_3]\| &\leq g^{(1)}(0)\|D^2F(x)[\Delta_1, \Delta_2, \Delta_3]\| \\
&\quad + g^{(2)}(0)\|D^2F(x)[\Delta_1, \Delta_2]\|\|DF(x)[\Delta_3]\| \\
&\quad + g^{(2)}(0)\|D^2F(x)[\Delta_1, \Delta_3]\|\|DF(x)[\Delta_2]\| \\
&\quad + g^{(2)}(0)\|D^2F(x)[\Delta_2, \Delta_3]\|\|DF(x)[\Delta_1]\| \\
&\vdots \quad + g^{(3)}(0)\|DF(x)[\Delta_1]\|\|DF(x)[\Delta_2]\|\|DF(x)[\Delta_3]\|.
\end{aligned}$$

Now, we use $F \leq_x f$ to get

$$\begin{aligned} \|DH(x)[\Delta_1]\| &\leq g^{(1)}(0)f^{(1)}(0) = \left(g \circ \overline{f}\right)^{(1)}(0) \\ \|D^2H(x)[\Delta_1,\Delta_2]\| &\leq g^{(1)}(0)f^{(2)}(0) + g^{(2)}(0)f^{(1)}(0)^2 = \left(g \circ \overline{f}\right)^{(2)}(0), \\ \|D^3H(x)[\Delta_1,\Delta_2,\Delta_3]\| &\leq g^{(1)}(0)f^{(3)}(0) + 2g^{(2)}(0)f^{(2)}(0)f^{(1)}(0) \\ \vdots \quad & \quad + g^{(3)}(0)f^{(1)}(0)^3 = \left(g \circ \overline{f}\right)^{(3)}(0). \end{aligned}$$

Therefore, we have that $\|D^kH(x)[\Delta_i]\| \leq \left(g \circ \overline{f}\right)^{(k)}(0)$ for all $k \geq 1$. For $k = 0$, we have that $\|H(x)\| = \|G(F(x))\| \leq g(0) = g(\overline{f}(0))$. □

Next, we give some extra calculus rule for generating upper bounds.

Lemma 14.7.3. *Given that $H_i \leq_x h_i$ for all $i = 1, \cdots, k$. Then, we have that*

$$\sum_{i=1}^k H_i \leq_x \sum_{i=1}^k h_i \text{ and } \prod_{i=1}^k H_i \leq_x \prod_{i=1}^k h_i.$$

Given that $H \leq_x h$ and $\|H^{-1}(x)\| \leq C$, we have that

$$H^{-1} \leq_x \frac{1}{C^{-1} - (h(s) - h(0))}.$$

Proof. Fix $\Delta_1, \Delta_2, \cdots$ be unit vectors. For the first claim, let $H = \sum H_i$, we note that

$$D^jH[\Delta_1, \cdots, \Delta_j] = \sum_{i=1}^k D^jH_i[\Delta_1, \cdots, \Delta_j].$$

Therefore, we have that $\|D^jH[\Delta_1, \cdots, \Delta_j]\| \leq \sum_{i=1}^k \|D^jH_i[\Delta_1, \cdots, \Delta_j]\|$. Since $H \leq_x h_i$, we have that $\|D^jH[\Delta_1, \cdots, \Delta_j]\| \leq \sum_{i=1}^k h^{(j)}(0)$. Hence, we have $H \leq_x \sum_{i=1}^k h_i$.

For the second claim, we note that

$$D^jG = \sum_{i_1+i_2+\cdots+i_k=j} \prod_{l=1}^k D^{i_l}H_l.$$

Let $g = \prod_{i=1}^k h_i$. Then, we have that

$$\|D^jG\| \leq \sum_{i_1+i_2+\cdots+i_k=j} \prod_{l=1}^k \|D^{i_l}H_l\| \leq \sum_{i_1+i_2+\cdots+i_k=j} \prod_{l=1}^k h_i^{(i_l)}(0) = D^jg(0).$$

Hence, $G \leq_x g$.

For the last claim, we first consider the function $\Phi(M) = M^{-1}$. Note that $D\Phi[\Delta_1] = -M^{-1}\Delta_1M^{-1}$, $D^2\Phi[\Delta] = M^{-1}\Delta_1M^{-1}\Delta_2M^{-1} + M^{-1}\Delta_2M^{-1}\Delta_1M^{-1}$ and hence

$$\|D^j\Phi\| \leq j!\|M^{-1}\|^{j+1} = j!C^{j+1}$$

Hence, we have $\Phi \leq_M \sum_{j=0}^\infty \frac{j!C^{j+1}}{j!}s^j = \frac{1}{C^{-1}-s}$. By Lemma 14.7.2, we see that $H^{-1} \leq_x \frac{1}{C^{-1}-(h(s)-h(0))}$. □

14.7.2 Explicit ODE

In this section, we study the Taylor expansion of the solution of ODE (14.31).

Lemma 14.7.4. *Let $u(t)$ be the solution of the ODE $u'(t) = F(u(t))$. Suppose that $F \leq_{u(0)} f$ and let $\psi(t)$ be the solution of the ODE $\psi'(t) = f(\psi(t))$ and $\psi(0) = 0$. Then, we have*

$$\|u^{(k)}(0)\|_2 \leq \psi^{(k)}(0)$$

for all $k \geq 1$.

Proof. Since $u'(t) = F(u(t))$, we have that

$$\begin{aligned}
u^{(2)}(t) &= DF(u(t))[u^{(1)}(t)], \\
u^{(3)}(t) &= DF(u(t))[u^{(2)}(t)] + D^2F(u(t))[u^{(1)}(t), u^{(1)}(t)], \\
u^{(4)}(t) &= DF(u(t))[u^{(3)}(t)] + 2D^2F(u(t))[u^{(2)}(t), u^{(1)}(t)] \\
&\quad + D^3F(u(t))[u^{(1)}(t), u^{(1)}(t), u^{(1)}(t)] \\
&\vdots
\end{aligned}$$

Therefore, we have

$$\begin{aligned}
\|u^{(2)}(0)\|_2 &\leq f^{(1)}(0)\|u^{(1)}(0)\|_2, \\
\|u^{(3)}(0)\|_2 &\leq f^{(1)}(0)\|u^{(2)}(0)\|_2 + f^{(2)}(0)\|u^{(1)}(0)\|_2^2, \\
\|u^{(4)}(0)\|_2 &\leq f^{(1)}(0)\|u^{(3)}(0)\|_2 + 2f^{(2)}(0)\|u^{(2)}(0)\|_2\|u^{(1)}(0)\|_2 + f^{(3)}(0)\|u^{(1)}(0)\|_2^3, \\
&\vdots
\end{aligned}$$

By expanding $\psi'(t) = f(\psi(t))$ at $t = 0$, we see that

$$\begin{aligned}
\psi^{(2)}(0) &= f^{(1)}(0)\psi^{(1)}(0), \\
\psi^{(3)}(0) &= f^{(1)}(0)\psi^{(2)}(0) + f^{(2)}(0)\left(\psi^{(1)}(0)\right)^2, \\
\psi^{(4)}(0) &= f^{(1)}(0)\psi^{(3)}(0) + 2f^{(2)}(0)\psi^{(2)}(0)\psi^{(1)}(0) + f^{(2)}(0)\left(\psi^{(1)}(0)\right)^3, \\
&\vdots
\end{aligned}$$

Since $\|u^{(1)}(0)\|_2 = \|F(u(0))\|_2 \leq f(0) = \psi^{(1)}(0)$, we have that $\|u^{(k)}(0)\|_2 \leq \psi^{(k)}(0)$ for all $k \geq 1$. □

Now, we apply Lemma 14.7.4 to second order ODE.

Lemma 14.7.5. *Let $u(t)$ be the solution of the ODE $u''(t) = F(u'(t), u(t), t)$. Given some $\alpha > 0$ and define $\overline{F}(y, x, t) = \alpha^2 F(\alpha^{-1}y(t) + u'(0), x(t) + \alpha t u'(0), \alpha t)$. Suppose that $\overline{F} \leq_{(0,u(0),0)} f$ and let $\psi(t)$ be the solution of the ODE $\psi'(t) = 1 + \psi(t) + f(\psi(t))$ and $\psi(0) = 0$. Then, we have*

$$\|u^{(k)}(0)\|_2 \leq \frac{\psi^{(k)}(0)}{\alpha^k}$$

for all $k \geq 2$.

Proof. Let $x(t) = u(\alpha t) - \alpha t u'(0)$ and $y(t) = \alpha u'(\alpha t) - \alpha u'(0)$. Then, we can write the problem into

first order ODE

$$\begin{aligned} y'(t) &= \alpha^2 F(\alpha^{-1}y(t) + u'(0), x(t) + \alpha t u'(0), \alpha t) = \overline{F}(y(t), x(t), t), \\ x'(t) &= y(t), \\ t' &= 1. \end{aligned}$$

Let $\mathbf{F}(y, x, t) = (\overline{F}(y, x, t), y, 1)$. Then, we have that $\|D^k\mathbf{F}\| \leq \|D^k 1\| + \|D^k y\| + \|D^k\overline{F}\|$. Using $\overline{F} \leq_{(0,u(0),0)} f$, we have that

$$\mathbf{F} \leq_{(0,x(0),0)} 1 + t + f.$$

By Lemma 14.7.4, we know that

$$\|u^{(k)}(0)\|_2 = \frac{\|x^{(k)}(0)\|_2}{\alpha^k} \leq \frac{\psi^{(k)}(0)}{\alpha^k}.$$

□

Bibliography

[1] Jacob Abernethy and Elad Hazan. "Faster Convex Optimization: Simulated Annealing with an Efficient Universal Barrier". In: *arXiv preprint arXiv:1507.02528* (2015).

[2] Dimitris Achlioptas. "Database-friendly random projections: Johnson-Lindenstrauss with binary coins". In: *Journal of computer and System Sciences* 66.4 (2003), pp. 671–687.

[3] Rudolf Ahlswede and Andreas Winter. "Strong converse for identification via quantum channels". In: *Information Theory, IEEE Transactions on* 48.3 (2002), pp. 569–579.

[4] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network flows*. New York, NY, USA: Elsevier North-Holland, Inc., 1989.

[5] Ravindra K. Ahuja et al. "Applications of Network Optimization". In: *Network Models*. Vol. 7. Handbooks in Operations Research and Management Science. North-Holland, 1995, pp. 1–75.

[6] Martin Aigner and Thomas A Dowling. "Matching theory for combinatorial geometries". In: *Transactions of the American Mathematical Society* 158.1 (1971), pp. 231–245.

[7] Boris Alexeev et al. "Phase retrieval with polarization". In: *SIAM Journal on Imaging Sciences* 7.1 (2014), pp. 35–66.

[8] Zeyuan Allen-Zhu, Yin Tat Lee, and Lorenzo Orecchia. "Using Optimization to Obtain a Width-Independent, Parallel, Simpler, and Faster Positive SDP Solver". In: *arXiv preprint arXiv:1507.02259* (2015).

[9] Zeyuan Allen-Zhu, Zhenyu Liao, and Lorenzo Orecchia. "Spectral Sparsification and Regret Minimization Beyond Multiplicative Updates". In: *Proceedings of the 47th Annual ACM Symposium on Theory of Computing (STOC)*. 2015, pp. 237–245.

[10] Kurt M. Anstreicher. "Large step volumetric potential reduction algorithms for linear programming". In: *Annals of Operations Research* 62.1 (1996), pp. 521–538.

[11] Kurt M Anstreicher. "On Vaidya's volumetric cutting plane method for convex programming". In: *Mathematics of Operations Research* 22.1 (1997), pp. 63–89.

[12] Kurt M. Anstreicher. "The volumetric barrier for convex quadratic constraints". In: *Mathematical Programming* 100.3 (2004), pp. 613–662.

[13] Kurt M. Anstreicher. "The volumetric barrier for semidefinite programming". In: *Mathematics of Operations Research* 25.3 (2000), pp. 365–380.

[14] Kurt M. Anstreicher. "Towards a practical volumetric cutting plane method for convex programming". In: *SIAM Journal on Optimization* 9.1 (1998), pp. 190–206.

[15] Kurt M. Anstreicher. "Volumetric path following algorithms for linear programming". In: *Math. Program.* 76 (1996), pp. 245–263.

[16] Mica Arie-Nachimson et al. "Global motion estimation from point matches". In: *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012 Second International Conference on*. IEEE. 2012, pp. 81–88.

[17] Sanjeev Arora, Elad Hazan, and Satyen Kale. "Fast algorithms for approximate semidefinite programming using the multiplicative weights update method". In: *Foundations of Computer Science, 2005. FOCS 2005. 46th Annual IEEE Symposium on*. IEEE. 2005, pp. 339–348.

[18] Sanjeev Arora, Elad Hazan, and Satyen Kale. "The Multiplicative Weights Update Method: A Meta-Algorithm and Applications".

[19] Sanjeev Arora and Satyen Kale. "A combinatorial, primal-dual approach to semidefinite programs". In: *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*. ACM. 2007, pp. 227–236.

[20] David S Atkinson and Pravin M Vaidya. "A cutting plane algorithm for convex programming that uses analytic centers". In: *Mathematical Programming* 69.1-3 (1995), pp. 1–43.

[21] Y. Aumann and Y. Rabani. "An O(log k) Approximate Min-Cut Max-Flow Theorem and Approximation Algorithm". In: *SIAM Journal on Computing* 27.1 (1998), pp. 291–301.

[22] Haim Avron, Petar Maymounkov, and Sivan Toledo. "Blendenpik: Supercharging LAPACK's least-squares solver". In: *SIAM J. Scientific Computing* 32.3 (2010), pp. 1217–1236.

[23] Haim Avron and Sivan Toledo. "Effective stiffness: Generalizing effective resistance sampling to finite element matrices". In: *arXiv preprint arXiv:1110.4437* (2011).

[24] Francis Bach. "Learning with submodular functions: A convex optimization perspective". In: *Foundations and Trends in Machine Learning* (2013).

[25] Mihai Badoiu, Sariel Har-Peled, and Piotr Indyk. "Approximate clustering via core-sets". In: *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*. 2002, pp. 250–257.

[26] Olivier Bahn et al. "A cutting plane method from analytic centers for stochastic programming". In: *Mathematical Programming* 69.1-3 (1995), pp. 45–73.

[27] Chanderjit Bajaj. "The algebraic degree of geometric optimization problems". English. In: *Discrete & Computational Geometry* 3.2 (1988), pp. 177–191.

[28] Egon Balas and Chang-Sung Yu. "A Note on the Weiszfeld-Kuhn Algorithm for the General Fermat Problem". In: *Managme Sci Res Report* 484 (1982), pp. 1–6.

[29] Afonso S Bandeira, Amit Singer, and Daniel A Spielman. "A Cheeger inequality for the graph connection Laplacian". In: *SIAM Journal on Matrix Analysis and Applications* 34.4 (2013), pp. 1611–1630.

[30] Francisco Barahona and William H Cunningham. "A submodular network simplex method". In: *Mathematical Programming at Oberwolfach II*. Springer, 1984, pp. 9–31.

[31] Joshua Batson, Daniel Spielman, and Nikhil Srivastava. "Twice-Ramanujan sparsifiers". In: *SIAM Journal on Computing* 41.6 (2012), pp. 1704–1721.

[32] Stephen R Becker, Emmanuel J Candès, and Michael C Grant. "Templates for convex cone problems with applications to sparse signal recovery". In: *Mathematical programming computation* 3.3 (2011), pp. 165–218.

[33] András A Benczúr and David R Karger. "Approximating st minimum cuts in $\tilde{O}$ (n 2) time". In: *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*. ACM. 1996, pp. 47–55.

[34] M. Bern et al. "Support-graph preconditioners". In: *SIAM J. Matrix Anal. & Appl* 27.4 (2006), pp. 930–951.

[35] Dimitri Bertsekas. "Nonlinear programming". In: (1999).

[36] Dimitris Bertsimas and Santosh Vempala. "Solving convex programs by random walks". In: *Journal of the ACM (JACM)* 51.4 (2004), pp. 540–556.

[37] Prosenjit Bose, Anil Maheshwari, and Pat Morin. "Fast approximations for sums of distances, clustering and the Fermat-Weber problem". In: *Computational Geometry* 24.3 (2003), pp. 135–146.

[38] Jean Bourgain, Joram Lindenstrauss, and V Milman. "Approximation of zonoids by zonotopes". In: *Acta mathematica* 162.1 (1989), pp. 73–141.

[39] Carl Brezovec, Gerard Cornuéjols, and Fred Glover. "Two algorithms for weighted matroid intersection". In: *Mathematical Programming* 36.1 (1986), pp. 39–53.

[40] Sébastien Bubeck. "Convex Optimization: Algorithms and Complexity". In: *Foundations and Trends in Machine Learning* 8.3-4 (2015), pp. 231–357.

[41] Sébastien Bubeck. "Theory of Convex Optimization for Machine Learning". In: *Arxiv preprint arXiv:1405.4980* (2014).

[42] Sébastien Bubeck and Ronen Eldan. "The entropic barrier: a simple and optimal universal self-concordant barrier". In: *arXiv preprint arXiv:1412.1587* (2014).

[43] Sébastien Bubeck, Ronen Eldan, and Joseph' Lehec. "Sampling from a log-concave distribution with Projected Langevin Monte Carlo". In: *arXiv preprint arXiv:1507.02564* (2015).

[44] Yang Cai, Constantinos Daskalakis, and S Matthew Weinberg. "Optimal multi-dimensional mechanism design: Reducing revenue to welfare maximization". In: *Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on*. IEEE. 2012, pp. 130–139.

[45] Yang Cai, Constantinos Daskalakis, and S Matthew Weinberg. "Reducing revenue to welfare maximization: Approximation algorithms and other generalizations". In: *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM. 2013, pp. 578–595.

[46] Marcel K. de Carli Silva, Nicholas J. A. Harvey, and Cristiane M. Sato. "Sparse Sums of Positive Semidefinite Matrices". In: *CoRR* abs/1107.0088 (2011).

[47] R. Chandrasekaran and A. Tamir. "Open questions concerning Weiszfeld's algorithm for the Fermat-Weber location problem". English. In: *Mathematical Programming* 44.1-3 (1989), pp. 293–295.

[48] Chih-Chung Chang and Chih-Jen Lin. "LIBSVM: A library for support vector machines". In: *ACM Transactions on Intelligent Systems and Technology (TIST)*. 2.3 (2011). Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm., p. 27.

[49] L. Paul Chew. "There Are Planar Graphs Almost as Good as the Complete Graph". In: *Journal of Computer and System Sciences* 39.2 (1989), pp. 205–219.

[50] Hui Han Chin et al. "Runtime guarantees for regression problems". In: *ITCS*. 2013, pp. 269–282.

[51] Paul Christiano et al. "Electrical flows, laplacian systems, and faster approximation of maximum flow in undirected graphs". In: *Proceedings of the 43rd annual ACM symposium on Theory of computing*. ACM. 2011, pp. 273–282.

[52] Nam-Kee Chung and Dong-Wan Tcha. "A dual algorithm for submodular flow problems". In: *Operations research letters* 10.8 (1991), pp. 489–495.

[53] Kenneth Clarkson and David Woodruff. "Low rank approximation and regression in input sparsity time". In: *Proceedings of the 45th annual ACM symposium on Symposium on theory of computing*. ACM. 2013, pp. 81–90.

[54] Kenneth Clarkson and David Woodruff. "Numerical linear algebra in the streaming model". In: *Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC)*. 2009, pp. 205–214.

[55] E. S. Coakley, Vladimir Rokhlin, and Mark Tygert. "A Fast Randomized Algorithm for Orthogonal Projection". In: *SIAM J. Scientific Computing* 33.2 (2011), pp. 849–868.

[56] Michael B Cohen and Richard Peng. "Lp Row Sampling by Lewis Weights". In: *arXiv preprint arXiv:1412.0588* (2014).

[57] Michael B. Cohen et al. "Solving SDD Linear Systems in Nearly Mlog1/2N Time". In: *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*. STOC '14. New York, New York: ACM, 2014, pp. 343–352.

[58] Leon Cooper and I.Norman Katz. "The Weber problem revisited". In: *Computers and Mathematics with Applications* 7.3 (1981), pp. 225–234.

[59] B. Cousins and S. Vempala. "A Cubic Algorithm for Computing Gaussian Volume". In: *SODA*. 2014, pp. 1215–1228.

[60] William H Cunningham. "Improved bounds for matroid partition and intersection algorithms". In: *SIAM Journal on Computing* 15.4 (1986), pp. 948–957.

[61] William H Cunningham. "On submodular function minimization". In: *Combinatorica* 5.3 (1985), pp. 185–192.

[62] William H Cunningham and András Frank. "A primal-dual algorithm for submodular flows". In: *Mathematics of Operations Research* 10.2 (1985), pp. 251–262.

[63] Samuel I Daitch and Daniel A Spielman. "Faster approximate lossy generalized flow via interior point algorithms". In: *Proceedings of the 40th annual ACM symposium on Theory of computing*. ACM. 2008, pp. 451–460.

[64] Arnak S Dalalyan. "Theoretical guarantees for approximate sampling from smooth and log-concave densities". In: *arXiv preprint arXiv:1412.7392* (2014).

[65] Constantinos Daskalakis and S Matthew Weinberg. "Bayesian truthful mechanisms for job scheduling from bi-criterion approximation algorithms". In: *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM. 2015, pp. 1934–1952.

[66] James Demmel et al. "Fast matrix multiplication is stable". In: *Numerische Mathematik* 106.2 (2007), pp. 199–224.

[67] A. B. Dieker. "Reflected Brownian Motion". In: *Wiley Encyclopedia of Operations Research and Management Science*. John Wiley & Sons, Inc., 2010.

[68] EA Dinic. "An algorithm for the solution of the max-flow problem with the polynomial estimation". In: *Doklady Akademii Nauk SSSR* 194.4 (1970), pp. 1277–1280.

[69] Zvi Drezner et al. "Facility location". In: Springer, 2002. Chap. The Weber problem, pp. 1–36.

[70] Petros Drineas, Ravi Kannan, and Michael W. Mahoney. "Fast Monte Carlo algorithms for matrices I: Approximating matrix multiplication". In: *SIAM J. Comput.* 36.1 (2006). Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 132–157.

[71] Petros Drineas, Ravi Kannan, and Michael W. Mahoney. "Fast Monte Carlo Algorithms for Matrices II: Computing a Low-Rank Approximation to a Matrix". In: *SIAM J. Comput.* 36.1 (2006), pp. 158–183.

[72] Petros Drineas, Ravi Kannan, and Michael W. Mahoney. "Fast Monte Carlo Algorithms for Matrices III: Computing a Compressed Approximate Matrix Decomposition". In: *SIAM J. Comput.* 36.1 (2006). Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 184–206.

[73] Petros Drineas and Michael W Mahoney. "Effective resistances, statistical leverage, and applications to linear equation solving". In: *arXiv preprint arXiv:1005.3097* (2010).

[74] Petros Drineas, Michael W. Mahoney, and S. Muthukrishnan. "Relative-Error CUR Matrix Decompositions". In: *SIAM J. Matrix Anal. Appl.* 30.2 (2008), pp. 844–881.

[75] Petros Drineas, Michael W. Mahoney, and S. Muthukrishnan. "Sampling Algorithms for ℓ_2 Regression and Applications". In: *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2006, pp. 1127–1136.

[76] Petros Drineas, MichaelW. Mahoney, and S. Muthukrishnan. "Subspace Sampling and Relative-Error Matrix Approximation: Column-Based Methods". In: *Proceedings of the 10th International Workshop on Randomization and Computation (RANDOM)*. 2006, pp. 316–326.

[77] A Dutt, M Gu, and V Rokhlin. "Fast algorithms for polynomial interpolation, integration, and differentiation". In: *SIAM Journal on Numerical Analysis* 33.5 (1996), pp. 1689–1711.

[78] M. E. Dyer and A. M. Frieze. "Computing the volume of a convex body: a case where randomness provably helps". In: *Proc. of AMS Symposium on Probabilistic Combinatorics and Its Applications*. 1991, pp. 123–170.

[79] M. E. Dyer, A. M. Frieze, and R. Kannan. "A random polynomial-time algorithm for approximating the volume of convex bodies". In: *J. ACM* 38.1 (1991), pp. 1–17.

[80] Jack Edmonds. "Matroid intersection". In: *Annals of discrete Mathematics* 4 (1979), pp. 39–49.

[81] Jack Edmonds. "Matroid partition". In: *Mathematics of the Decision Sciences* 11 (1968), pp. 335–345.

[82] Jack Edmonds. "Submodular functions, matroids, and certain polyhedra". In: *Edited by G. Goos, J. Hartmanis, and J. van Leeuwen* (1970), p. 11.

[83] Jack Edmonds and Rick Giles. "A min-max relation for submodular functions on graphs". In: *Studies in Integer Programming (PL Hammer, EL Johnson and BH Korte, eds.), Ann. Discrete Math* 1 (1977), pp. 185–204.

[84] Jack Edmonds and Richard M Karp. "Theoretical improvements in algorithmic efficiency for network flow problems". In: *Journal of the ACM (JACM)* 19.2 (1972), pp. 248–264.

[85] Shimon Even and R. Endre Tarjan. "Network Flow and Testing Graph Connectivity". In: *SIAM Journal on Computing* 4.4 (Dec. 1975), pp. 507–518.

[86] Dan Feldman and Michael Langberg. "A unified framework for approximating and clustering data". In: *Proceedings of the forty-third annual ACM symposium on Theory of computing*. ACM. 2011, pp. 569–578.

[87] Lisa Fleischer and Satoru Iwata. "A push-relabel framework for submodular function minimization and applications to parametric optimization". In: *Discrete Applied Mathematics* 131.2 (2003), pp. 311–322.

[88] Lisa Fleischer and Satoru Iwata. "Improved algorithms for submodular function minimization and submodular flow". In: *Proceedings of the thirty-second annual ACM symposium on Theory of computing*. ACM. 2000, pp. 107–116.

[89] Lisa Fleischer, Satoru Iwata, and S Thomas McCormick. "A faster capacity scaling algorithm for minimum cost submodular flow". In: *Mathematical Programming* 92.1 (2002), pp. 119–139.

[90] András Frank. "A weighted matroid intersection algorithm". In: *Journal of Algorithms* 2.4 (1981), pp. 328–336.

[91] András Frank and Éva Tardos. "An application of simultaneous Diophantine approximation in combinatorial optimization". In: *Combinatorica* 7.1 (1987), pp. 49–65.

[92] S. Fujishige. "Algorithms for solving the independent-flow problems". In: *Journal of the Operations Research Society of Japan* (1978).

[93] Satoru Fujishige. "An out-of-kilter method for submodular flows". In: *Discrete applied mathematics* 17.1 (1987), pp. 3–16.

[94] Satoru Fujishige and Satoru Iwata. "Algorithms for submodular flows". In: *IEICE TRANSACTIONS on Information and Systems* 83.3 (2000), pp. 322–329.

[95] Satoru Fujishige, Hans Röck, and Uwe Zimmermann. "A strongly polynomial algorithm for minimum cost submodular flow problems". In: *Mathematics of Operations Research* 14.1 (1989), pp. 60–69.

[96] Satoru Fujishige and Zhang Xiaodong. "An efficient cost scaling algorithm for the independent assignment problem". In: *Journal of the Operations Research Society of Japan* 38.1 (1995), pp. 124–136.

[97] Mituhiro Fukuda et al. "Exploiting sparsity in semidefinite programming via matrix completion I: General framework". In: *SIAM Journal on Optimization* 11.3 (2001), pp. 647–674.

[98] Zvi Galil and Éva Tardos. "An O (n 2 (m+ n log n) log n) min-cost flow algorithm". In: *Journal of the ACM (JACM)* 35.2 (1988), pp. 374–386.

[99] François Le Gall. "Powers of tensors and fast matrix multiplication". In: *arXiv preprint arXiv:1401.7714* (2014).

[100] François Le Gall. "Powers of tensors and fast matrix multiplication". In: *arXiv:1401.7714* (2014).

[101] Jean-Louis Goffin, Zhi-Quan Luo, and Yinyu Ye. "Complexity analysis of an interior cutting plane method for convex feasibility problems". In: *SIAM Journal on Optimization* 6.3 (1996), pp. 638–652.

[102] Jean-Louis Goffin and Jean-Philippe Vial. "Convex nondifferentiable optimization: A survey focused on the analytic center cutting plane method". In: *Optimization Methods and Software* 17.5 (2002), pp. 805–867.

[103] Jean-Louis Goffin and Jean-Philippe Vial. "Shallow, deep and very deep cuts in the analytic center cutting plane method". In: *Mathematical Programming* 84.1 (1999), pp. 89–103.

[104] Andrew V. Goldberg and Satish Rao. "Beyond the flow decomposition barrier". In: *J. ACM* 45.5 (1998), pp. 783–797.

[105] Andrew V Goldberg and Robert E Tarjan. "A new approach to the maximum-flow problem". In: *Journal of the ACM (JACM)* 35.4 (1988), pp. 921–940.

[106] Andrew V Goldberg and Robert E Tarjan. "Finding minimum-cost circulations by successive approximation". In: *Mathematics of Operations Research* 15.3 (1990), pp. 430–466.

[107] Clovis C Gonzaga. "Path-following methods for linear programming". In: *SIAM review* 34.2 (1992), pp. 167–224.

[108] Clóvis C Gonzaga and Elizabeth W Karas. "Fine tuning Nesterov's steepest descent algorithm for differentiable convex programming". In: *Mathematical Programming* 138.1-2 (2013), pp. 141–166.

[109] Martin Grötschel, László Lovász, and Alexander Schrijver. "Geometric Algorithms and Combinatorial Optimization". In: Springer, 1988.

[110] Martin Grötschel, László Lovász, and Alexander Schrijver. "The ellipsoid method and its consequences in combinatorial optimization". In: *Combinatorica* 1.2 (1981), pp. 169–197.

[111] B. Grünbaum. "Partitions of mass-distributions and of convex bodies by hyperplanes". In: *Pacific J. Math* 10.4 (1960), pp. 1257–1261.

[112] N. Halko, P. G. Martinsson, and J. A. Tropp. "Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions". In: *SIAM Review* 53.2 (2011), pp. 217–288.

[113] Sariel Har-Peled and Akash Kushal. "Smaller coresets for k-median and k-means clustering". In: *Proceedings of the twenty-first annual symposium on Computational geometry*. ACM. 2005, pp. 126–134.

[114] J. Michael Harrison. *Brownian motion and stochastic flow systems*. Wiley series in probability and mathematical statistics. New York: Wiley, 1985.

[115] Christoph Helmberg and Franz Rendl. "A spectral bundle method for semidefinite programming". In: *SIAM Journal on Optimization* 10.3 (2000), pp. 673–696.

[116] P. Indyk and Stanford University. Computer Science Dept. *High-dimensional computational geometry*. Stanford University, 2000.

[117] Satoru Iwata. "A capacity scaling algorithm for convex cost submodular flows". In: *Mathematical programming* 76.2 (1997), pp. 299–308.

[118] Satoru Iwata. "A faster scaling algorithm for minimizing submodular functions". In: *SIAM Journal on Computing* 32.4 (2003), pp. 833–840.

[119] Satoru Iwata. "A fully combinatorial algorithm for submodular function minimization". In: *Journal of Combinatorial Theory, Series B* 84.2 (2002), pp. 203–212.

[120] Satoru Iwata. "Submodular function minimization". In: *Mathematical Programming* 112.1 (2008), pp. 45–64.

[121] Satoru Iwata, Lisa Fleischer, and Satoru Fujishige. "A combinatorial strongly polynomial algorithm for minimizing submodular functions". In: *Journal of the ACM (JACM)* 48.4 (2001), pp. 761–777.

[122] Satoru Iwata, S Thomas McCormick, and Maiko Shigeno. "A fast cost scaling algorithm for submodular flow". In: *Information Processing Letters* 74.3 (2000), pp. 123–128.

[123] Satoru Iwata, S Thomas McCormick, and Maiko Shigeno. "A Faster Algorithm for Minimum Cost Submodular Flows." In: *SODA*. 1998, pp. 167–174.

[124] Satoru Iwata, S Thomas McCormick, and Maiko Shigeno. "A strongly polynomial cut canceling algorithm for the submodular flow problem". In: *Integer Programming and Combinatorial Optimization*. Springer, 1999, pp. 259–272.

[125] Satoru Iwata and James B Orlin. "A simple combinatorial algorithm for submodular function minimization". In: *Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics. 2009, pp. 1230–1237.

[126] Rahul Jain and Penghui Yao. "A parallel approximation algorithm for positive semidefinite programming". In: *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*. IEEE. 2011, pp. 463–471.

[127] Klaus Jansen. "Approximate strong separation with application in fractional graph coloring and preemptive scheduling". In: *Theoretical Computer Science* 302.1 (2003), pp. 239–256.

[128] Fritz John. "Extremum problems with inequalities as side conditions". In: *Studies and Essays presented to R. Courant on his 60th Birthday* (1948).

[129] R. Johnson and T. Zhang. "Accelerating Stochastic Gradient Descent using Predictive Variance Reduction". In: *Advances in Neural Information Processing Systems (NIPS)*. 2013.

[130] William Johnson and Joram Lindenstrauss. "Extensions of Lipschitz mappings into a Hilbert space". In: *Conference on modern analysis and probability*. Vol. 26. Contemporary Mathematics. 1984, pp. 189–206.

[131] R. Kannan, L. Lovász, and M. Simonovits. "Random walks and an $O^*(n^5)$ volume algorithm for convex bodies". In: *Random Structures and Algorithms* 11 (1997), pp. 1–50.

[132] R. Kannan and H. Narayanan. "Random walks on polytopes and an affine interior point method for linear programming". In: *STOC*. 2009, pp. 561–570.

[133] Ravindran Kannan and Hariharan Narayanan. "Random walks on polytopes and an affine interior point method for linear programming". In: *Mathematics of Operations Research* 37.1 (2012), pp. 1–20.

[134] David R Karger. "Better random sampling algorithms for flows in undirected graphs". In: *Proceedings of the ninth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics. 1998, pp. 490–499.

[135] David Karger and Matthew Levine. "Random sampling in residual graphs". In: *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*. ACM. 2002, pp. 63–66.

[136] Narendra Karmarkar. "A new polynomial-time algorithm for linear programming". In: *Proceedings of the sixteenth annual ACM symposium on Theory of computing*. ACM. 1984, pp. 302–311.

[137] Narendra Karmarkar. "Riemannian geometry underlying interior-point methods for linear programming". In: *Contemporary Mathematics* 114 (1990), pp. 51–75.

[138] Richard M Karp and Christos H Papadimitriou. "On linear characterizations of combinatorial optimization problems". In: *SIAM Journal on Computing* 11.4 (1982), pp. 620–632.

[139] Jonathan A. Kelner and Petar Maymounkov. "Electric routing and concurrent flow cutting". In: *CoRR* abs/0909.2859 (2009).

[140] Jonathan A. Kelner, Gary L. Miller, and Richard Peng. "Faster approximate multicommodity flow using quadratically coupled flows". In: *Proceedings of the 44th symposium on Theory of Computing*. STOC '12. New York, New York, USA: ACM, 2012, pp. 1–18.

[141] Jonathan A. Kelner et al. "A simple, combinatorial algorithm for solving SDD systems in nearly-linear time". In: *Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC)*. 2013, pp. 911–920.

[142] Leonid G Khachiyan. "Polynomial algorithms in linear programming". In: *USSR Computational Mathematics and Mathematical Physics* 20.1 (1980), pp. 53–72.

[143] Leonid G Khachiyan. "Rounding of polytopes in the real number model of computation". In: *Mathematics of Operations Research* 21.2 (1996), pp. 307–320.

[144] LG Khachiyan, SP Tarasov, and II Erlikh. "The method of inscribed ellipsoids". In: *Soviet Math. Dokl.* Vol. 37. 1. 1988, pp. 226–230.

[145] Adam R Klivans and Daniel Spielman. "Randomness efficient identity testing of multivariate polynomials". In: *Proceedings of the thirty-third annual ACM symposium on Theory of computing*. ACM. 2001, pp. 216–223.

[146] Ioannis Koutis, Gary L. Miller, and Richard Peng. "A Nearly-m log n Time Solver for SDD Linear Systems". In: *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*. Oct. 2011, pp. 590–598.

[147] Ioannis Koutis, Gary L. Miller, and Richard Peng. "Approaching Optimality for Solving SDD Linear Systems". In: *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2010, pp. 235–244.

[148] Jakob Krarup and Steven Vajda. "On Torricelli's geometrical solution to a problem of Fermat". In: *IMA Journal of Management Mathematics* 8.3 (1997), pp. 215–224.

[149] Andreas Krause. *http://submodularity.org/*.

[150] Dilip Krishnan, Raanan Fattal, and Richard Szeliski. "Efficient preconditioning of laplacian matrices for computer graphics". In: *ACM Transactions on Graphics (TOG)* 32.4 (2013), p. 142.

[151] Kartik Krishnan and John E Mitchell. "A unifying framework for several cutting plane methods for semidefinite programming". In: *Optimization methods and software* 21.1 (2006), pp. 57–74.

[152] Kartik Krishnan and John E Mitchell. "Properties of a cutting plane method for semidefinite programming". In: *Pacific Journal of Optimization* 8.4 (2012), pp. 779–802.

[153] Kartik Krishnan and Tamás Terlaky. "Interior point and semidefinite approaches in combinatorial optimization". In: *Graph theory and combinatorial optimization*. Springer, 2005, pp. 101–157.

[154] Richard A. Kronmal and Arthur V. Peterson. "The Alias and Alias-rejection-mixture Methods for Generating Random Variables from Probability Distributions". In: *Proceedings of the 11th Conference on Winter Simulation - Volume 1*. WSC '79. San Diego, California, USA: IEEE Press, 1979, pp. 269–280.

[155] HaroldW. Kuhn. "A note on Fermat's problem". English. In: *Mathematical Programming* 4.1 (1973), pp. 98–107.

[156] Sanjiv Kumar, Mehryar Mohri, and Ameet Talwalkar. "Sampling Methods for the Nystrom Method". In: *Journal of Machine Learning Research* 13.1 (2012), pp. 981–1006.

[157] Eugene L Lawler. "Matroid intersection algorithms". In: *Mathematical programming* 9.1 (1975), pp. 31–56.

[158] Gregory Lawler and Hariharan Narayanan. "Mixing times and lp bounds for Oblivious Routing". In: *Workshop on Analytic Algorithms and Combinatorics, (ANALCO 09)*. 2009.

[159] N. Le Roux, M. Schmidt, and F. Bach. "A Stochastic Gradient Method with an Exponential Convergence Rate for Strongly-Convex Optimization with Finite Training Sets". In: *Advances in Neural Information Processing Systems (NIPS)*. 2012.

[160] Yin Tat Lee, Richard Peng, and Daniel A Spielman. "Sparsified Cholesky Solvers for SDD linear systems". In: *arXiv preprint arXiv:1506.08204* (2015).

[161] Yin Tat Lee, Satish Rao, and Nikhil Srivastava. "A New Approach to Computing Maximum Flows using Electrical Flows". In: *Proceedings of the 45th symposium on Theory of Computing - STOC '13* (2013).

[162] Yin Tat Lee and Aaron Sidford. "Efficient Accelerated Coordinate Descent Methods and Faster Algorithms for Solving Linear Systems." In: *The 54th Annual Symposium on Foundations of Computer Science (FOCS)*. 2013.

[163] Yin Tat Lee and Aaron Sidford. "Efficient Accelerated Coordinate Descent Methods and Faster Algorithms for Solving Linear Systems". In: *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science*. 2013.

[164] Yin Tat Lee and Aaron Sidford. "Efficient Inverse Maintenance and Faster Algorithms for Linear Programming". In: *arXiv:1503.01752* (2015).

[165] Yin Tat Lee and Aaron Sidford. "Path Finding I: Solving Linear Programs with\~ O (sqrt (rank)) Linear System Solves". In: *arXiv preprint arXiv:1312.6677* (2013).

[166] Yin Tat Lee and Aaron Sidford. "Path Finding II: An\~ O (m sqrt (n)) Algorithm for the Minimum Cost Flow Problem". In: *arXiv preprint arXiv:1312.6713* (2013).

[167] F. Thomson Leighton and Ankur Moitra. "Extensions and limits to vertex sparsification". In: *Proceedings of the 42nd ACM symposium on Theory of computing*. STOC '10. New York, NY, USA: ACM, 2010, pp. 47–56.

[168] A. Yu Levin. "On an algorithm for the minimization of convex functions". In: *Soviet Math. Doklady* (1965).

[169] D Lewis. "Finite dimensional subspaces of $L\_\{p\}$". In: *Studia Mathematica* 63.2 (1978), pp. 207–212.

[170] Mu Li, Gary L Miller, and Richard Peng. "Iterative row sampling". In: *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2013, pp. 127–136.

[171] E Lieb and W Thirring. "Inequalities for the moments of the eigenvalues of the Schrödinger equation and their relation to Sobolev inequalities". In: *Studies in Mathematical Physics: Essays in honor of Valentine Bargman, Lieb, E., Simon, B., Wightman, AS (eds.)* (1976), pp. 269–303.

[172] Hendrik P. Lopuhaa and Peter J. Rousseeuw. "Breakdown Points of Affine Equivariant Estimators of Multivariate Location and Covariance Matrices". In: *Ann. Statist.* 19.1 (Mar. 1991), pp. 229–248.

[173] L. Lovász. "Hit-and-Run mixes fast". In: *Math. Prog* 86 (1998), pp. 443–461.

[174] L. Lovász. "How to compute the volume?" In: *Jber. d. Dt. Math.-Verein, Jubiläumstagung 1990* (1990), pp. 138–151.

[175] L. Lovász and M. Simonovits. "Mixing rate of Markov chains, an isoperimetric inequality, and computing the volume". In: *ROCS*. 1990, pp. 482–491.

[176] L. Lovász and M. Simonovits. "On the randomized complexity of volume and diameter". In: *Proc. 33rd IEEE Annual Symp. on Found. of Comp. Sci.* 1992, pp. 482–491.

[177] L. Lovász and M. Simonovits. "Random walks in a convex body and an improved volume algorithm". In: *Random Structures and Alg.* Vol. 4. 1993, pp. 359–412.

[178] L. Lovász and S. Vempala. "Fast Algorithms for Logconcave Functions: sampling, Rounding, Integration and Optimization". In: *FOCS*. 2006, pp. 57–68.

[179] L. Lovász and S. Vempala. "Hit-and-run from a corner". In: *SIAM J. Computing* 35 (4 2006), pp. 985–1005.

[180] László Lovász and Santosh Vempala. "Simulated annealing in convex bodies and an $O^*(n^4)$ volume algorithm". In: *J. Comput. Syst. Sci.* 72.2 (2006), pp. 392–417.

[181] A. Lubotzky, R. Phillips, and P. Sarnak. "Ramanujan Graphs". In: *Combinatorica* 8.3 (1988), pp. 261–277.

[182] Aleksander Madry. "Fast approximation algorithms for cut-based problems in undirected graphs". In: *Proceedings of the 51st Annual Symposium on Foundations of Computer Science*. 2010.

[183] Aleksander Madry. "Navigating Central Path with Electrical Flows: from Flows to Matchings, and Back". In: *Proceedings of the 54th Annual Symposium on Foundations of Computer Science*. 2013.

[184] Michael W. Mahoney. "Randomized Algorithms for Matrices and Data". In: *Foundations and Trends in Machine Learning* 3.2 (2011), pp. 123–224.

[185] Michael W Mahoney and Xiangrui Meng. "Low-distortion subspace embeddings in input-sparsity time and applications to robust linear regression". In: *Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC)*. 2013, pp. 91–100.

[186] Stefano Marchesini, Yu-Chao Tu, and Hau-tieng Wu. "Alternating projection, ptychographic imaging and phase synchronization". In: *arXiv preprint arXiv:1402.0550* (2014).

[187] Adam W Marcus, Nikhil Srivastava, and Daniel A Spielman. "Interlacing Families IV: Bipartite Ramanujan Graphs of All Sizes". In: *arXiv preprint arXiv:1505.08010* (2015). to appear in FOCS 2015.

[188] G. A. Margulis. "Explicit group theoretical constructions of combinatorial schemes and their application to the design of expanders and concentrators". In: *Problems of Information Transmission* 24.1 (July 1988), pp. 39–46.

[189] S McCormick. *Submodular Function Minimization*. 2013.

[190] S Thomas and McCormick. "Canceling most helpful total submodular cuts for submodular flow." In: *IPCO*. 1993, pp. 343–353.

[191] J. A. Meijerink and H. A. van der Vorst. "An Iterative Solution Method for Linear Systems of Which the Coefficient Matrix is a Symmetric M-Matrix". In: *Mathematics of Computation* 31.137 (1977), pp. 148–162.

[192] Xiangrui Meng, Michael A. Saunders, and Michael W. Mahoney. "LSRN: A Parallel Iterative Solver for Strongly Over- or Under-Determined Systems". In: *SIAM J. Scientific Computing* 36.2 (2014).

[193] Carl D Meyer Jr. "Generalized inversion of modified matrices". In: *SIAM J. Appl. Math* 24.3 (1973), pp. 315–323.

[194] Gary L. Miller and Richard Peng. "Approximate Maximum Flow on Separable Undirected Graphs". In: *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*. New Orleans, Louisiana, USA: SIAM, 2013, pp. 1151–1170.

[195] Mehryar Mohri and Ameet Talwalkar. "Can matrix coherence be efficiently and accurately estimated?" In: *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2011, pp. 534–542.

[196] Renato DC Monteiro. "First-and second-order methods for semidefinite programming". In: *Mathematical Programming* 97.1-2 (2003), pp. 209–244.

[197] Kazuhide Nakata et al. "Exploiting sparsity in semidefinite programming via matrix completion II: Implementation and numerical results". In: *Mathematical Programming* 95.2 (2003), pp. 303–327.

[198] Hariharan Narayanan. "Randomized interior point methods for sampling and optimization". In: *Annals of Applied Probability* 26.1 (Feb. 2016), pp. 597–641.

[199] Hariharan Narayanan and Alexander Rakhlin. "Random walk approach to regret minimization". In: *Advances in Neural Information Processing Systems*. 2010, pp. 1777–1785.

[200] Jelani Nelson and Huy L. Nguyen. "OSNAP: Faster Numerical Linear Algebra Algorithms via Sparser Subspace Embeddings". In: *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2013, pp. 117–126.

[201] A. Nemirovski and D. Yudin. *Problem Complexity and Method Efficiency in Optimization*. Wiley Interscience, 1983.

[202] Arkadi Nemirovski. "Efficient Methods in Convex Programming". In: (1994).

[203] Yu Nesterov. "A method of solving a convex programming problem with convergence rate $O(1/k^2)$". In: *Soviet Mathematics Doklady* 27.2 (1983), pp. 372–376.

[204] Yu Nesterov. "Complexity estimates of some cutting plane methods based on the analytic barrier". In: *Mathematical Programming* 69.1-3 (1995), pp. 149–176.

[205] Yu Nesterov. "Efficiency of coordinate descent methods on huge-scale optimization problems". In: *Core discussion papers* 2 (2010), p. 2010.

[206] Yu Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Vol. I. 2003.

[207] Yu Nesterov. "Rounding of convex sets and efficient gradient methods for linear programming problems". In: *Available at SSRN 965658* (2004).

[208] Yu Nesterov. "Smooth minimization of non-smooth functions". In: *Mathematical Programming* 103.1 (2005), pp. 127–152.

[209] Yu E Nesterov and Michael J Todd. "Self-scaled barriers and interior-point methods for convex programming". In: *Mathematics of Operations research* 22.1 (1997), pp. 1–42.

[210] Yu Nesterov and Arkadi Nemirovski. "Conic formulation of a convex programming problem and duality". In: *Optimization Methods and Software* 1.2 (1992), pp. 95–115.

[211] Yu Nesterov and Arkadi Nemirovski. *Interior-Point Polynomial Algorithms in Convex Programming*. Studies in Applied Mathematics. SIAM, 1994.

[212] Yu Nesterov and Arkadi Nemirovski. "Primal Central Paths and Riemannian Distances for Convex Sets". In: *Foundations of Computational Mathematics* 8.5 (2008), pp. 533–560.

[213] Yu Nesterov and Arkadi Nemirovski. *Self-concordant functions and polynomial-time methods in convex programming*. USSR Academy of Sciences, Central Economic & Mathematic Institute, 1989.

[214] Yurii E Nesterov, Michael J Todd, et al. "On the Riemannian geometry defined by self-concordant barriers and interior-point methods". In: *Foundations of Computational Mathematics* 2.4 (2002), pp. 333–361.

[215] James B Orlin. "A faster strongly polynomial minimum cost flow algorithm". In: *Operations research* 41.2 (1993), pp. 338–350.

[216] James B Orlin. "A faster strongly polynomial time algorithm for submodular function minimization". In: *Mathematical Programming* 118.2 (2009), pp. 237–251.

[217] James B Orlin. "Genuinely Polynominal Simplex and Non-Simplex Algorithms for the Minimum Cost Flow Problem". In: (1984).

[218] James B Orlin. "Max flows in O (nm) time, or better". In: *Proceedings of the 45th annual ACM symposium on Symposium on theory of computing*. ACM. 2013, pp. 765–774.

[219] James B Orlin, John VandeVate, et al. "On a" primal" matroid intersection algorithm". In: (1983).

[220] Lawrence M. Ostresh. "On the Convergence of a Class of Iterative Methods for Solving the Weber Location Problem". In: *Operations Research* 26.4 (1978), pp. 597–609.

[221] Onur Ozyesil, Amit Singer, and Ronen Basri. "Stable camera motion estimation using convex programming". In: *SIAM Journal on Imaging Sciences* 8.2 (2015), pp. 1220–1262.

[222] Pablo A. Parrilo and Bernd Sturmfels. "Minimizing Polynomial Functions". In: *DIMACS Workshop on Algorithmic and Quantitative Aspects of Real Algebraic Geometry in Mathematics and Computer Science, March 12-16, 2001, DIMACS Center, Rutgers University, Piscataway, NJ, USA*. 2001, pp. 83–100.

[223] Richard Peng and Daniel A Spielman. "An Efficient Parallel Solver for SDD Linear Systems". In: *arXiv preprint arXiv:1311.3286* (2013).

[224] Frank Plastria and Mohamed Elosmani. "On the convergence of the Weiszfeld algorithm for continuous single facility location allocation problems". English. In: *TOP* 16.2 (2008), pp. 388–406.

[225] Harald Räcke. "Optimal hierarchical decompositions for congestion minimization in networks". In: *Proceedings of the 40th annual ACM symposium on Theory of computing*. STOC '08. Victoria, British Columbia, Canada: ACM, 2008, pp. 255–264.

[226] Srinivasan Ramaswamy and John E Mitchell. "A long step cutting plane algorithm that uses the volumetric barrier". In: *Department of Mathematical Science, RPI, Troy, NY* (1995).

[227] James Renegar. "A polynomial-time algorithm, based on Newton's method, for linear programming". In: *Mathematical Programming* 40.1-3 (1988), pp. 59–93.

[228] Vladimir Rokhlin and Mark Tygert. "A fast randomized algorithm for overdetermined linear least-squares regression". In: *Proc. Natl. Acad. Sci. USA* 1005.36 (2008), pp. 13212–7.

[229] Tamas Sarlos. "Improved approximation algorithms for large matrices via random projections". In: *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2006, pp. 143–152.

[230] M Schmidt. "minFunc: unconstrained differentiable multivariate optimization in Matlab". In: *URL http://www. di. ens. fr/mschmidt/Software/minFunc. html* (2012).

[231] Alexander Schrijver. "A combinatorial algorithm minimizing submodular functions in strongly polynomial time". In: *Journal of Combinatorial Theory, Series B* 80.2 (2000), pp. 346–355.

[232] Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*. Vol. 24. Springer, 2003.

[233] Jonah Sherman. "Nearly Maximum Flows in Nearly Linear Time". In: *Proceedings of the 54th Annual Symposium on Foundations of Computer Science*. 2013.

[234] Maiko Shigeno and Satoru Iwata. "A dual approximation approach to weighted matroid intersection". In: *Operations research letters* 18.3 (1995), pp. 153–156.

[235] Yoel Shkolnisky and Amit Singer. "Viewing direction estimation in cryo-EM using synchronization". In: *SIAM journal on imaging sciences* 5.3 (2012), pp. 1088–1110.

[236] Naum Z Shor. "Cut-off method with space extension in convex programming problems". In: *Cybernetics and systems analysis* 13.1 (1977), pp. 94–96.

[237] Amit Singer and Yoel Shkolnisky. "Three-dimensional structure determination from common lines in cryo-EM by eigenvectors and semidefinite programming". In: *SIAM journal on imaging sciences* 4.2 (2011), pp. 543–572.

[238] Amit Singer and H-T Wu. "Vector diffusion maps and the connection Laplacian". In: *Communications on pure and applied mathematics* 65.8 (2012).

[239] Maurice Sion. "On general minimax theorems". In: *Pacific J. Math* 8.1 (1958), pp. 171–176.

[240] Daniel D. Sleator and Robert Endre Tarjan. "A data structure for dynamic trees". In: *Proceedings of the thirteenth annual ACM symposium on Theory of computing*. STOC '81. Milwaukee, Wisconsin, USA: ACM, 1981, pp. 114–122.

[241] R.L. Smith. "Efficient Monte-Carlo procedures for generating points uniformly distributed over bounded regions". In: *Operations Res.* 32 (1984), pp. 1296–1308.

[242] Daniel A. Spielman and Nikhil Srivastava. "Graph Sparsification by Effective Resistances". In: *SIAM Journal on Computing* 40.6 (2011), pp. 1913–1926.

[243] Daniel A. Spielman and Shang-Hua Teng. "Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems". In: *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*. ACM. 2004, pp. 81–90.

[244] Daniel A. Spielman and Shang-Hua Teng. "Spectral Sparsification of Graphs". In: *CoRR* abs/0808.4134 (2008).

[245] Michel Talagrand. "Embedding subspaces of L1 into L1". In: *Proceedings of the American Mathematical Society* 108.2 (1990), pp. 363–369.

[246] Michel Talagrand. "Embedding Subspaces of Lp in Lp". In: *Geometric Aspects of Functional Analysis*. Springer, 1995, pp. 311–326.

[247] Éva Tardos. "A strongly polynomial minimum cost circulation algorithm". In: *Combinatorica* 5.3 (1985), pp. 247–255.

[248] Michael J Todd. "Semidefinite optimization". In: *Acta Numerica 2001* 10 (2001), pp. 515–560.

[249] Nobuaki Tomizawa and Masao Iri. "ALGORITHM FOR DETERMINING RANK OF A TRIPLE MATRIX PRODUCT AXB WITH APPLICATION TO PROBLEM OF DISCERNING EXISTENCE OF UNIQUE SOLUTION IN A NETWORK". In: *ELECTRONICS & COMMUNICATIONS IN JAPAN* 57.11 (1974), pp. 50–57.

[250] Burt Totaro. "The curvature of a Hessian metric". In: *International Journal of Mathematics* 15.04 (2004), pp. 369–391.

[251] Joel A. Tropp. "User-Friendly Tail Bounds for Sums of Random Matrices". In: *Foundations of Computational Mathematics* 12.4 (2012), pp. 389–434.

[252] Ulrich Trottenberg, Cornelius W Oosterlee, and Anton Schuller. *Multigrid*. Academic press, 2000.

[253] Pravin M. Vaidya. "A New Algorithm for Minimizing Convex Functions over Convex Sets". In: *FOCS*. 1989, pp. 338–343.

[254] Pravin M Vaidya. "An algorithm for linear programming which requires O (((m+ n) n 2+(m+ n) 1.5 n) L) arithmetic operations". In: *Mathematical Programming* 47.1-3 (1990), pp. 175–201.

[255] Pravin M. Vaidya. "Reducing the Parallel Complexity of Certain Linear Programming Problems". In: *FOCS*. 1990, pp. 583–589.

[256] Pravin M. Vaidya. "Solving linear equations with symmetric diagonally dominant matrices by constructing good preconditioners." Unpublished manuscript UIUC 1990. A talk based on the manuscript was presented at the IMA Workshop on Graph Theory and Sparse Matrix Computation, October 1991, Minneapolis. 1990.

[257] Pravin M Vaidya. "Speeding-up linear programming using fast matrix multiplication". In: *Foundations of Computer Science, 1989., 30th Annual Symposium on*. IEEE. 1989, pp. 332–337.

[258] Pravin M Vaidya and David S Atkinson. "A technique for bounding the number of iterations in path following algorithms". In: *Complexity in Numerical Optimization* (1993), pp. 462–489.

[259] Lieven Vandenberghe and Stephen Boyd. "Semidefinite programming". In: *SIAM review* 38.1 (1996), pp. 49–95.

[260] Yehuda Vardi and Cun-Hui Zhang. "The multivariate L1-median and associated data depth". In: *Proceedings of the National Academy of Sciences* 97.4 (2000), pp. 1423–1426.

[261] S. Vempala. "Geometric Random Walks: A Survey". In: *MSRI Combinatorial and Computational Geometry* 52 (2005), pp. 573–612.

[262] Santosh S Vempala. "Recent progress and open problems in algorithmic convex geometry". In: *LIPIcs-Leibniz International Proceedings in Informatics*. Vol. 8. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik. 2010.

[263] Vincenzo Viviani. "De Maximis et Minimis Geometrica Divinatio Liber 2". In: *De Maximis et Minimis Geometrica Divinatio* (1659).

[264] Jens Vygen. "A note on Schrijver's submodular function minimization algorithm". In: *Journal of Combinatorial Theory, Series B* 88.2 (2003), pp. 399–402.

[265] S. Fujishige W. Cui. "A primal algorithm for the submodular flow problem with minimum mean cycle selection". In: *Journal of the Operations Research Society of Japan* (1988).

[266] C Wallacher and Uwe T Zimmermann. "A polynomial cycle canceling algorithm for submodular flows". In: *Mathematical programming* 86.1 (1999), pp. 1–15.

[267] Alfred Weber. *The Theory of the Location of Industries*. Aber den I der Industrien. Chicago University Press, 1909.

[268] E. Weiszfeld. "Sur le point pour lequel la somme des distances de n points donnes est minimum". In: *Tohoku Mathematical Journal* (1937), pp. 355–386.

[269] Virginia Vassilevska Williams. "Multiplying matrices faster than Coppersmith-Winograd". In: *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*. ACM. 2012, pp. 887–898.

[270] Przemyslaw Wojtaszczyk. *Banach spaces for analysts*. Vol. 25. Cambridge University Press, 1996.

[271] Guoliang Xue and Yinyu Ye. "An efficient algorithm for minimizing a sum of Euclidean norms with applications". In: *SIAM Journal on Optimization* 7 (1997), pp. 1017–1036.

[272] Yinyu Ye. "Complexity analysis of the analytic center cutting plane method that uses multiple cuts". In: *Mathematical Programming* 78.1 (1996), pp. 85–104.

[273] Yinyu Ye. *Interior point algorithms: theory and analysis*. Vol. 44. John Wiley & Sons, 2011.

[274] David B Yudin and Arkadii S Nemirovski. "Evaluation of the information complexity of mathematical programming problems". In: *Ekonomika i Matematicheskie Metody* 12 (1976), pp. 128–142.

[275] Zhizhen Zhao and Amit Singer. "Rotationally invariant image representation for viewing direction classification in cryo-EM". In: *Journal of structural biology* 186.1 (2014), pp. 153–166.

[276] U Zimmermann. "Minimization on submodular flows". In: *Discrete Applied Mathematics* 4.4 (1982), pp. 303–323.

[277] Uwe Zimmermann. "Negative circuits for flows and submodular flows". In: *Discrete applied mathematics* 36.2 (1992), pp. 179–189.

[278] Anastasios Zouzias. "A Matrix Hyperbolic Cosine Algorithm and Applications". In: *Proceedings of the 39th International Colloquium on Automata, Languages and Programming (ICALP)*. 2012, pp. 846–858.