Algorithms for Robot Grasp and Delivery

PRASAD CHALASANI* RAJEEV MOTWANI† ANIL RAO‡

Abstract

A major thrust of industrial automation today is towards rapidly configuring assembly lines to manufacture products described by a CAD model using parametrized modular components such as conveyer belts, low degrees of freedom robot arms, modular fixtures, flexible parts feeders, light beams, and simple 2D vision systems. A number of challenging algorithmic problems arise with respect to the efficient design, simulation, and implementation of such assembly lines. We focus on the following specific issue arising in a typical application: parts arrive on a conveyer belt and have to be grasped by a robot arm of finite capacity and then delivered to specified delivery points for packaging or other processing. An intelligent choice of the order of picking up the parts can significantly reduce the total distance traveled by the robot arm and, consequently, its cycle time.

We model this application as problems of the form: Given n identical parts initially located on a conveyer belt, and a robot arm of capacity k parts, compute the shortest route for the robot arm to grasp and deliver the n parts, handling at most k at a time. We consider arbitrary finite k as well as unbounded k. We also allow the belt to be either motionless or moving with a known fixed velocity. For the moving belt problem, we require all parts to be dropped off at a specified origin point, while for the static case we allow up to n different destinations for the parts. In either case the optimization problems that result are NP-hard; the focus is therefore on devising efficient approximation algorithms. While the approaches taken for the static and the moving belt cases are very different, all our algorithms are simple to implement, efficient, and produce a solution within a small constant factor of the optimum.

*Los Alamos National Laboratory, New Mexico. E-mail: chal@lanl.gov.

†Department of Computer Science, Stanford University. E-mail: rajeev@cs.stanford.edu. This work was supported by the ARO MURI Grant DAAH04-96-1-0007. The author is also partially supported by an Alfred P. Sloan Research Fellowship, an IBM Faculty Partnership Award, and NSF Young Investigator Award CCR-9357849, with matching funds from IBM, Schlumberger Foundation, Shell Foundation, and Xerox Corporation.

‡This research was done when the author was with the Department of Computer Science, Utrecht University and supported by European Strategic Program for Research in Information Technology (ESPRIT) Basic Research Action No. 6546 (project PRoMotion). The author is now with Qualcomm, Inc., San Diego. Email: arao@qualcomm.com.

1 Introduction

A major thrust of industrial automation today is towards rapidly configuring assembly lines to manufacture products described by a CAD model using parametrized modular components such as conveyer belts, low degrees of freedom robot arms, modular fixtures, flexible parts feeders, light beams, and simple 2D vision systems. The general scenario of rapidly configuring robots working in their cells to handle parts in an assembly line has been termed *rapid deployment automation* [3, 9, 28]. A number of challenging algorithmic problems arise in the efficient design, simulation, and implementation of such assembly lines.

We focus on the following specific issue related to the planning and scheduling of the motion of a robot arm in a typical industrial application. After manufacture, identical parts are dumped onto a conveyer belt in arbitrary positions. Prior to packaging or assembly, the parts must be collected by a four degree of freedom (x, y, z translation and z rotation) robot arm of capacity k and delivered to an empty pallet at a specified location. Once filled, the pallet is moved away by another conveyor belt, and a new empty pallet appears at the origin. An automated solution involves using an integrated vision system and a robot arm with k suction cups. The vision system identifies the positions of the n parts. The robot arm translates to k part positions, each time positioning an empty suction cup over a part and picking it up. The robot arm then drops off the collection of k parts onto the pallet before returning to pick up more parts. The goal is to plan the motion of the robot arm so that the total time to grasp and deliver the n parts is minimized.

The cost metric assumed in this paper is the total translation of the robot arm. We are neglecting the time for rotating the end-effector in order to correctly reorient itself over the part prior to grasping. This is done for simplicity, as we describe our algorithms for the planar configuration space; taking planar rotation into account would result in a three-dimensional configuration space. Our algorithms can be extended to higher-dimensional configuration spaces, with an exponential dependence on dimension. We also neglect the translation involved between the suction cups. This can be justified by a fast spinning suction cup. Finally, we ignore the vertical (z) distance traveled by the robot as this is fixed for a given n and k. These assumptions allow us to consider the robot arm and the parts to be points in the plane.

In this paper, we study two distinct scenarios: the *static* case where the conveyor belt is held fixed for the duration of the grasps/pick-ups, and the *dynamic* case where the conveyor belt continues to move at a fixed rate throughout. Since the minimization problems that result in either case are easily seen to be NP-hard, we focus on devising efficient approximation algorithms for these problems. Our algorithms return a solution whose cost is within a constant factor of the optimum. (For an instance of a minimization problem whose optimum solution has cost C^*, a k-approximation algorithm must return a solution of cost $C \leq kC^*$ [33].) While the approximation analysis (proving $C \leq kC^*$) is non-trivial, and is indeed the main contribution of this paper, the algorithms themselves are fairly simple and well-suited for practical implementations. The running-times of all our algorithms are bounded by low-degree polynomials in n and, wherever meaningful, in k.

The classical TSP problem requires finding a path of minimum length for a "salesman" who must start at the origin and visit n cities before returning to the origin. The problems we consider can be viewed as the variants of the classical traveling salesman problem (TSP) where the salesman's can visit at most k cities at a time, before returning to the origin. However, for the static case, we consider a generalization where we allow parts to be delivered to up to n different locations, leading to a problem that cannot be viewed as such a variant of TSP. We call this problem the **k-Delivery TSP**, which may be stated formally as: *Given n source points and n sink points in some metric space, with exactly one part placed at each source (all parts are identical), and a robot arm of capacity k, compute a minimum length route for the robot arm to deliver exactly one part to each sink.* The capacity k may be unbounded and the sources and sinks need not lie at distinct locations. The problem is easily seen to be NP-hard via a reduction from TSP: place a source and a sink very close to each point of the TSP problem, and set $k = 1$; now, an optimal solution to the 1-Delivery TSP is an optimal solution to the TSP instance. Similar reductions can be devised for arbitrary finite k as well as for unbounded k.

We provide what appear to be the first known polynomial runtime constant-factor approximation algorithms for the k-Delivery TSP. This problem can also be seen to be an instance of vehicle routing or scheduling problems that have been the subject of intensive study in the literature [11, 21]. The problem of efficiently rearranging parts in the plane with a centrally placed gripper that can rotate and extend telescopically was studied by Atallah and Kosaraju [3] under different cost metrics.

We next consider the case of the moving belt and partially extend our previous results to this case. Formally, we define the following problem. **Dynamic k-Collect TSP**: *Given n point-objects moving in the Euclidean plane with fixed and identical velocities, and a robot arm with capacity k initially located at the origin, find a minimum length route to pick up and deliver these objects to the origin.* Note that the *static* version of this problem is the special case of k-Delivery TSP where the sinks are all at the origin. This problem has been shown to be NP-hard by Lenstra and Rinnooy Kan [30]. This problem also has some features of the *time-dependent* TSP [16, 31] wherein the distance function varies with time.

Although there does not appear to be any prior work on TSP with moving points, heuristics for related problems have been studied in the robotics literature. For example, Li and Latombe [28] consider a situation similar to ours: a vision system identifies parts on a moving belt; these parts must be grasped and shelved elsewhere. They consider heuristics for this on-line problem that plan the motions of two cooperating robots.

Two subtleties arise in the case of moving points leading to some complications. First, we lose symmetry in the distance matrix: given a robot initially at A, the time it requires to travel to B is not the same as the time required to travel to A starting from B initially. Fortunately, however, since the points are all moving at the same velocity, there is some structure in this asymmetry. The second problem is that the distance of points to the origin is time-dependent, although the distances between moving points are time-independent. The dynamic variants have some other interesting and counterintuitive aspects.

Consider a numerical example. Suppose $k = 1$ and there are two points, initially at $(10, 0)$ and $(15, 0)$. The robot is initially at the origin. Assuming that the robot can move at speed 1 and the points move at speed 1/2 in the negative x direction, which point should be visited first? It is easy to check that visiting the *further* point first produces a smaller total time (20) than the other way round (roughly 22). In general, the more time we spend visiting points early in the tour, the closer the later points would have moved to the origin, and so we would spend less time visiting them from the origin.

Thus, the dynamic problem has some features of **Maximum Latency TSP:** *given a set of n points and a symmetric distance matrix (d_{ij}) satisfying the triangle inequality, find a path starting at p_1 and visiting all other points so as to* maximize *the total latency of the points, where the latency of a point is the length of the subpath up to that point in the tour.* We give an approximation algorithm for this problem and also study some variants that arise implicitly in the dynamic settings. While approximation algorithms for *Minimum* Latency TSP are known [6, 20], there does not appear to be any prior work on our version of the problem.

1.1 Organization and Results

In Section 2, we begin by considering 1-Delivery TSP. The 1-Delivery problem is to find an optimal tour that alternately visits the sources and sinks; therefore, we refer to it as the *Bipartite TSP*. This is closely related to the *Swapping Problem* for which Anily and Hassin [2] present a 2.5-approximation algorithm. We use matroid intersection to obtain a 2-approximation algorithm for this problem. In Section 2.1, based on the 1-Delivery approximation algorithm and additional lower bound arguments, we devise a 9.5-approximation algorithm for k-Delivery TSP with arbitrary finite k. Then, in Section 2.2, we present a 2-approximation algorithm for the case of unbounded k.

In Section 3, we give a $1/2$-approximation algorithm for Maximum Latency TSP. We also study some variants that arise implicitly in the dynamic settings discussed in Section 4.

In Sections 4.1–4.3 we establish some basic properties of travel-times involving moving points (such as the triangle inequality) that are not as obvious as they might seem. In the rest of Section 4 we show that an optimal Dynamic 1-Collect tour must visit points in decreasing order of distances to the origin and that the case $k = \infty$ is an asymmetric TSP with bounded asymmetry. We then present a constant factor approximation for the Dynamic k-Collect problem for arbitrary finite k.

Finally, we summarize our work in Section 5. We discuss some problems left open in our work, as well as new directions for algorithmic research in the area of rapid deployment automation.

In the rest of the paper we will refer to the sources as blue points and the sinks as red points. We will also assume for convenience that n is a multiple of k; our results can be extended to the general case by introducing a few dummy sources and sinks close to one of the original points.

1.2 Relation to Previous Work

This paper contributes new results to the field of robot motion and task planning [26, 29, 38]. It can also be seen as contributing to the extensive Operations Research literature on capacitated delivery problems [11, 21]. Most prior research on such problems has focused on heuristics for finding the optimal solution (using sophisticated branch and bound techniques, for example) and not on finding provably good approximations. Examples of such work include the following. **(a)** *The Capacitated Vehicle Routing Problem (CVRP)* [1, 21, 24]: given n points in a metric space and an infinite fleet of vehicles of capacity k, find a collection of vehicle routes starting at origin such that every point is visited by exactly one vehicle. Typical objective functions to be minimized included number of vehicles, total distance traveled, maximum distance traveled by a vehicle, or some combination thereof. **(b)** *The Dial-a-Ride Problem* [5, 25, 35]: compute an optimal route for a k-capacity van to pick-up and drop-off n persons between different origin-destination pairs. Note that this problem differs from the k-Delivery TSP in that each person must be dropped off at a specific destination. **(c)** *The Precedence-Constrained TSP* [5]: for each vertex i there is a set $P(i)$ of vertices that must be visited before visiting i, and we are required to find an optimal TSP satisfying these constraints.

The *static* k-Collect TSP has received considerable attention in the literature and the best-known result is the 2.5-approximation algorithm of Altinkemer and Gavish [1]. Also, the k-Person TSP is a related problem and has a 1.5-approximation algorithm due to Frieze [18] (see also Frederickson, Hecht, and Kim [17]). Another problem with a similar flavor is the k-MST problem and a recent series of papers [4, 7, 19, 32, 36] has led to good approximation algorithms for it.

2 Static Conveyor Belts: The k-Delivery TSP

In this section we consider the case of a static belt and generalize possible part destinations to n in number; the robot arm, which can carry a maximum of k parts per cycle, must deliver one part to each sink. We begin by considering a unit-capacity robot, i.e., the case $k = 1$. When the robot arm has unit capacity, any delivery route must alternate between part pick-up and and part delivery. Viewing the 1-Delivery TSP as a graph problem, we can re-define it as follows:

> **Bipartite Traveling Salesman Problem.** Given an edge-weighted graph G satisfying the triangle inequality, with n blue vertices (sources) and n red vertices (sinks), find the optimal *bipartite* tour starting and ending at a designated blue vertex s and visiting all vertices. A tour is bipartite if every pair of adjacent vertices in it have different colors.

Anily and Hassin [2] have shown a 2.5-approximation algorithm for a generalization of this problem known as the Swapping Problem. Their algorithm finds a perfect matching M consisting of edges that connect red and blue vertices, and uses Christofides's heuristic [10] to find a tour T of the blue vertices. The final delivery route consists of visiting the blue vertices in the sequence specified by the tour T, using the matching edges in M to deliver

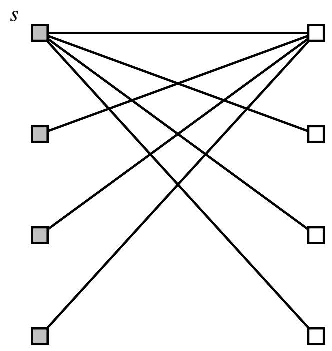

Figure 1: *A bipartite spanning tree for which no depth-first traversal yields a bipartite tour.*

an item to a sink and return to the blue vertex (or "short-cut" to the next blue vertex on T). If OPT is the optimal delivery tour, clearly $T \leq 1.5\text{OPT}$ and $M \leq 0.5\text{OPT}$, whereas the total length of the delivery tour is at most $T + 2M \leq 2.5\text{OPT}$. We exploit some combinatorial properties of bipartite spanning trees and matroid intersection to improve this factor to 2.

A naive approach towards a 2-approximation is to mimic the well-known 2-approximation algorithm for the TSP problem: pick a *bipartite* spanning tree of G, perform a depth-first traversal followed by short-cutting. A spanning tree of G is bipartite if each edge connects a red and blue vertex. Given a bipartite spanning tree T, we can think of it as a tree rooted at s, and do a depth-first traversal of T with short-cuts (there may in general be several ways to short-cut) and obtain a tour of G. However, such a tour may not be bipartite; there are bipartite spanning trees that do not yield a bipartite tour regardless of how we perform the depth-first traversal and short-cuts (see Fig. 1).

Our 2-approximation algorithm is based on the following very simple observations.

> **Observations.** Let T be a bipartite spanning tree where each blue vertex has degree at most 2; then, T has exactly one blue vertex v_1 of degree 1. If T is rooted at v_1 then every blue vertex has exactly one (red) child. Clearly, if we traverse this rooted tree T in (any) depth-first order, then the sequence of vertices visited are of alternating color.

Clearly, the OPT bipartite tour contains a bipartite spanning tree where all blue vertices have degree at most 2. Therefore the weight of the minimum-weight bipartite spanning tree whose blue vertices have degree at most 2, is a lower bound on OPT. So if we can find (in polynomial time) the minimum-weight bipartite spanning tree T whose blue vertices have degree at most 2, then a depth first traversal of T (with short-cuts) will yield a tour whose length is at most twice OPT.

We now claim that the problem of finding T can be viewed as that of finding the minimum-weight, maximum-cardinality subset in the *intersection* of two matroids [27, 12, 13]. The two matroids in this case are: M_1, the matroid of all bipartite forests, and M_2, the matroid of all bipartite subgraphs whose blue vertices have degree at most 2. For completeness we review here the definition of a matroid, following the standard text [27].

Definition 1 *A* matroid $M = (E, \mathcal{I})$ *is a structure in which* E *a finite set of elements and* $\mathcal{I}$ *is a family of subsets (called* independent sets*) of* E*, such that:* $\phi \in \mathcal{I}$ *and all proper subsets of a set* $I \in \mathcal{I}$ *are in* $\mathcal{I}$*; and, if* I_p *and* I_{p+1} *are sets in* $\mathcal{I}$ *containing* p *and* $p+1$ *elements respectively, then there exists an element* $e \in I_{p+1}$ *such that* $I_p \cup \{e\} \in \mathcal{I}$*.*

An example of a matroid is the *graphic matroid* $M = (E, \mathcal{I})$ where E is the set of edges of an undirected graph, and a subset $I \subset E$ is in $\mathcal{I}$ if and only if I is cycle-free. Another example of a matroid is the *matrix matroid* $M = (C, \mathcal{I})$ where C is the set of columns of a fixed matrix A and a subset S of columns is in $\mathcal{I}$ if and only if the columns of S are linearly independent. A maximal-cardinality independent subset of a matroid is called a *base* of a matroid; all bases of a matroid have the same cardinality.

Returning to our problem, let E be the set of all edges that connect red vertices to blue vertices. Let $\mathcal{F}$ denote the collection of all subsets of E that are cycle-free, and let $\mathcal{D}$ denote the collection of subsets S of E such that no more than two edges of S are incident on any blue vertex. Then it is easily seen that $M_1 = (E, \mathcal{F})$ and $M_2 = (E, \mathcal{D})$ are matroids. In addition, the problem of finding *a minimum-weight bipartite spanning tree where the blue vertices have degree at most two,* is equivalent to the problem of finding *a minimum-weight common base of* M_1 *and* M_2*.*

This is a special case of the *matroid intersection* problem, which was first solved in polynomial time by Edmonds [12, 13]. Other authors [8] have exploited the special structure of problems such as ours to improve the running times. We obtain the following theorem.

Theorem 1 *There is a polynomial time 2-approximation algorithm for the Bipartite TSP.*

2.1 Finite Capacity Robot Arms

We will now show how to obtain a constant-factor approximation for the case of arbitrary finite k using the algorithm for the unit capacity case. But first, we will establish some lower bounds on the optimal solution. Let C_k denote the (length of the) optimal k-Delivery tour, and let C_r and C_b denote the (length of the) optimal tours on the red and blue points respectively. Let A denote the weight of the minimum-weight perfect matching in the bipartite graph with red vertices on one side and blue vertices on the other. To keep the notation simple, we will often use the same symbol to denote a graph and its weight, and the context will make it clear which one is intended.

Lemma 2 **(a)** $A \leq C_1/2$. **(b)** $\frac{1}{2k}C_1 \leq C_k$.

Proof: Part (a) is easy to see since C_1 consists of two perfect matchings and the one with smaller weight is at least as heavy as A. To see part (b), start with an optimal k-Delivery tour C_k: this defines an ordering $r_1, r_2, \ldots, r_n$ on the red points and an ordering $b_1, b_2, \ldots, b_n$ on the blue points. We then construct a 1-Delivery tour T starting at the blue vertex b_1 as follows: Consider the blue vertices in the order imposed by C_k, connecting the ith blue vertex b_i to the earliest red vertex in the C_k-ordering that has not already been connected to a blue vertex; then add another edge connecting this red vertex to the next blue vertex b_{i+1} (if this red vertex is the last one, connect it to the starting blue vertex b_1). By the triangle inequality, each edge e of T is no longer than the sum of the C_k-edges connecting the endpoints of e; we can thus "charge off" each edge of T to a collection of edges of C_k. Since there is never a sequence of more than k consecutive red or consecutive blue points in the tour C_k, it follows that no edge of T is charged more than $2k$ times. Thus $T \leq 2kC_k$, from which the lemma follows since $C_1 \leq T$. ■

We omit the proof of the following lemma.

Lemma 3 **(a)** $C_r \leq C_k$. **(b)** $C_b \leq C_k$.

We now use the lower bounds just presented to design a constant-factor approximation algorithm for the k-Delivery problem. We first use Christofides' heuristic to obtain a 1.5-approximate tour T_r of the red vertices and a 1.5-approximate tour T_b of the blue vertices. Next we (arbitrarily) break up T_r and T_b into paths of k vertices each, by deleting edges appropriately. It will be convenient to view each k-path as a "super-node" in the following. We now overlay the minimum-weight perfect matching of cost A on this graph. Note that any (red or blue) super-node now has degree exactly k, and that there may be several edges between two given super-nodes. Thus, we obtain a k-regular bipartite multi-graph. The following lemma is crucial to the design of our algorithm.

Lemma 4 *The edges of a d-regular bipartite multi-graph can be partitioned into d perfect matchings.*

Proof: First, note that the d-regularity implies there are an equal number of vertices on each side of the bipartition. Consider any subset S of the vertices on the left side of the bipartite graph and say $|S| = m$. Clearly the number of edges emanating from S is dm and by the Pigeonhole Principle if the set S has fewer than m neighbors on the right side then some vertex on the right side has degree greater than d, which is a contradiction. Thus any subset S of the left-vertices has at least $|S|$ neighbors. Clearly, this remains true even if replace each multi-edge by one of the edges. Then, by Hall's theorem [22, 27] for the existence of perfect matchings, it follows that the multigraph contains a perfect matching. If we delete this perfect matching, we are left with a $(d-1)$-regular bipartite multigraph, and the same argument can be repeated, each time removing a perfect matching. This proves the lemma. ■

We can thus partition the perfect matching A into k perfect matchings on the super-nodes. We pick the least-weight matching M out of these and delete all other edges of T_1. Clearly $M \leq A/k \leq \frac{1}{2k}C_1$. At this stage we have a collection of n/k subgraphs $H_1, H_2, \ldots, H_{n/k}$, each consisting of a red supernode connected via an edge of M to a blue supernode. Now we re-introduce the edges of T_b that were removed when breaking T_b into k-paths; this imposes a cyclic ordering on the subgraphs H_i; let us relabel them $H_1, H_2, \ldots, H_{n/k}$ with this cyclic ordering, where H_1 contains the start blue vertex. We now traverse the subgraphs $H_1, H_2, \ldots, H_{n/k}$ in sequence as follows. Within each subgraph H_i first visit all the blue vertices, then use the edge of M to go to the red side and visit all the red vertices, and then return to the blue side, and go to the blue vertex that is connected via an edge e of T_b to the next subgraph H_{i+1}, and use the edge e to go to H_{i+1} (or H_1 if $i = n/k$).

We claim that this tour T is within a constant factor of the optimal k-Delivery tour. To verify this, notice that in short-cutting, by triangle inequality, we "charge" each edge of T_b no more than 3 times, each edge of T_r no more than 2 times and each edge of M at most 2 times. Thus, we obtain that

$$
\begin{aligned}
T &\leq 3T_b + 2T_r + 2M \\
&\leq 3 \times 1.5C_b \ + \ 2 \times 1.5C_r + \frac{2}{2k}C_1 \\
&\leq 9.5C_k.
\end{aligned}
$$

Theorem 5 *The above algorithm gives a 9.5-approximation to the optimal k-Delivery TSP.*

2.2 Unbounded Capacity Robot Arms

When the robot arm has unbounded capacity, the only restriction is that at any stage during the delivery, the number of items picked up so far be at least as many as the number of sinks visited so far. The corresponding graph problem is defined as follows:

> **Blue-Dominant TSP.** Given an edge-weighted graph G satisfying the triangle inequality, with n blue vertices (sources) and n red vertices find an optimal tour

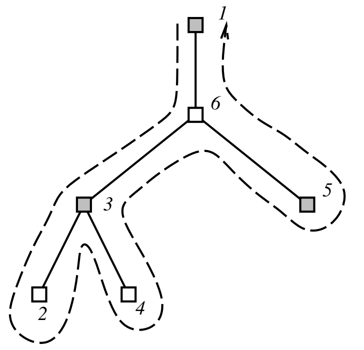

Figure 2: *A bipartite spanning tree that does not contain a perfect matching and yet there is a depth-first traversal that can be short-cut to yield a bipartite tour.*

> that starts at a blue vertex s and visits all the vertices (and returns to s) in *blue-dominant* order. An ordering of vertices is blue-dominant if in each prefix of the ordering, there are at least as many blue vertices as red vertices.

We show below a 2-approximation algorithm for this problem. It turns out that one can obtain a 2-approximation for this problem by first finding a minimum-weight spanning tree (not necessarily bipartite) and then using a *specific* depth first traversal and short-cutting scheme. As Figure 2 shows, for a fixed depth-first traversal there may be several ways to perform short-cuts.

Let us make precise the distinction between a depth-first traversal and a specific short-cutting scheme. Note that we can view a depth-first traversal as traversing edges rather than as visiting vertices. In the edge-traversing viewpoint, each edge is traversed exactly twice: first downward (away from the root) and later upward (toward the root). Thus each vertex may be "visited" several times by a depth-first traversal. The short-cutting scheme *marks* exactly one out of these several visits to a vertex. For a fixed depth-first traversal and short-cutting scheme, the marking order defines a tour of all the vertices. We will consider two types of short-cutting rules in particular: when the depth-first traversal reaches a subtree rooted at v, we may mark v either *before* or *after* visiting all its descendants. In the former case we say v is *pre-marked*, and in the latter case we say v is *post-marked*.

Theorem 6 *For any spanning tree T rooted at the blue vertex s, there is a depth-first traversal and a short-cutting scheme that marks the vertices in blue-dominant order.*

Proof: For any vertex v, let $\mathbf{r}(v)$ and $\mathbf{b}(v)$ be defined as before. For a marking sequence defined by a depth-first traversal and a short-cutting scheme, for any vertex v let $R(v)$ denote the number of red vertices marked just before the depth-first traversal *first* reaches v, and let $B(v)$

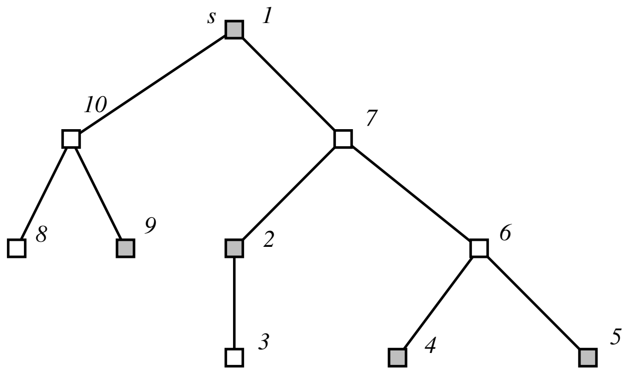

Figure 3: *When the vertices of this spanning tree are marked in depth-first order using the short-cutting scheme of Theorem 6, we get a valid delivery tour for an unbounded-capacity robot arm. The sources are shaded.*

denote the number of blue vertices marked just before the depth-first traversal *first* reaches v.

Consider a depth-first traversal of T satisfying the following:

> **Depth First Traversal Rule.** After visiting a vertex v, if there are any children w of v such that $\mathbf{b}(w) \geq \mathbf{r}(w)$ then visit them (and their descendants) before visiting any remaining children.

The short-cutting scheme is:

> **Short-Cutting Rule.** Pre-mark blue-vertices (sources) and post-mark red vertices.

Figure 3 illustrates an application of these rules.

We would like to show that $B(v) \geq R(v)$ holds at all times. We will in fact claim that the following invariant holds whenever the depth-first traversal first reaches a vertex v:

$$B(v) \geq R(v), \quad \text{and}$$
$$B(v) + \mathbf{b}(v) \geq R(v) + \mathbf{r}(v).$$

The proof of the invariant is by structural induction on the rooted tree T. It is trivially true at the start vertex s since $B(s) = R(s) = 0$ and $\mathbf{b}(s) = \mathbf{r}(s) = n$. Assume inductively that the invariant holds just before the depth-first traversal reaches some vertex v. We argue that it will hold at each child of v, visited say in the order $w_1, w_2, \ldots, w_k$. For any vertex v, let $c(v)$ denote the quantity $\mathbf{r}(v) - \mathbf{b}(v)$, the "surplus of sinks", and let $D(v)$ denote $B(v) - R(v)$, the "source surplus". Our inductive assumption $D(v) \geq c(v)$ implies, if v is blue,

$$D(v) \geq \sum_{i=1}^{k} c(w_i) - 1,$$

and if v is red,

$$D(v) \geq \sum_{i=1}^{k} c(w_i) + 1.$$

If v is blue, it is marked before the depth-first traversal visits w_1, so

$D(w_1) = D(v) + 1$. If v is red, it is not marked before visiting w_1, so $D(w_1) = D(v)$. In either case we have $D(w_1) \geq 0$ and

$$D(w_1) \geq \sum_{i=1}^{k} c(w_i).$$

Note that for each $i = 1, 2, \ldots, k-1$,

$$\begin{aligned} D(w_{i+1}) &= D(w_i) - c(w_i) \\ &\geq c(w_{i+1}) + c(w_{i+2}) + \ldots + c(w_k). \end{aligned}$$

Our depth-first traversal rule specifies that children with $c(w_i) \leq 0$ must be visited first, and the above equation implies that the $D(.)$ value does not decrease after traversing the subtree rooted at such a child and therefore remains positive. So our invariant holds at each such child. After the subtree under the last such child (say w_j) has been traversed, we have $D(w_{j+1}) \geq 0$ and

$$D(w_{j+1}) \geq c(w_{j+1}) + c(w_{j+2}) + \ldots + c(w_k),$$

where each term on the right hand side is positive. It is now easy to see that $D(w_i) \geq 0$ and $D(w_i) \geq c(w_i)$ will hold for each $i = j+1, \ldots, k$. ■

3 Maximum Latency TSP

This section is concerned with the latency measure for TSP tours, a minor digression from the family of robot grasp and delivery problems under consideration. A variant of the following latency TSP problem arises in the moving points case and is also of independent interest. Given a set of n points $\{p_1, \ldots, p_n\}$ in a metric space, find a path visiting all points, starting at a given point p_0, such that the total *latency* of the points is *maximized*. If in a given path P the length of the ith edge traversed is e_i then the latency of the jth point visited ($j > 0$) is $L_j = \sum_{i=1}^{j} e_i$ and the total latency $L(P)$ is

$$L(P) = \sum_{j=1}^{n} L_j = \sum_{i=1}^{n} (n - i + 1) e_i.$$

We would like to find a path P for which $L(P)$ is maximized. We can show that this problem is NP-hard, by reducing from the Maximum Hamiltonian Path (MaxHP) problem. A related problem, *minimum* latency TSP, has been addressed earlier [6, 20] and constant-factor approximation algorithms are known.

We show that the greedy strategy, "*At any stage, visit the farthest unvisited point from the current point,*" achieves a total latency at least half of the maximum latency path. In fact, the greedy path also has *length* at least half of the MaxHP from the starting point p_0. The key observation is the following lemma.

Lemma 7 *Let G_i be the length of the first i edges in the greedy path starting from p_0. Let P_i be the length of the maximum i-path, i.e., the longest path that visits i vertices from p_0. Then for $i \leq n$, $G_i \geq P_i/2$.*

Proof: For brevity, paths/edges and their lengths are denoted by the same symbols. Consider the maximum matching M in the maximum i-path P_i, i.e., the maximum-length collection M of independent edges from P_i. Let the (lengths of) edges of M be $m_1 \geq m_2 \geq \ldots \geq m_k$. Note that $P_i \leq 2(m_1 + m_2 + \ldots + m_k)$. We will argue that $G_i \geq (m_1 + m_2 + \ldots + m_k)$.

Consider the edges in G_i as being *directed* in the direction of travel, starting from p_0. Call an edge m of M an *anchor* if there is an edge of G_i that *starts* at an endpoint of m; the *earliest* such edge g of G_i is said to be *anchored* at m.

Claim (a): If g is anchored at m then $g \geq m$, since G_i is greedy and the other endpoint of m has not yet been visited at the time g was traversed by the greedy path.

Claim (b): If $m \in M$ is not an anchor, then *every* edge of G_i has length at least $m/2$. To see this, note that neither end-point of m is visited by G_i (except possibly at the end). By the triangle inequality, from any point p, at least one of the end-points of m is at distance at least $m/2$. Since G_i is greedy, its edges have length at least $m/2$.

Now suppose $m_1, m_2, \ldots, m_u$ are anchors, and m_{u+1} is not, i.e., m_{u+1} is the heaviest edge of M that is *not* an anchor. (Note that u could be 0.) Let $g_1, g_2, \ldots, g_u$ be the corresponding anchored edges of G_i. Then Claims (a) and (b) imply that $g_1 + g_2 + \ldots + g_u \geq m_1 + m_2 + \ldots + m_u$, and that every edge of G_i has length at least $m_{u+1}/2$. There are now two cases to be considered. Recall $k = |M|$.

Case 1. [$u = 0$ and $i = 2k - 1$] If $u = 0$ then m_1 is not an anchor, so all edges of G_i have length at least $m_1/2$. Also $i = 2k - 1$ implies that the starting point p_0 must occur in the matching M. Thus the first edge of G_i must be anchored at an edge m' of M, and the total length of G_i is at least

$$\begin{aligned} m' + (i-1)m_1/2 &\geq m' + (2k-2)m_1/2 \\ &= m' + (k-1)m_1 \\ &\geq m_1 + m_2 + \ldots + m_k. \end{aligned}$$

Case 2. [$u \geq 1$ or $i \geq 2k$] Note that the total length L of unanchored edges of G_i is at least $(i - u)m_{u+1}/2$. If $u \geq 1$,

$$L \geq (2k - 1 - u - (u - 1))m_{u+1}/2 = (k - u)m_{u+1},$$

and if $i \geq 2k$,

$$L \geq (2k - u - u)m_{u+1}/2 = (k - u)m_{u+1}.$$

Thus, G_i is at least $(m_1+m_2+...+m_u)+(k-u)m_{u+1} \geq m_1 + m_2 + ... + m_k$. ■

Clearly the latency of the ith point in the maximum latency path is at most P_i, so the lemma implies that the total latency of the greedy path is at least half that of the maximum latency path. Since P_n is the MaxHP, it also follows that the greedy path is a $\frac{1}{2}$-approximation of the MaxHP. Lemma 7 implies our result.

Theorem 8 *The greedy strategy of always visiting the furthest unvisited point achieves a total latency at least half that of the maximum latency path. The total length of the greedy path is at least half that of the maximum length path that visits all points exactly once (i.e., the MaxHP).*

We can also show that the greedy heuristic works well for a different "latency measure" that arises implicitly in Dynamic k-Collect TSP.

Theorem 9 *Let $\alpha < 1$ be a positive constant. For a Hamiltonian path P starting at p_0, define the cost $L_\alpha(P) = \sum_{i=1}^{n-1}(1 - \alpha^{n-i})e_i$. The greedy heuristic produces a path whose L_α cost is at least $(1 - \alpha)/2$ of the maximum-L_α path.*

Proof: Let G denote the greedy path, and H^* denote the Maximum Hamiltonian Path. Let P^*_α denote the path that maximizes $L_\alpha(.)$ and P^* be the path that maximizes $L(.)$. Since $(1 - \alpha^{n-i}) < 1$ for $i = 1, 2, \ldots, n-1$, we have

$$L_\alpha(P^*_\alpha) \leq length(P^*_\alpha) \leq length(H^*),$$

and since $(1 - \alpha^{n-i}) \geq 1 - \alpha$ for $i = 1, 2, \ldots, n-1$, we have

$$L_\alpha(G) \geq (1 - \alpha)length(G).$$

The desired result follows then follows from the fact that $length(G) \geq length(H^*)/2$. ■

4 Moving Conveyor Belts: Dynamic k-Collect TSP

In this section, we consider the case of parts on a moving conveyer belt. For convenience, we switch to the L_1 metric to measure costs. That is, we assume that the robot only translates parallel to the x and y axes. While there are situations where this applies directly, it is also easy to see that this only causes an error of factor $\sqrt{2}$ with respect to the L_2 metric. We assume that the points $p_1, p_2, \ldots, p_n$

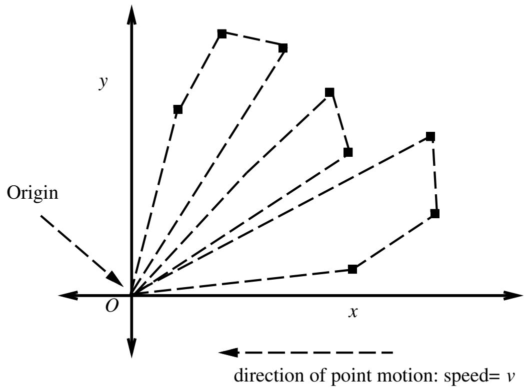

Figure 4: *Illustrating the moving points model. A* 3*-collect tour is shown.*

are always within the positive quadrant of the coordinate frame centered at the origin $p_0 = (0, 0)$ (see Figure 4). The robot moves with speed 1, and the belt (and each point p_i) moves with a velocity v directed in the negative-x direction. The y-axis represents the end of the conveyor belt, and to obtain meaningful results we must assume at the very least that v is suitably bounded below 1, since a slow robot may be unable to catch up with some points. In fact, in our application it turns out it is sufficient to have $v \leq \frac{k}{2n}$ to ensure that no p_i crosses the y-axis while the robot is in the process of executing the tour. We will assume this upper bound on v throughout the rest of this paper. Also, we define $\alpha = \frac{1-v}{1+v}$.

For clarity, we refer to a fixed point in space (such as the origin) as a *space-point*, to distinguish it from a *moving-point* p_i. Define (x_i, y_i) as the coordinates of the point p_i at time 0, and let $d_i = x_i + y_i$ denote the L_1 distance of p_i from the origin at time 0. Clearly the x-coordinate of p_i at time t is $x_i - vt$ and the y-coordinate doesn't change. The distance of p_i from the origin at time t is therefore $x_i + y_i - vt = d_i - vt$.

4.1 Shortest Paths and Triangle Inequality

We establish several basic properties of travel-times that are not obvious as they may seem. Henceforth, when we say the robot moves from a point A to a space-point B or meets a moving point p_i, we will assume that the robot takes the shortest-time path. What is the quickest way for a robot to meet a moving-point p? The following lemma characterizes such paths. A robot path is said to be *monotone* if no two points on the path have the same x-coordinate or the same y-coordinate.

Lemma 10 (Shortest Paths) *Suppose the robot is at a space-point A and meets moving-point p at the earliest possible time, say at space-point B. Then the robot's path from A to B is necessarily a shortest path between those* *points, and the robot never stops at any time before it meets p.*

Proof: At any time the robot may either "sit and wait," or move parallel to the x-axis, or move parallel to the y-axis. Suppose the robot meets point p after time T. If the robot's path to B is not the shortest path to B (from A) or the robot stopped at some time, this means that the robot could have arrived at B at an earlier time $T' < T$ (by either using a shorter path to B or not waiting along the way). At this time T' the moving-point p must be to the right of B a distance $(T - T')v$ away on the same y-coordinate, and the robot can meet p in time $(T - T')\frac{v}{1+v}$ by moving toward it. Thus, the total time to meet p could have been $T' + \frac{v}{1+v}(T - T')$, and the time saved would be $(T - T') - \frac{v}{1+v}(T - T') = \frac{1}{1+v}(T - T')$, which is positive. This contradicts our assumption that the robot met p at the earliest possible time. ■

We have the following important corollaries of the above lemma.

Corollary 11 (Monotone Path) *Suppose the robot moves from a space-point A to meet a moving point p. Then, the robot's path must be monotone. In particular a quickest way for the robot to meet p is to first move to the y-coordinate of p, then move toward p.*

Proof: The following fact will be useful in this and other proofs about shortest-paths:

Fact 2 *The shortest (and therefore least travel-time) path for the robot to move from a given space point A to a given space point B is a monotone path; in particular the monotone path that first moves to the y-coordinate of B and then to the x-coordinate of B is optimal.*

The monotonicity follows from Lemma 10 and Fact 2. Any monotone path that meets p can be replaced by one where all the x-motion is made after the y-motion, without changing the rendezvous time or coordinates. Clearly the y-motion consists simply of moving to the y-coordinate of p. At this time p may either be left or right of the robot. In case p is left of the robot, by monotonicity the quickest way to meet p is to move toward p. In case p is to the right of the robot, if the robot moves to the left then either the path becomes non-monotone before it meets p or it must stop and wait for p to catch up with it, both of which are not possible in a shortest path, by Lemma 10. Thus, in this case also the robot must move toward p. ■

Corollary 12 (Triangle Inequality 1) *If the moving-points p_i and p_j are in the positive quadrant, and the robot is at p_i, then the time t_{ij} to travel directly to p_j is bounded by the time to travel to p_j via the origin.*

Proof: If the composition of the robot's path from p_i to the origin and the path from the origin to p_j is not monotone, then by the Monotone Path Corollary 11 it cannot be shorter than the shortest p_i-to-p_j path; if it is monotone then it cannot be shorter than the particular shortest p_i-p_j path described in Lemma 11. ■

Corollary 13 (Triangle Inequality 2) *Let t_{ij} denote the shortest time of travel from moving-point p_i to moving-point p_j. For any three points p_i, p_j, p_k, $t_{ij} + t_{jk} \geq t_{ik}$*

It is natural to wonder whether a given order of visiting the points takes less time when the points are moving than when they are at rest. We show that this is indeed the case (recall that $v \leq \frac{k}{2n}$).

Theorem 14 *Consider a tour T that starts at the origin, visits some set of moving-points, and returns to the origin, and let $T(v)$ be the time taken to do so. Then, for $v \geq 0$, $T(v) \leq T(0)$.*

Proof: Denote the origin by $p_0 = (x_0, y_0) = (0, 0)$, and suppose T visits the points in the sequence $p_1, p_2, \ldots, p_m$ and returns to the origin. For $i = 0, 1, 2, \ldots, m - 1$, let $X_i = |x_i - x_{i+1}|$ and $Y_i = |y_i - y_{i+1}|$. Clearly then

$$T(0) = \sum_{i=0}^{m-1} (X_i + Y_i) \ + \ d_m.$$

For a positive speed v, from equation (2), the total time is given by

$$T(v) = (1 - v) \sum_{i=0}^{m-1} T_i \ + \ d_m,$$

where T_i is the time taken by T to go from p_i to p_{i+1}, and is given by one of the expressions (4), (5), (6). For each possible expression for T_i, it is easy to check that $(1 - v)T_i \leq (X_i + Y_i)$, so $T(v) \leq T(0)$. ■

4.2 Useful Time Expressions

The following time expressions are easily verified; we will use them in the proofs in this paper. In all of the following keep in mind that d_i denotes the distance of p_i from the origin at time 0.

- **Origin to Moving-Point**. The robot is at the origin at time τ, and meets moving point p_i. The earliest rendezvous time is

$$\tau + \frac{d_i - v\tau}{1 + v} = \frac{\tau + d_i}{1 + v}. \qquad (1)$$

- **Moving-point to Origin**. At time τ the robot is at a moving-point p_i and it then returns to the origin. The time of return to the origin is

$$\tau + d_i - v\tau = d_i + (1-v)\tau. \qquad (2)$$

- **Moving-Point to Origin to Moving Point** At time τ the robot is at a moving-point p_i; it then returns to the origin and goes to moving-point p_j. From the previous expression, the time of arrival at the origin is $d_i + (1-v)\tau$, and during this time the point p_j moves closer to the origin by a distance $v(d_i + \tau(1-v))$. So the time of arrival at p_j is

$$\begin{aligned} d_i \quad &+ \quad \tau(1-v) + \frac{d_j - v[d_i + \tau(1-v)]}{1+v} \\ &= \quad \frac{d_i + d_j}{1+v} + \tau\frac{1-v}{1+v}. \end{aligned} \qquad (3)$$

4.3 Optimal Rendezvous

We can use the above characterization to derive some expressions for the shortest time (or distance) to meet a moving point. Suppose at time t_0 the robot is at space-point $A = (x_1, y_1)$, and moving-point p is at (x_2, y_2). Let $x = |x_1 - x_2|$ and $y = |y_1 - y_2|$. If the robot moves and meets moving-point p at the earliest possible time $t_0 + t$, then t equals one of the following. In view of Lemma 11 we may assume that the robot first moves to the y-coordinate of p, then meets p by moving parallel to the x-axis.

- If p is right of A at time t_0, and $vy \leq x$, then p remains on the right of the robot when it reaches the y-coordinate of p, so

$$t = y + \frac{x - vy}{1+v} = \frac{x+y}{1+v}. \qquad (4)$$

- If p is right of A at time t_0 and $vy > x$, the moving-point p will be left of the robot by the time the robot reaches the y-coordinate of p, so

$$t = y + \frac{vy - x}{1-v} = \frac{y-x}{1-v}. \qquad (5)$$

- If p is to the left of A at time t_0, then

$$t = y + \frac{vy + x}{1-v} = \frac{x+y}{1-v}. \qquad (6)$$

4.4 Robot Arms with Capacity 1

In the remainder of the paper we write C_k to denote the (length of the) optimal Dynamic k-Collect tour.

In Dynamic 1-Collect TSP, the robot must visit the points one at a time, returning to the origin after visiting each point. Suppose that the robot visits the points $p_1, \ldots, p_n$ in that order. We now derive an expression for the total time $T^{(1)}$ taken by a 1-Collect tour. Let T_m denote the time taken by the tour after it has visited m points and returned to the origin. We proceed by induction on m. The base case, $m = 1$, follows from an application of Equation 1 using $\tau = 0$ and $i = 1$; note that the factor of 2 comes from the requirement that the robot returns to the origin.

For the induction step, we assume that

$$T_{m-1} = \frac{2}{1+v}\sum_{j=1}^{m-1} \alpha^{m-(i+1)} d_i.$$

Suppose that at time T_{m-1} the robot has just returned to the origin after visiting the point p_{m-1}, and now it is ready to go visit point p_m. The time taken to reach p_m is given by an application of Equation 1 with $\tau = T_{m-1}$ and $i = m$, and this equals

$$\frac{d_m}{1+v} - \frac{vT_{m-1}}{1+v}.$$

The total time required is given by

$$\begin{aligned} T_m \quad &= \quad T_{m-1} + 2\left(\frac{d_m}{1+v} - \frac{vT_{m-1}}{1+v}\right) \\ &= \quad \left(1 - \frac{2v}{1+v}\right) T_{m-1} + \left(\frac{2}{1+v} d_m\right) \\ &= \quad \alpha T_{m-1} + \frac{2}{1+v} d_m \\ &= \quad \frac{2}{1+v}\left(\sum_{j=1}^{m-1} \alpha^{m-(i+1)+1} d_i\right) + \frac{2}{1+v} d_m \\ &= \quad \frac{2}{1+v}\left(\sum_{j=1}^{m} \alpha^{m-i} d_i\right). \end{aligned}$$

So the total time $T^{(1)}$ taken by a 1-Collect tour is given by

$$T^{(1)} = \frac{2}{1+v}\sum_{j=1}^{n} \alpha^{n-i} d_i. \qquad (7)$$

Consequently, we can easily obtain the following result:

Theorem 15 *For Dynamic 1-Collect TSP, the minimum-time tour is one that visits moving-points in decreasing order of their distance from the origin at time 0.*

Thus the optimal Dynamic 1-Collect tour has some aspects of a *Maximum Latency* tour. In fact the maximum latency problem is implicit in the Dynamic k-Collect problem for arbitrary finite k. We make this formal in Section 4.7 where we introduce the notion of *Geometric Latency* of a tour.

4.5 Robot Arms with Unbounded Capacity

In Dynamic k-Collect TSP with $k = \infty$, the robot must visit all n moving-points before returning to the origin. For the optimal tour suppose the last moving-point visited is p, and say p is at distance d from the origin at time 0. If the tour reaches p at time T_p, then by (2) the total time taken is $C_\infty = d + (1-v)T_p$.

Our strategy for approximating C_∞ is to "guess" the last point p visited by the optimal tour (there are only n possibilities for p) and approximate the minimum-length path T_p^* from the origin to p that visits every other moving-point before visiting p (i.e., the Minimum Hamiltonian Path from the origin to p).

The problem of approximating T_p^* can be set up as an Asymmetric TSP instance on a graph with $n+1$ vertices $v_0, v_1, \ldots, v_n$, where v_0 represents the origin and v_i the moving-point p_i. The directed distance $d(v_i, v_j)$ is defined to be the appropriate one out of the expressions (4), (5), and (6). Note that the ratio of the two directed distances between a given pair of vertices is bounded by either $(1+v)/(1-v)$ or $(y+x)/(y-x)$, where in the second case $vy > x$, i.e., $x/y < v < 1$. Thus the ratio never exceeds $1/\alpha$. The best-known approximation algorithm [23] for the Minimum Hamiltonian Path between two specified vertices for a *symmetric* distance matrix has a ratio $\frac{5}{3}$. Therefore, using this algorithm we can approximate T_p^* to within a factor $\frac{5}{3\alpha}$, and thereby approximate C_∞ to within the same ratio.

Theorem 16 *There is a $\frac{5}{3\alpha}$-approximation algorithm for Dynamic k-Collect TSP with $k = \infty$.*

4.6 Robot Arms with Finite Capacity k

We now consider Dynamic k-Collect TSP. We derive an expression for the time $T^{(k)}$ taken by a k-Collect tour, Let $m = n/k$ denote the number of returns to the origin. The sequence of edges traversed can be viewed as

$$\begin{array}{lll} (p_0, p_{v(0)}) & L_1 & (p_{u(1)}, p_0), (p_0, p_{v(1)}) \\ & L_2 & (p_{u(2)}, p_0), (p_0, p_{v(2)}) \\ & L_3 & \ldots \quad L_m \quad (p_{u(m)}, p_0), \end{array}$$

where each L_i denotes the sequence of edges involving non-origin points in the ith excursion from the origin. We abuse notation and denote by L_i the time spent traversing the corresponding edges. Let $D_0 = d_{v(0)}$, $D_m = d_{u(m)}$, and for $2 \le i \le m-1$, $D_i = d_{u(i)} + d_{v(i)}$.

Notice first that the time till the end of L_1 is $T_1 = D_0/(1+v) + L_1$ and the time to the end of L_2 (from Equation 1) is

$$T_2 = \alpha\left(\frac{D_0}{1+v} + L_1\right) + \frac{D_1}{1+v} + L_2$$

If we were to return to the origin after L_2 (and d denotes the time-0 distance of the end of L_2 to the origin) the total time would be

$$T' = d + (1-v)T_2 = D_0\alpha^2 + D_1\alpha + d + (1-v)[L_1\alpha + L_2].$$

Generalizing this gives the following expression for $T^{(k)}$:

$$T^{(k)} = D_0\alpha^m + D_1\alpha^{m-1} + \ldots + D_{m-1}\alpha + D_m + (1-v)[L_1\alpha^{m-1} + L_2\alpha^{m-2} + \ldots + L_m] \qquad (8)$$

It might appear from this expression that $T^{(k)} \to 0$ as $\alpha \to 0$. However, it must be kept in mind that the expression is only valid if at all times, all unvisited moving-points remain to the right of the origin (i.e., in the positive quadrant). In the moving belt scenario, the origin represents the end of the belt. A small α corresponds to a v close to 1, whereas the problem is meaningful only if v is small enough that the points do not cross the origin when the robot is in the process of grasping them. A reasonable restriction on v is that the time for an optimal k-Collect TSP tour should not suffice for any point to cross the origin. This implies that $C_1/k \le d_{av}/v$, where d_{av} denotes the average of the distances d_i and C_1/k is a lower bound on the optimal in the static problem (see for instance [1]). Thus, we will assume that $v(2d_{av}n)/k \le d_{av}$, i.e., $v \le \frac{k}{2n}$.

Given that $v \le \frac{k}{2n}$, in the expression (8) for $T^{(k)}$, the smallest coefficient on any term is

$$\alpha^m = \alpha^{n/k} \ge \frac{(1-\frac{k}{2n})^{n/k}}{(1+\frac{k}{2n})^{n/k}} \ge \frac{1}{e}$$

This implies that $T^{(k)} \ge \frac{1}{e}(D_0 + L_1 + D_1 + L_2 + D_2 + \ldots + L_m + D_m)$.

Let us denote the sum of terms in the parentheses by T'. Minimizing T' is a static k-Collect TSP problem on $n+1$ vertices $v_0, v_1, \ldots, v_n$ where the directed distances are defined as in the C_∞ approximation, except that for all $i > 0$, $d(v_0, v_i) = d_i = d(v_i, v_0)$. As we saw before, the ratio of the two directed distances between a pair of vertices never exceeds $1/\alpha$, so the k-Collect TSP approximation algorithm of [1] can be used to approximate T' to within a factor $2.5/\alpha$ of optimal. Thus, we can approximate the optimal k-Collect tour time C_k to within a factor $2.5e/\alpha$.

Theorem 17 *For $v \le k/2n$, there is a $(2.5e/\alpha)$-approximation algorithm for Dynamic k-Collect TSP.*

The following lower bound, although not used in this paper, may lead to a different approximation algorithm for C_k than the one we presented above.

Theorem 18 *For any finite k, $C_\infty \le C_k$.*

Proof: Consider the optimal tour C_k for capacity k. (We will abuse notation and use the same symbol to stand for a tour as well as its length). Let d be the time-0 distance from the origin, of the final moving-point p visited by C_k before returning to the origin, and let T_p be the time taken by T up to this final point p. From equation 2, the total time taken is $C_k = T_p(1-v) + d$. Note that before reaching the final point p, T may do the following several times: go from a moving-point p_i to the origin, then to moving-point p_j. However by the Triangle Inequality I (Corollary 12), a direct path from p_i to p_j is no longer than the path via the origin, so we can short-circuit each such indirect path by a direct path and gain time. By applying these short-circuits to every origin-return except the last one, we can modify the optimal tour C_k to an unbounded-capacity tour that reaches p at an earlier time $T'_p \leq T_p$, and the total time of this tour would be $(1-v)T'_p + d$ which is no larger than C_k. ■

4.7 Geometric Latency

The somewhat unwieldy expression (8) for $T^{(k)}$ can be lower-bounded by a more pleasant cost expression which we call the *geometric latency* of a tour. We describe this below and show how a variant of the *maximum* latency problem arises implicitly in minimizing the geometric latency.

Consider the asymmetric k-Collect TSP problem with directed distances as defined in Section 4.6. Now consider a capacity-k tour on this graph, and let $e_1, e_2, \ldots, e_u$ be the sequence of (weights of) edges traversed. Fix some positive $\beta < 1$, and define the *geometric latency* G_β of this tour to be $G_\beta = \sum_{i=1}^{u} \beta^{u-i} e_i$. Assuming for convenience that n is a multiple of k, $u = m + n$.

Lemma 19 *Let $\beta = \alpha^{1/k}$ and fix a capacity-k tour of length $T^{(k)}$ for the moving points problem. Let G_β be the geometric latency of the corresponding (i.e., same order of visiting points) tour in the corresponding asymmetric k-Collect TSP instance as described above. Then $T^{(k)} \geq G_\beta$.*

Proof: Consider expression (8) for $T^{(k)}$. Since $(1-v) \geq \alpha$,

$$\begin{aligned} T^{(k)} \quad &\geq \quad (D_0 + L_1)\alpha^m \ + \ (D_1 + L_2)\alpha^{m-1} \\ &\quad + \ \ldots \ + \ (D_{m-1} + L_m)\alpha \ + \ D_m. \end{aligned}$$

The lemma follows by comparing weights of corresponding terms in this expression and the one for G_β. ■

Thus, if we are able to find a constant-factor approximation to a capacity-k tour with minimum Geometric latency G_β, we would have a constant factor approximation for the dynamic k-Collect TSP problem. We know of no such algorithm that runs in polynomial time. It is worth noticing that G_β may be rewritten as

$$G_\beta = \sum_{i=1}^{u} e_i - \sum_{i=1}^{u-1} (1 - \beta^{u-i} e_i),$$

so minimizing G_β roughly involves simultaneously minimizing the total tour length and maximizing the second term, which is a variant of the linear latency. As we mentioned in Section 3, we can approximately maximize the second term, but we do not know how to simultaneously bound the length of the tour.

5 Conclusion

In this paper we have initiated a formal algorithmic study of problems arising in the context of rapid deployment automation. Specifically, we have considered problems related to rearranging parts by a robot in an industrial assembly line setting. Our work applies to parts both on a static as well as a moving conveyer belt. The static belt problems, which are variants of a problem we define as k-Delivery TSP, and the dynamic belt problems, variants of k-Collect TSP, are related to well known problems in the literature.

All the algorithms we have presented are, to the best of our knowledge, either the first polynomial time constant-factor approximation algorithms for the problems, or provide a better constant approximation than was previously known. The algorithms themselves are very simple and lend themselves to easy implementation. We briefly mention some easy extensions of the model and results for the moving belt scenario; the details are omitted.

- The first extension is to the case where all points are moving *away* from the origin at velocity v. For instance, this would be the case in mid-air refueling of a formation of planes. The results and analysis are similar to that presented above.

- The second extension is to the case where the conveyor belt is circular and is rotating around the origin. We can obtain a constant-factor approximation for the case where the rotation speed is bounded below the robot's speed.

A variety of open problems and new directions for algorithmic and robotic research arise out of our work. In general, there are a large number of problems related to the efficient design, simulation, and implementation of rapid deployment automation [9]. For instance: a typical design problem is the optimal placement of modules, e.g., given a fixed path, determine a location for the robot base so as to minimize cycle time; a typical simulation problem is that of finding space for dropped parts on a conveyor belt, i.e., find a pose where a new part can be placed without collision.

More specific to our work, it is important to experimentally study the performance of the algorithms presented here. Then, there is the natural question of whether the approximation results obtained in this paper can be improved, hopefully without sacrificing on the efficiency of the algorithms. In the case of a moving conveyor belt, there is the issue of weakening the assumption we made regarding the relative velocity of the conveyor belt. One approach to this would be to devise an efficient approximation algorithm for the Maximum Latency TSP problem with respect to the notion of geometric latency. It also remains to be seen whether any of our results can be extended to other vehicle routing and scheduling problems such as Dial-a-Ride and Precedence-Constrained TSP.

An important extension to our work would be to consider more complex cost metrics. The cost metric used in this paper is the total translational (x, y) distance, D, traveled by the end-effector to deliver all n parts. We neglected the cost of rotating the end-effector to orient itself correctly over each part prior to pick-up as well as the reorientation at delivery. Considering rotation would increase our configuration space by one dimension. As long as the rotational velocity of the end-effector is proportional to its (assumed constant) translational velocity, our algorithms can be extended to higher dimensions, with exponential dependence on dimension. We could also consider cost metrics of the form $D + c$, where c is a constant (or only dependent on n). Such a cost metric could model, for example, damping and the z translation by the end-effector (the total z-translation depends only on the number of parts, n, and not on the actual configuration of the parts). Finally, an assumption that could be relaxed is that of uniform (x, y) velocity of the robot end-effector so as to model the non-zero accelerations of the end-effector as it starts its motion to a new part and as it gets there. This leads to a more complex bounded-velocity problem instead of our constant-velocity problem.

Acknowledgments

We are deeply indebted to Ken Goldberg for introducing us to this family of problems, and for his valuable feedback and constant encouragement during the course of this work. We are also grateful to Alan Frieze for his help with the Bipartite Traveling Salesman Problem as well as for his encouragement. We thank Rafi Hassin for pointing out a simplification in our arguments in Section 2. Finally, we are grateful to an anonymous referee for suggestions that improved the presentation.

References

[1] K. Altinkemer and B. Gavish. Heuristics for delivery problems with constant error guarantees. *Transportation Science*, 24(4):294–297, 1990.

[2] S. Anily and R. Hassin. The swapping problem. *Networks*, 22:419–433, 1992.

[3] M.J. Atallah and S.R. Kosaraju. Efficient solutions to some transportation problems with applications to minimizing robot arm travel. *SIAM Journal on Computing*, 17:849–869 (1988).

[4] B. Awerbuch, Y. Azar, A. Blum, and S. Vempala. Improved Approximation Guarantees for Minimum-Weight k-Trees and Prize-Collecting Salesmen. In *Proceedings of the 27th Annual ACM Symposium on the Theory of Computing* (1995), pp. 277–283.

[5] L. Bianco, A. Mingozzi, S. Riccardelli, and M. Spadoni. Exact and heuristic procedures for the traveling salesman problem with precedence constraints, based on dynamic programming. *INFOR*, 32(1):19–32, 1994.

[6] A. Blum, P. Chalasani, D. Coppersmith, B. Pulleyblank, P. Raghavan, and M. Sudan. The Minimum Latency Problem. In *Proceedings of the 26th Annual ACM Symposium on the Theory of Computing* (1994), pp. 163–171.

[7] A. Blum, P. Chalasani, and S. Vempala. A Constant-Factor Approximation for the k-MST Problem in the Plane. In *Proceedings of the 27th Annual ACM Symposium on the Theory of Computing* (1995), pp. 294–302.

[8] C. Brezovec, G. Cornuejols, and F. Glover. A matroid algorithm and its application to the efficient solution of two optimization problems on graphs. *Mathematical Programming*, 42:471–487, 1988.

[9] B. Carlisle and K.Y. Goldberg. Report on TARDA. *Symposium on Theoretical Aspects of Rapid Deployment Automation*, Adept Technology, San Jose (1994).

[10] N. Christofides. Worst-case analysis for a new heuristic for the Traveling Salesman problem. In *Symposium on New Directions and Recent Results in Algorithms and Complexity*, (Ed: J.F. Traub), Academic Press (1976).

[11] N. Christofides. Vehicle routing. In *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization* (Ed: E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys), John Wiley & Sons (1985), pp. 431–448.

[12] J. Edmonds. Submodular functions, matroids and certain polyhedra. In *Combinatorial Structures and their Applications, Proceedings of Calgary International Conference*, pages 69–87, 1970.

[13] J. Edmonds. Matroid intersection. *Annals of Discrete Mathematics*, 4:39–49, 1979.

[14] Flexible Assembly Systems. *Proceedings of the 3rd American Society of Mechanical Engineers (ASME) Conference on Flexible Assembly Systems.* Block September 1991.

[15] B. Carlisle. Is U.S. Robotics Research of Any Use to U.S. Industry?. *IEEE Robotics and Automation Society Newsletter.* July, 1991.

[16] K.R. Fox, B. Gavish, and S.C. Graves. An n-constraint formulation of the (time-dependent) traveling salesman problem. *Operations Research* 28:1018–1021 (1980).

[17] G. N. Frederickson, M. Hecht, and C. Kim. Approximation algorithms for some routing problems. *SIAM Journal on Computing*, 7(2):178–193, 1978.

[18] A.M. Frieze. An Extension of Christofides' Heuristic to the k-person Traveling Salesman Problem. *Discrete Applied Mathematics*, 6:79–83, 1983.

[19] N. Garg and D.S. Hochbaum. An $O(\log k)$ approximation algorithm for the k minimum spanning tree problem in the plane. In *Proceedings of the 26th Annual ACM Symposium on the Theory of Computing* (1994), pp. 432–438.

[20] M.X. Goemans and J.M. Kleinberg. An Improved Approximation Ratio for the Minimum Latency Problem. In *Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms* (1996), to appear.

[21] B.L. Golden and A.A. Assad, Eds. *Vehicle Routing: Methods and Studies.* North-Holland, Amsterdam (1988).

[22] P. Hall. On representatives of subsets. *Journal of London Mathematics Society*, 10:26–30, 1935.

[23] J.A. Hoogeveen. Analysis of Christofides' heuristic: some paths are more difficult than cycles. *Operations Research Letters*, 10:291–295 (1991).

[24] K. Jansen. Bounds for the general capacitated routing problem. *Networks*, 23(3):165–173, 1993.

[25] M. Kubo and H. Kagusai. Heuristic algorithms for the single-vehicle dial-a-ride problem. *Journal of the Operations Research Society of Japan*, 30(4):354–365, 1990.

[26] J-C. Latombe. *Robot Motion Planning*. Kluwer Academic Press (1991).

[27] E. Lawler. *Combinatorial Optimization: Networks and Matroids*. Holt, Reinhart and Winston, New York, 1976.

[28] T-Y. Li and J-C. Latombe. On-line manipulation planning for two robot arms in a dynamic environment. In *Proceedings of the 12th Annual IEEE International Conference on Robotics and Automation* (1995), pp. 1048–1055.

[29] T. Lozano-Perez, J. Jones, E. Mazer, and P. O'Donnell. *Handey: A Robot Task Planner.* MIT Press, 1992.

[30] J. Lenstra, A. R. Kan. Complexity of Vehicle Routing and Scheduling Problems. *Networks* 11:221–227 (1981).

[31] A . Lucena. Time-dependent traveling salesman problem – the deliveryman case. *Networks* 20:753–763 (1990).

[32] J. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple new method for the geometric k-mst problem. In *7th Annual ACM-SIAM Symposium on Discrete Algorithms*, 1996.

[33] R. Motwani. *Approximation Algorithms*, Technical Report STAN-CS-92-1435, Department of Computer Science, Stanford University, 1992.

[34] *Proceedings of Robotic Industries Association (RIA) Workshop on Flexible Part Feeding.* October 1993.

[35] H. Psaraftis. Scheduling large-scale advance-request dial-a-ride systems. *American J. Math. Manage. Sci.*, 6:327–367, 1986.

[36] R. Ravi, R. Sundaram, M.V. Marathe, D.J. Rosenkrantz, and S.S. Ravi. Spanning trees short or small. In *Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms* (1994), pp. 546–555.

[37] D.J. Rosenkrantz, R.E. Stearns, and P.M. Lewis. An analysis of several heuristics for the traveling salesman problem . *SIAM Journal on Computing* 6:563–581 (1977).

[38] J.T. Schwartz, J. Hopcroft, and M. Sharir. *Planning, Geometry, and Complexity of Robot Motion.* Ablex, 1987.