Parallel Dual Coordinate Descent Method for Large-scale Linear Classification in Multi-core Environments

Wei-Lin Chiang, Mu-Chu Lee and Chih-Jen Lin
Department of Computer Science & Information Engineering, National Taiwan University

Introduction

- Dual coordinate descent (CD) method is one of the most effective approaches for large-scale linear classification (e.g., linear SVM).
- However, its **sequential** design makes the parallelization difficult.
- In this work,
  - We investigate **multi-core dual CD** methods for linear classification.
  - We propose a new framework to parallelize dual CD and establish its **theoretical convergence properties**.
- Further, we demonstrate through experiments that the method is robust and efficient.

Formulations

- Given training data $(\boldsymbol{x}_i, y_i) \in \mathcal{R}^n \times \{-1,1\}, i = 1,\ldots,l$.
- Linear classification obtains its model vector $\boldsymbol{w}$ by solving:

$$\min_{\boldsymbol{w}} \quad \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + CL(\boldsymbol{w}) \qquad (1)$$

$$\text{where } L(\boldsymbol{w}) = \sum_{i=1}^{l} \xi(\boldsymbol{w}; \boldsymbol{x}_i, y_i). \qquad (2)$$

Eq. (2) is the loss function, and two losses are considered:

$$\xi(\boldsymbol{w};\boldsymbol{x}_i,y_i) \equiv \begin{cases} \max(0, 1-y_i\boldsymbol{w}^T\boldsymbol{x}_i) & \text{L1-loss SVM,} \\ \max(0, 1-y_i\boldsymbol{w}^T\boldsymbol{x}_i)^2 & \text{L2-loss SVM.} \end{cases}$$

- If (1) is referred to as the primal problem, then a dual CD method solves the following dual problem:

$$\min_{\boldsymbol{\alpha}} \quad \frac{1}{2}\boldsymbol{\alpha}^T\bar{Q}\boldsymbol{\alpha} - \sum_{i=1}^{l}\alpha_i$$

$$\text{subject to} \quad 0 \le \alpha_i \le U, \forall i = 1,\ldots,l,$$

where $\bar{Q} = Q + D$ with $Q_{ij} = y_i y_j \boldsymbol{x}_i^T\boldsymbol{x}_j$, and D is diagonal with

$$D_{ii} = \begin{cases} 0 \\ \frac{1}{2C} \end{cases} \quad U = \begin{cases} C & \text{for L1-loss SVM,} \\ \infty & \text{for L2-loss SVM.} \end{cases}$$

- Each time an α_i is selected and a one-variable sub-problem is solved:

$$\min_d f(\boldsymbol{\alpha} + d\boldsymbol{e}_i) \text{ subject to } 0 \le \alpha_i + d \le U, \qquad (3)$$

where $\boldsymbol{e}_i = [\underbrace{0,\ldots,0}_{i-1}, 1, 0, \ldots, 0]^T$. Clearly,

$$f(\boldsymbol{\alpha} + d\boldsymbol{e}_i) = \frac{1}{2}\bar{Q}_{ii}d^2 + \nabla_i f(\boldsymbol{\alpha})d + \text{constant}.$$

The solution of (3) can be easily seen as

$$d = \min\left(\max\left(\alpha_i - \frac{\nabla_i f(\boldsymbol{\alpha})}{\bar{Q}_{ii}}, 0\right), U\right) - \alpha_i.$$

- A crucial observation in Hsieh et al. [2008] notes that if

$$\boldsymbol{w} \equiv \sum_{j=1}^{l} y_j\alpha_j\boldsymbol{x}_j \qquad (4)$$

is maintained, then $\nabla_i f(\boldsymbol{\alpha})$ can be easily calculated by

$$\nabla_i f(\boldsymbol{\alpha}) = (\bar{Q}\boldsymbol{\alpha})_i - 1 = \sum_{j=1}^{l}\bar{Q}_{ij}\alpha_j - 1 = y_i\boldsymbol{w}^T\boldsymbol{x}_i - 1 + D_{ii}\alpha_i. \qquad (5)$$

We can then update $\boldsymbol{\alpha}$ and maintain the weighted sum in (4) by

$$\alpha_i \leftarrow \alpha_i + d \quad \text{and} \quad \boldsymbol{w} \leftarrow \boldsymbol{w} + dy_i\boldsymbol{x}_i. \qquad (6)$$

- The main computation for updating an α_i includes two $O(n)$ operations in (5) and (6).
- Unfortunately, the procedure is inherently **sequential**.

A Practical Implementation for Dual CD

1: Specify a feasible $\boldsymbol{\alpha}$ and calculate $\boldsymbol{w} = \sum_j y_j\alpha_j\boldsymbol{x}_j$
2: **while** true **do**
3: **for** $i = 1,\ldots,l$ **do**
4: $G \leftarrow y_i\boldsymbol{w}^T\boldsymbol{x}_i - 1 + D_{ii}\alpha_i$ $\triangleleft\, O(n)$
5: PG (proj. grad.) $= \begin{cases} G & \text{if } 0 < \alpha_i < U, \\ \min(0,G) & \text{if } \alpha_i = 0, \\ \max(0,G) & \text{if } \alpha_i = U. \end{cases}$
6: **if** $|PG| \ge 10^{-12}$ **then**
7: $d \leftarrow \min(\max(\alpha_i - G/\bar{Q}_{ii}, 0), U) - \alpha_i$
8: $\alpha_i \leftarrow \alpha_i + d$
9: $\boldsymbol{w} \leftarrow \boldsymbol{w} + dy_i\boldsymbol{x}_i$ $\triangleleft\, O(n)$

Existing Works: Mini-batch Dual CD

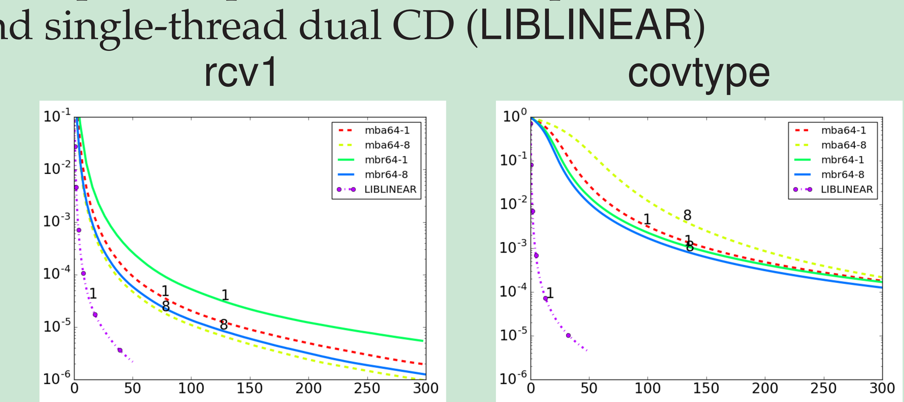
- Instead of running through $i = 1,\ldots,l$ in line 3 one by one, run a batch of i in parallel.
- For convergence, the step size d in line 7 is scaled down

$$\alpha_i \leftarrow \alpha_i + \beta d, \text{ where } \beta < 1$$

Takáč et al. [2015] discussed the condition of β and proved the convergence with suitable β.
- However, using conservative steps may cause **slower convergence**.
- In line 9, race conditions occur for multi-threading.

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \sum_{i \text{ in a batch}} d_i y_i \boldsymbol{x}_i \qquad (7)$$

Lee et al. [2015] detailed study this issue in a multi-core Newton method. They consider atomic and reduce operations.
  - However, even with careful settings, the overhead of (7) is significant because of the small batch size.
  - A simple comparison between parallel mini-batch CD and single-thread dual CD (LIBLINEAR)



- Therefore, we may give up parallelizing (7) in multi-core environments.

Existing Works: Asynchronous Dual CD

- To address the slow convergence of mini-batch CD, Hsieh et al. [2015] and Tran et al. [2015] parallelize the for loop (line 3) so each thread updates α_i **asynchronously**. For line 9, $\boldsymbol{w}$ can be updated by **atomic operations**.
- Since the processors are running concurrently, $\boldsymbol{w}$ may change between the start (line 4) and the end (line 9) of one CD step.
- For convergence, the iteration lag τ is the key variable for analysis. Specifically, the sequence $\{\boldsymbol{\alpha}^k\}$ should satisfy

$$k \le \bar{k} + \tau$$

where $\bar{k}$ is the iteration index when iteration k starts.
- The iteration lag τ must satisfy some conditions. However, the conditions may not hold, so asynchronous CD **may diverge**.

Our Idea and Design

- For convergence, we don't use asynchronous updates.
- We sequentially update $\boldsymbol{w}$ due to the race condition in (7). However, we ensure that this takes a **small portion** while others are **parallelizable**.

Proposed Parallel Dual CD Method

- In CD a selected α_i may not need to be updated. After calculating $\nabla_i f(\boldsymbol{\alpha})$, we know if that's the case in line 6. Practically we have

$$\underbrace{\alpha_1^k,\ldots,\alpha_i^k}_{\text{unchanged}}, \quad \alpha_{i+1}^k, \quad \underbrace{\alpha_{i+2}^k,\ldots,\alpha_j^k}_{\text{unchanged}}, \quad \alpha_{j+1}^k, \quad \ldots$$

- If we know α_i^k is unchanged, then $\nabla_i f(\boldsymbol{\alpha})$ doesn't need to be calculated.
- Idea: a setting to guess that some α_i are unchanged
  - Calculate $\nabla_i f(\boldsymbol{\alpha}), \forall i \in \bar{B}$ **in parallel**.
  - Select a **much smaller** subset B from $\bar{B}$ to do **sequential** CD updates.

  That is, we conjecture $\alpha_i, i \in \bar{B} \setminus B$ need not be updated.
- A new framework:

1: **while** true **do**
2: Select a set $\bar{B}$
3: Calculate $\nabla_{\bar{B}} f(\boldsymbol{\alpha})$ in parallel
4: Select $B \subset \bar{B}$ with $|B| \ll |\bar{B}|$
5: Sequentially update $\alpha_i, i \in B$

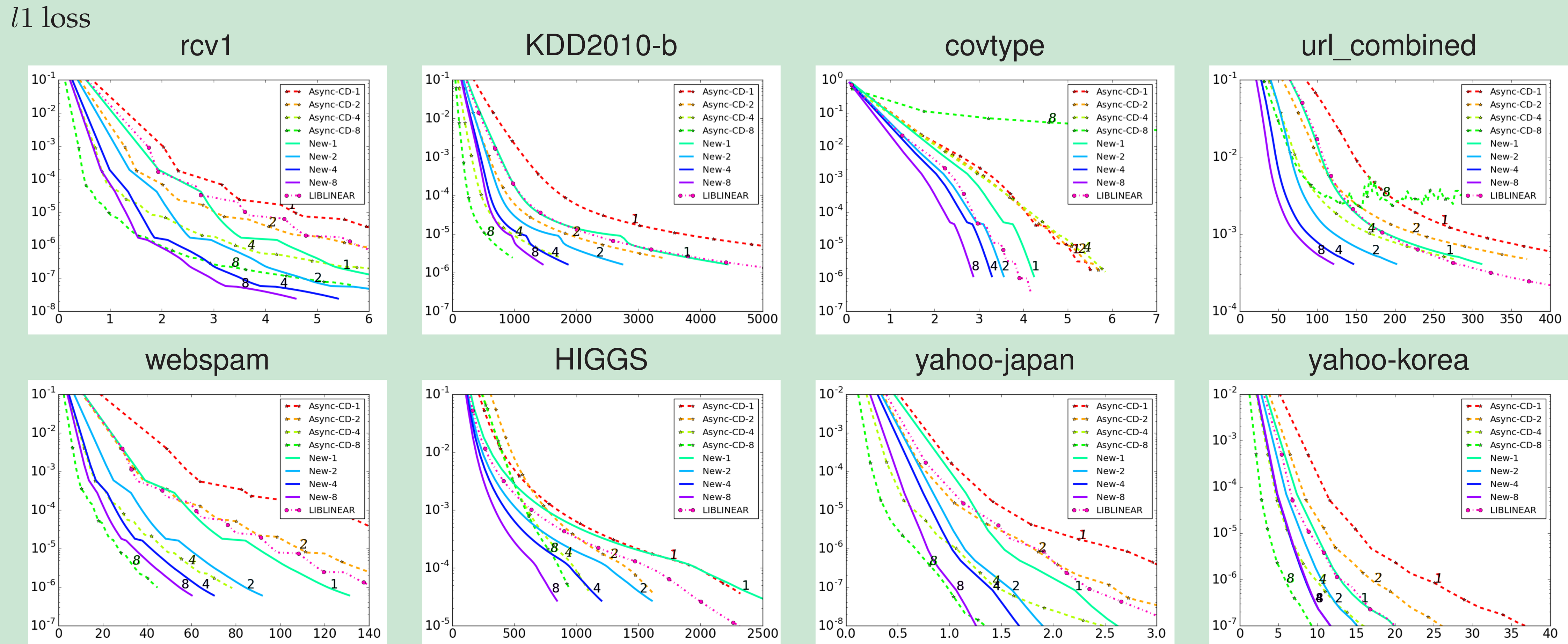
Implementation of the Proposed Framework

- The selection of B is essential. An example:
  - $\{1,\ldots,l\}$ splits to $\bar{B}_1,\ldots,\bar{B}_T$
  - For each $\bar{B}$ in $\bar{B}_1,\ldots,\bar{B}_T$ select elements in $\bar{B}$ with larger project gradient as B.
- **Theoretical convergence** is established.
- Other selections of B are possible.
- The block size $|\bar{B}|$ is also important
  - too small $|\bar{B}|$ may cause parallelization overhead
  - too large $|\bar{B}|$ may cause slower convergence

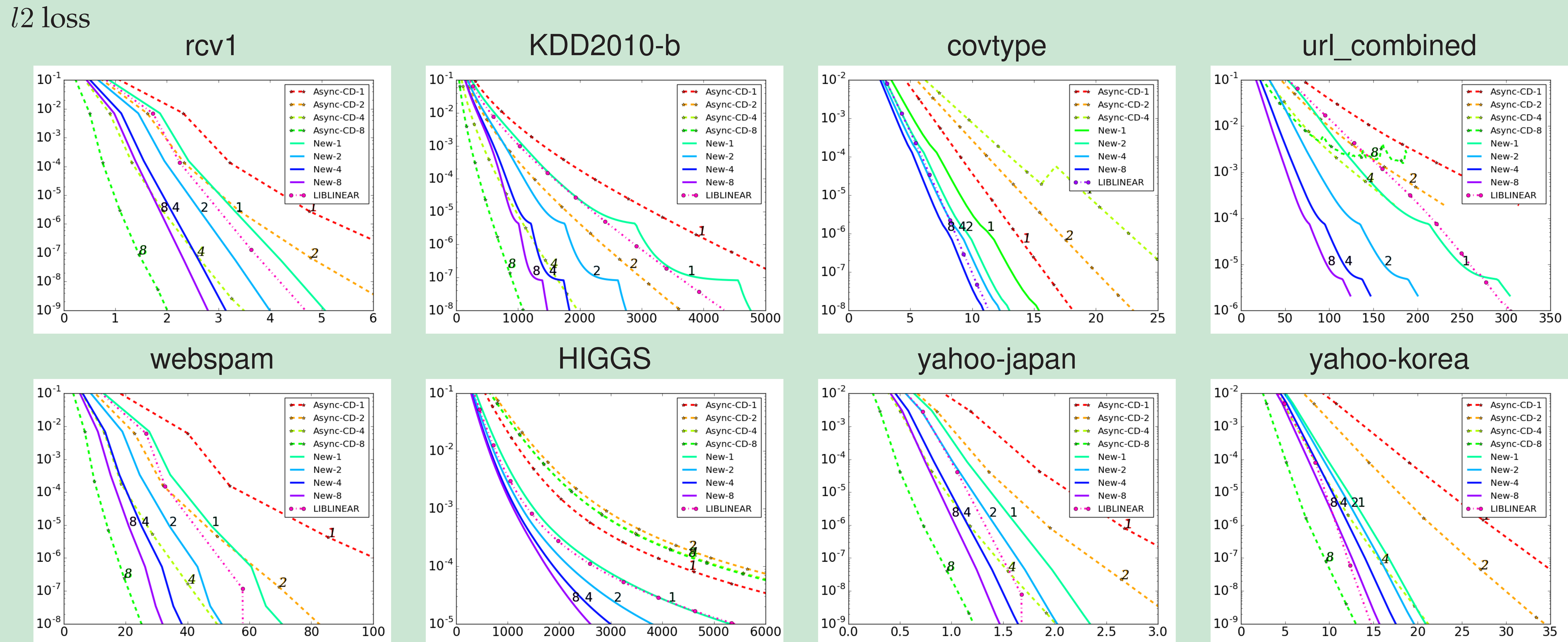
  Fortunately, we found that the training time is about the same when $|\bar{B}|$ is set to be a few hundreds.
- Shrinking technique in Hsieh et al. [2008] for removing some unnecessary α_i can be incorporated.

Comparison: asynchronous CD, our proposed method and single-core LIBLINEAR

- x-axis is the training time in seconds, y-axis is the relative error, and "New" is our method.
- $l1$ loss



- $l2$ loss



- Asynchronous CD is efficient, but may fail when using more threads.

Conclusions

- We propose an effective parallel dual CD framework for multi-core environments.
- Future direction: dual CD in multi-CPU environments.
- Multi-core LIBLINEAR is available at:
`http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/multicore-liblinear`