Journal of Parallel and Distributed Computing 143 (2020) 47–66

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Hybrid-DCA: A double asynchronous approach for stochastic dual coordinate ascent

Soumitra Pal [a,1], Tingyang Xu [b,1], Tianbao Yang [c], Sanguthevar Rajasekaran [d], Jinbo Bi [d,*]

[a] National Center for Biotechnology Information, National Library of Medicine, National Institutes of Health, Bethesda, MD 20894, USA
[b] Tencent AI Lab, Shenzhen, Guangzhou, 518000, China
[c] Department of Computer Science, University of Iowa, Iowa City, IW 52242, USA
[d] Department of Computer Science and Engineering, University of Connecticut, Storrs, CT 06269, USA

ARTICLE INFO

Article history:
Received 4 May 2017
Received in revised form 17 February 2020
Accepted 3 April 2020
Available online 13 April 2020

Keywords:
Dual coordinate descent
Distributed computing
Optimization

ABSTRACT

In prior works, stochastic dual coordinate ascent (SDCA) has been parallelized in a multi-core environment where the cores communicate through shared memory, or in a multi-processor distributed memory environment where the processors communicate through message passing. In this paper, we propose a hybrid SDCA framework for multi-core clusters, the most common high performance computing environment that consists of multiple nodes each having multiple cores and its own shared memory. We distribute data across nodes where each node solves a local problem in an asynchronous parallel fashion on its cores, and then the local updates are aggregated via an asynchronous across-node update scheme. The proposed double asynchronous method converges to a global solution for L-Lipschitz continuous loss functions, and at a linear convergence rate if a smooth convex loss function is used. Extensive empirical comparison has shown that our algorithm scales better than the best known shared-memory methods and runs faster than previous distributed-memory methods. Big datasets, such as one of 280 GB from the LIBSVM repository, cannot be accommodated on a single node and hence cannot be solved by a parallel algorithm. For such a dataset, our hybrid algorithm takes less than 30 s to achieve a duality gap of 10^{-5} on 16 nodes each using 12 cores, which is significantly faster than the best known distributed algorithms, such as CoCoA+, that take more than 160 s on 16 nodes.

1. Introduction

The immense growth of data has made it important to efficiently solve large scale machine learning problems. It is necessary to take advantage of modern high performance computing (HPC) environments such as multi-core settings where the cores communicate through shared memory, or multi-processor distributed memory settings where the processors communicate by passing messages. In particular, a large class of supervised learning formulations, including support vector machines (SVMs), logistic regression, ridge regression and many others, solve the following generic regularized risk minimization (RRM) problem: given a set of instance-label pairs of data points $(\mathbf{x}_i, y_i), i = 1, \ldots, n$,

$$\min_{\mathbf{w}\in\mathbb{R}^d} P(\mathbf{w}) := \frac{1}{n}\sum_{i=1}^{n}\phi(\mathbf{x}_i^\top \mathbf{w}; y_i) + \frac{\lambda}{2}g(\mathbf{w}), \tag{1}$$

where $y_i \in \mathbb{R}$ is the label for the data point $\mathbf{x}_i \in \mathbb{R}^d$, $\mathbf{w} \in \mathbb{R}^d$ is the linear predictor to be optimized, ϕ is a loss function that is convex with respect to its first argument, λ is a regularization parameter that balances between the loss and a regularizer $g(\mathbf{w})$, which for instance can take the squared ℓ_2-norm $\|\mathbf{w}\|_2^2$.

Many efficient sequential algorithms have been developed in the past decades to solve (1), *e.g.,* stochastic gradient descent (SGD) [25], or alternating direction method of multipliers (ADMM) [2]. Especially, (stochastic) dual coordinate ascent (DCA) algorithm [18] has been one of the most widely used algorithms for solving (1). It efficiently optimizes the following dual formulation (2)

$$\max_{\boldsymbol{\alpha}\in\mathbb{R}^n} D(\boldsymbol{\alpha}) := -\frac{1}{n}\sum_{i=1}^{n}\phi^*(-\alpha_i) - \frac{\lambda}{2}g^*\left(\frac{1}{\lambda n}\mathbf{X}\boldsymbol{\alpha}\right), \tag{2}$$

* Corresponding author.
E-mail addresses: soumitra.pal@nih.gov (S. Pal), tingyangxu@tencent.com (T. Xu), tianbao-yang@uiowa.edu (T. Yang), rajasekaran.sanguthevar@uconn.edu (S. Rajasekaran), jinbo.bi@uconn.edu (J. Bi).

[1] The first author S. Pal and the second author T. Xu contributed to this work when they were with the Department of Computer Science and Engineering, University of Connecticut, Storrs, CT.

https://doi.org/10.1016/j.jpdc.2020.04.002

where $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n] \in \mathbb{R}^{d \times n}$, $\phi^*(u)$ and $g^*(\mathbf{v})$ are the convex conjugates of $\phi(z; y)$ (or in short $\phi(z)$ where the scalar $z = \mathbf{x}^T\mathbf{w}$) and $g(\mathbf{w})$, respectively. The conjugate of the loss function $\phi(z)$ is defined as $\phi^*(u) = \max_z(zu - \phi(z))$. Let $\nabla g^*(\mathbf{v})$ be the gradient of g^* with respect to $\mathbf{v}$ where $\mathbf{v}(\boldsymbol{\alpha}) = \frac{1}{\lambda n}\mathbf{X}\boldsymbol{\alpha}$. We have

$$\mathbf{w}(\boldsymbol{\alpha}) = \nabla g^*(\mathbf{v}). \tag{3}$$

It is known from duality theory that if $\boldsymbol{\alpha}^*$ is an optimal dual solution, then the vector $\mathbf{w}^* = \mathbf{w}(\boldsymbol{\alpha}^*)$ is an optimal primal solution and $P(\mathbf{w}^*) = D(\boldsymbol{\alpha}^*)$. The dual objective has a separate dual variable α_i associated with each training data point $\mathbf{x}_i$. The stochastic DCA updates dual variables, one at a time, while maintaining the primal variables by calculating (3) from the dual variables.

Recently, much effort has been undertaken to solve Problem (1) in a distributed or parallel framework. It has been shown that distributed DCA algorithms have comparable and sometimes even better convergence than SGD-based or ADMM-based distributed algorithms [23]. The distributed DCA algorithms can be grouped into two sets. The first set contains synchronous algorithms in which a random dual variable is updated by each processor and the primal variables are synchronized across the processors in every iteration [8,11,23]. This approach incurs a large communication overhead. The second set of algorithms avoids communication overhead by exploiting the shared memory in a multi-core setting [7] where the primal variables are stored in a primary memory shared across all the processors. Further speedups have been obtained by using (asynchronous) atomic memory operations instead of costly locks for shared memory updates [7,16]. Nevertheless, this approach is difficult to scale up for big datasets that cannot be fully accommodated in the shared memory. This leads to a challenging question: how do we scale up the asynchronous shared memory approach for big data while maintaining the speed up?

We address this challenge by proposing and implementing a hybrid strategy. The modern HPC platforms can be viewed as a collection of K nodes interconnected through a network as shown in Fig. 1(a). Each node contains a memory shared among R processing cores. Our strategy exploits this architecture by equally distributing the data across the local shared memory of the K nodes. Each of the R cores within a node runs a computing thread that asynchronously updates a random dual variable from those associated with the data allocated to the node. Each node also runs a communicating thread. One of the communicating threads is designated as a *master* and the rest are *workers*. After every round of H local iterations in each computing thread, each worker thread sends the local update to the master. After accumulating the local updates from S of the K workers, the master broadcasts the global update to the contributing workers. However, to avoid a slower worker falling back too far, the master ensures that in every Γ consecutive global updates there is at least one local update from each worker. Fig. 1(b) shows how our scheme is a generalization of the existing approaches: for $K = 1$, our setup coincides with the shared memory multi-core setting [7] and for $R = 1, S = K$ our setup coincides with the synchronous algorithms in distributed memory setting [8,11,23]. With a proper adjustment of the parameters H, S, Γ our strategy could balance the computation time of the first setting with the communication time of the second one, while ensuring scalability in big data applications.

Thus, our contributions are (1) we propose and analyze a hybrid asynchronous shared memory and asynchronous distributed memory implementation (*Hybrid-DCA*) of the mostly used DCA algorithm to solve (1); (2) we prove a strong guarantee of convergence for L-Lipschitz continuous loss functions, and further linear convergence when a smooth convex loss function is used; and (3) the experimental results using our light-weight OpenMP+MPI implementation show that our algorithms are much faster than existing distributed memory algorithms [8,11], and easily scale up with the volume of data in comparison with the shared memory based algorithms [7] as the shared memory size is limited.

2. Related work

Sequential Algorithms. SGD is the oldest and simplest method for solving problem (1). Though SGD is easy to implement and converges to modest accuracy quickly, it requires a long tail of iterations to reach 'good' solutions and also requires adjusting a step-size parameter. On the other hand, SDCA methods are free of learning-rate parameters and have faster convergence rate around the end [14,15]. A modified SGD has also been proposed with faster convergence by switching to SDCA after quickly reaching a modest solution [18]. Recently, 'variance reduced' modifications to the original SGD have also caught attention. These modifications estimate stochastic gradients with corrections to reduce the estimation variance. Mini-batch algorithms are also proposed to update several dual variables (data points) in a batch rather than a single data point per iteration [22]. Mini-batch versions of both SGD and SDCA have slower convergence when the batch size increases [17,19]. These sequential algorithms become ineffective when the datasets get bigger.

Distributed Algorithms. In the early single communication scheme [5,12,13], a dataset is 'decomposed' into smaller parts that can be solved independently. The final solution is reached by 'accumulating' the partial solutions using a single round of communications. This method has limited utility because most datasets cannot be decomposed in such a way. Using the primal–dual relationship (3), fully distributed algorithms of DCA are later developed where each processor updates a separate α_i which is then used to update $\mathbf{w}(\boldsymbol{\alpha})$, and synchronizes $\mathbf{w}$ across all processors (*e.g.,* CoCoA [8]). To trade off communications vs computations, a processor can solve its subproblem with H dual updates before synchronizing the primal variable (*e.g.,* CoCoA+ [11], DisDCA [23]). In [11,23], a more general framework is proposed in which the subproblem can be solved using not only SDCA but also any other sequential solver that can guarantee a Θ-approximation of the local solution at a processor for some $\Theta \in (0, 1]$. Nevertheless, the synchronized update to the primal variables has the inherent drawback that the overall algorithm runs at a speed of the slowest processor even when there are fast processors [1].

Parallel Algorithms. Multi-core shared memory systems have also been exploited, where the primal variables are maintained in a shared memory, removing the communication cost. However, updates to shared memory requires synchronization primitives, such as locks, which again slows down computation. Recent methods [7,10] avoid locks by exploiting (asynchronous) atomic memory updates in modern memory systems. There is even a wild version in [7] that takes arbitrarily one of the simultaneous updates. Though the shared memory algorithms are faster than the distributed versions, they have an inherent drawback of being not scalable, as there can be only a few cores in a processor board.

Other Distributed Methods for RRM. Besides distributed DCA methods, there are several recent distributed versions of other algorithms with faster convergence, including distributed Newton-type methods (DISCO [28], DANE [20]) and distributed stochastic variance reduced gradient method (DSVRG [9]). It has been shown that they can achieve the same accurate solution using fewer rounds of communication, however, with additional computational overhead. In particular, DISCO and DANE need to solve a linear system in each round, which could be very expensive for higher dimensions. DSVRG requires each machine to load

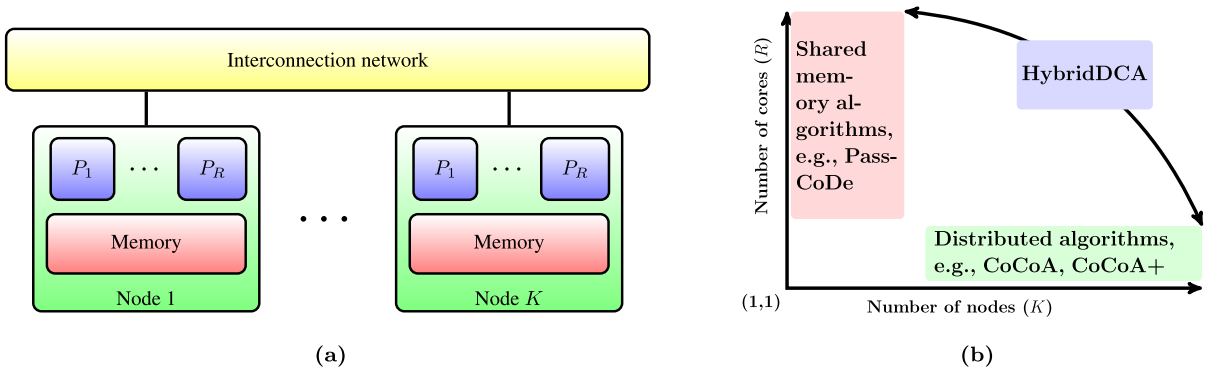


Fig. 1. (a) A simplified view of the modern HPC system and (b) Algorithms on this architecture.

and store a second subset of the data sampled from the original training data, which also increases its running time.

The ADMM [2] and quasi-Newton methods such as L-BFGS also have distributed solutions. These methods have low communication cost, however, their inherent drawback of computing the full batch gradient does not give computation vs communications trade-off. In the context of consensus optimization, [27] gives an asynchronous distributed ADMM algorithm but that does not directly apply to solving (1).

To the best of our knowledge, this paper is the first to propose, implement and analyze a hybrid approach exploiting modern HPC architecture. Our approach is the amalgamation of three different ideas – (1) CoCoA+/DisDCA distributed framework, (2) asynchronous multi-core shared-memory solver [7] and (3) asynchronous distributed approach [27] – taking the best of each of them. In a sense ours is the first algorithm which asynchronously uses updates which themselves have been computed using asynchronous methods.

3. The proposed algorithm

At the core of our algorithm, the data are distributed across K nodes and each node, called a *worker*, repeatedly solves a perturbed dual formulation on its data partition and sends the local update to one of the workers additionally designated as the *master* which merges the local updates and sends back the accumulated global update to the workers to solve the subproblem once again, unless a global convergence is reached. Let $\mathcal{I}_k \subseteq \{1, 2, \ldots, n\}$, $k = 1, \ldots, K$ denote the indices of the data and the dual variables residing on node k and $n_k = |\mathcal{I}_k|$. For any $\boldsymbol{\alpha} \in \mathbb{R}^n$ let $\boldsymbol{\alpha}_{[k]}$ denote the vector in $\mathbb{R}^n$ defined in such a way that the ith component $(\boldsymbol{\alpha}_{[k]})_i = \alpha_i$ if $i \in \mathcal{I}_k$, 0 otherwise, so that $\boldsymbol{\alpha} = \sum_k \boldsymbol{\alpha}_{[k]}$. Let $\mathbf{X}_{[k]} \in \mathbb{R}^{d\times n}$ denote the matrix consisting of the columns of the $\mathbf{X} \in \mathbb{R}^{d\times n}$ indexed by $\mathcal{I}_k$ and replaced with zeros in all other columns, so that $\mathbf{X} = \sum_k \mathbf{X}_{[k]}$.

Ideally, the dual problem solved by node k is (2) with $\mathbf{X}, \boldsymbol{\alpha}$ replaced by $\mathbf{X}_{[k]}, \boldsymbol{\alpha}_{[k]}$, respectively, and hence is independent of other nodes. However, following the efficient practical implementation in [11,23], we let the workers communicate among them a vector $\mathbf{v} \in \mathbb{R}^d$, an estimate of $\mathbf{w}(\boldsymbol{\alpha}) = \frac{1}{\lambda n}\mathbf{X}\boldsymbol{\alpha}$ that summarizes the last known global solution $\boldsymbol{\alpha}$. Also following [11,23] for faster convergence, each worker in our algorithm solves the following perturbed local dual problem, which we henceforth call the *subproblem*:

$$\max_{\boldsymbol{\delta}_{[k]}\in\mathbb{R}^n} D_k(\boldsymbol{\delta}_{[k]}; \mathbf{v}, \boldsymbol{\alpha}_{[k]}) := -\frac{1}{n_k}\sum_{i\in\mathcal{I}_k} \phi^*(-\alpha_i - \delta_i) - \frac{\lambda}{2S} g^*(\mathbf{v}) - \langle \frac{1}{n}\mathbf{X}_{[k]}^\top \nabla g^*(\mathbf{v}), \boldsymbol{\delta}_{[k]}\rangle - \frac{\lambda\sigma}{2}\left\| \frac{1}{\lambda n}\mathbf{X}_{[k]}\boldsymbol{\delta}_{[k]}\right\|^2 \quad (4)$$

where $\boldsymbol{\delta}_{[k]}$ denotes the local (incremental) update to the dual variable $\boldsymbol{\alpha}_{[k]}$, the *bounding barrier parameter* S denotes the number of workers from which the updates would be merged by the master in a global iteration and the *scaling parameter* σ measures the difficulty of solving the given data partition (see [11,23]) and must be chosen such that

$$\sigma \geq \sigma_{min} := \nu \max_{\boldsymbol{\alpha}\in\mathbb{R}^n} \frac{\|\mathbf{X}\boldsymbol{\alpha}\|^2}{\sum_{k=1}^K \left\|\mathbf{X}_{[k]}\boldsymbol{\alpha}_{[k]}\right\|^2} \quad (5)$$

where the *aggregation parameter* $\nu \in [\frac{1}{S}, 1]$ is the weight given by the master to each of local updates from the contributing workers while computing the global update. Unlike the synchronous all reduce approach in [11], our asynchronous method merges the local updates from only S out of K nodes in each global update and the second term in the objective of our subproblem (4) has denominator S instead of K. By Lemma 3.2 in [11], $\sigma := \nu S$ is a safe choice to hold condition (5).

3.1. Asynchronous updates by cores in a worker node

In each communication round, each worker k solves its subproblem using a parallel asynchronous DCA method [7] on the R cores. Let the data partition $\mathcal{I}_k$ stored in the shared memory be logically divided into R subparts where subpart $\mathcal{I}_{k,r} \subseteq \mathcal{I}_k, r = 1, \ldots, R$, is exclusively used by core r. In each of the H iterations, core r chooses a random coordinate $i \in \mathcal{I}_{k,r}$ and updates $\boldsymbol{\delta}_{[k]}$ in the ith unit direction by a step size ε computed using a single variable optimization problem:

$$\varepsilon = \operatorname*{argmax}_{\varepsilon\in\mathbb{R}} D_k(\varepsilon\mathbf{e}_i; \mathbf{v}, \boldsymbol{\alpha}_{[k]} + \boldsymbol{\delta}_{[k]}) \quad (6)$$

which has a closed form solution for SVM problems [4], and a solution using an iterative solver for logistic regression problems [24]. The local updates to $\mathbf{v}$ are also maintained appropriately. Because the coordinates used by any two cores are randomly chosen in parallel, the corresponding updates to $\boldsymbol{\delta}_{[k]}$ are independent of each other. Thus, there might be conflicts in the updates to $\mathbf{v}$ if the corresponding columns (the allocated examples in different cores) in X have nonzero values in the same row (resulting in different updates at the same element position of $\mathbf{v}$). We use lock-free *atomic* memory updates to handle such conflicts. When all cores complete H iterations, worker k sends the accumulated update $\Delta\mathbf{v}$ from the current round to the master; waits until it receives the globally updated $\mathbf{v}$ from the master; and repeats for another round unless the master indicates termination.

Algorithm 1: Hybrid-DCA: Worker k

Input: Initial $\boldsymbol{\alpha}_{[k]} \in \mathbb{R}^n$, data partition $\mathcal{I}_k$, initial $\mathbf{v} = \frac{1}{\lambda n}\mathbf{X}\boldsymbol{\alpha}$, scaling parameter σ, aggregation parameter ν, barrier bound parameter S

for $t \leftarrow 0, 1, \ldots$ **do**
  $\boldsymbol{\delta}_{[k]} \leftarrow \mathbf{0}$, $\mathbf{v}_{old} \leftarrow \mathbf{v}$;
  for *core* $r \leftarrow 1, \ldots, R$ ***in parallel*** **do**
    for $h \leftarrow 0, 1, \ldots, H-1$ **do**
      Randomly pick i from $\mathcal{I}_{k,r}$;
      $\varepsilon \leftarrow \text{argmax}_{\varepsilon} D_k(\varepsilon \mathbf{e}_i; \mathbf{v}, \boldsymbol{\alpha}_{[k]} + \boldsymbol{\delta}_{[k]})$;
      $\boldsymbol{\alpha}_{[k]} \leftarrow \boldsymbol{\alpha}_{[k]} + \varepsilon \mathbf{e}_i$;
      $\mathbf{v} \xleftarrow{atomic} \mathbf{v} + \nabla g^*\left(\frac{1}{\lambda n}\mathbf{X}_{[k]}\varepsilon \mathbf{e}_i\right)$ where $\mathbf{e}_i$ is a unit vector whose ith entry is 1;
  send $\Delta\mathbf{v} \leftarrow \mathbf{v} - \mathbf{v}_{old}$ to the master;
  receive $\mathbf{v}$ from the master;
  $\boldsymbol{\alpha}_{[k]} \leftarrow \boldsymbol{\alpha}_{[k]} + \nu \boldsymbol{\delta}_{[k]}$;

Algorithm 2: Hybrid-DCA: Master

Input: Initial $\mathbf{v}^{(0)} = \frac{1}{\lambda n}\mathbf{X}\boldsymbol{\alpha}$, aggregation parameter ν, barrier bound parameter S, delay bound parameter Γ, initial $\mathcal{P} = \emptyset$, initial delay counts $\mathbf{T} = \mathbf{0}$

for $t \leftarrow 0, 1, \ldots$ **do**
  while $|\mathcal{P}| < S$ **or** $\max_k T_k > \Gamma$ **do**
    receive update $\Delta\mathbf{v}_k$ from some worker k;
    $\mathcal{P} \leftarrow \mathcal{P} \cup \{k\}$; $T_k \leftarrow 1$;
  $\mathcal{P}_S^{(t)} \leftarrow S$ workers in $\mathcal{P}$ with oldest updates;
  $\mathbf{v}^{(t+1)} \leftarrow \mathbf{v}^{(t)} + \nu \sum_{k \in \mathcal{P}_S^{(t)}} \Delta\mathbf{v}_k$; $\mathcal{P} \leftarrow \mathcal{P} \setminus \mathcal{P}_S^{(t)}$;
  foreach $k \notin \mathcal{P}_S^{(t)}$ **do** $T_k \leftarrow T_k + 1$;
  broadcast $\mathbf{v}^{(t+1)}$ to all workers in $\mathcal{P}_S^{(t)}$;

3.2. Merging updates from workers by master

If the master had to wait for the updates from all the workers, it could compute the global updates only after the slowest worker finished. To avoid this problem, we use *bounded barrier*: in each round t, the master waits for updates from only a subset $\mathcal{P}_S$ of workers of size $S \leq K$, and sends them back the global update $\mathbf{v}^{(t+1)} = \mathbf{v}^{(t)} + \nu \sum_{k \in \mathcal{P}_S^{(t)}} \Delta\mathbf{v}_k^{(t)}$. However, due to this relaxation, there might be some slow workers with out-of-date $\mathbf{v}$. When updates from such workers are merged by the master, it may degrade the quality of the global solution and hence may cause slow convergence or even divergence. We ensure sufficient freshness of the updates using *bounded delay*: the master makes sure that no worker has a stale update older than Γ rounds. This asynchronous approach has two benefits: (1) the overall progress is no more bottlenecked by the slowest processor, and (2) the total number of communications is reduced. On the flip side, convergence may get slowed down for very small S or very large Γ.

Example. Fig. 2 shows a possible sequence of important events in a run of our algorithm on a dataset having $n = 12$ data points in $d = 3$ dimensions using $K = 3$ nodes each having $R = 2$ cores such that each core works with only $|\mathcal{I}_{k,r}| = 2$ data points. The activities in solving the subproblem using $H = 1$ local iterations in a round are shown in a rectangular box. For the first subproblem, core 1 and core 2 in worker 1 randomly select dual coordinates such that the corresponding data points have nonzero entries in the dimensions $\{1, 3\}$ and $\{1, 2, 3\}$, respectively. Each core first reads the entries of $\mathbf{v}$ corresponding to these nonzero data dimensions, and then computes the updates $[0.1, 0, 0.7]$ and $[0.15, 0.5, 0.4]$, respectively, and finally applies these updates to $\mathbf{v}$ (where v_1 is first updated to be $0.1 + 0.15 = 0.25$ from both of the cores, then v_2 is updated to 0.5 from core 2 while v_3 is updated to 0.7 from core 1, and then v_3 is augmented by 0.4 from core 2 to reach $0.7 + 0.4 = 1.1$). The atomic memory updates ensure that all the conflicting writes to $\mathbf{v}$, such as v_1 in the first write-cycle, happen completely. At the end of H local iterations by each core, worker 1 sends $\Delta\mathbf{v} = [0.25, 0.5, 1.1]$ to the master, the responsibility of which is shared by one of the 3 nodes, but shown separately in the figure. By this time, the faster workers 2 and 3 already complete 3 rounds. As $S = 2$, the master takes first 2 updates from $\mathcal{P}_S^{(1)} = \mathcal{P}_S^{(2)} = \{2, 3\}$ and computes the global updates using $\nu = 1$. However, as $\Gamma = 2$, the master holds back the third updates from workers 2, 3 until the first update from worker 1 reaches the master. The subsequent events in the run are omitted in the figure.

3.3. Communication cost analysis

In each communication round, the algorithms based on synchronous updates on all K nodes require $2K$ transmissions, each consisting of all values of $\mathbf{v}$ or $\Delta\mathbf{v}$. Half of these transmissions are from the workers to the master and the rest are from the master to the workers. Whereas, our asynchronous update scheme requires $2S$ transmissions in each round.

4. Convergence analysis

In this section we prove the convergence of the global solution computed by our hybrid algorithm. We prove for the case of regularizer $g(\mathbf{w}) = \|\mathbf{w}\|^2$ as an example; the proof can be similarly extended to other regularizers $g(\mathbf{w})$. For this special case, $g^*(\mathbf{v}) = \|\mathbf{v}\|^2$, $\nabla g^*(\mathbf{v}) = 2\mathbf{v}$, the simplified dual formulation is the following

$$\max_{\boldsymbol{\alpha} \in \mathbb{R}^n} D(\boldsymbol{\alpha}) := -\frac{1}{n}\sum_{i=1}^{n} \phi^*(-\alpha_i) - \frac{\lambda}{2}\left\|\left(\frac{1}{\lambda n}\mathbf{X}\boldsymbol{\alpha}\right)\right\|^2, \tag{7}$$

and the corresponding subproblem formulation is the following

$$\max_{\boldsymbol{\delta}_{[k]} \in \mathbb{R}^n} D_k(\boldsymbol{\delta}_{[k]}; \mathbf{v}, \boldsymbol{\alpha}_{[k]}) := -\frac{1}{n_k}\sum_{i \in \mathcal{I}_k} \phi^*(-\alpha_i - \delta_i) - \frac{\lambda}{2S}\|\mathbf{v}\|^2 - \langle \frac{1}{2n}\mathbf{X}_{[k]}^\top \mathbf{v}, \boldsymbol{\delta}_{[k]} \rangle - \frac{\lambda\sigma}{2}\left\|\frac{1}{\lambda n}\mathbf{X}_{[k]}\boldsymbol{\delta}_{[k]}\right\|^2. \tag{8}$$

The analysis is divided into two parts. First we show that the solution of the subproblem computed by each node locally is indeed not far from the optimum of the subproblem. Using this result on the subproblem, we next show the convergence of the global solution. Though our proofs for the two parts are based on the works [7] and [11], respectively, we need to make significant adjustments in the proofs due to our modified framework handling two cascaded levels of asynchronous updates.

In our analysis we focus on all the events that are important for the local updates that are merged by the master in global update t. Fig. 3 shows an example where the master merges local updates from $S = 2$ workers.

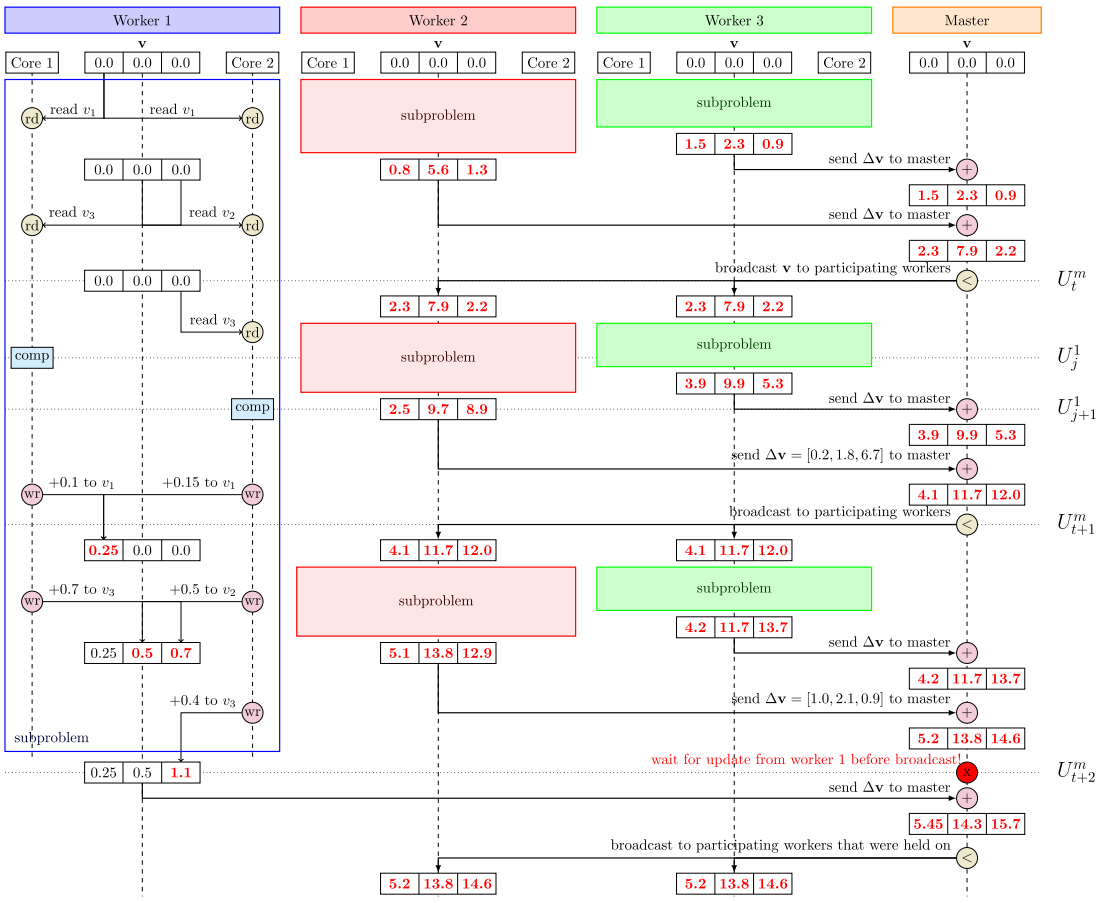


Fig. 2. Sequence of important events in an example run of Hybrid-DCA where $n = 12, d = 3, K = 3, R = 2, S = 2, \Gamma = 2, \nu = 1$.

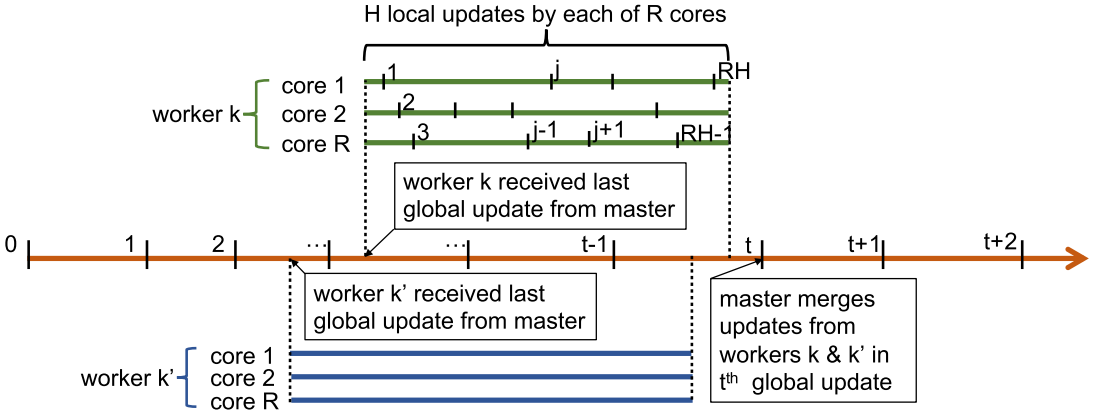


Fig. 3. Timeline of important events for the local updates from two workers that are merged by master at global update t.

4.1. Near optimality of the solution to the local subproblem

In this section we prove that the solution returned by the parallel asynchronous stochastic DCA solver used by each worker k in Algorithm 1 is not far from the optimal solution for the subproblem (4).

Definition 1 (Θ-*approximate*)**.** For given $\mathbf{v}, \boldsymbol{\alpha}_{[k]}$, a solution $\boldsymbol{\delta}_{[k]}$ to the subproblem (4) is said to be Θ-approximate, $\Theta \in [0, 1)$, if

$$\mathbb{E}\left[D_k(\boldsymbol{\delta}^*_{[k]}; \mathbf{v}, \boldsymbol{\alpha}_{[k]}) - D_k(\boldsymbol{\delta}_{[k]}; \mathbf{v}, \boldsymbol{\alpha}_{[k]})\right] \leq \Theta\left(D_k(\boldsymbol{\delta}^*_{[k]}; \mathbf{v}, \boldsymbol{\alpha}_{[k]}) - D_k(\mathbf{0}; \mathbf{v}, \boldsymbol{\alpha}_{[k]})\right) \tag{9}$$

where $\boldsymbol{\delta}^*_{[k]}$ is an optimum solution to (4).

Though our proof is based on the results in [7], the main challenge here is to tackle the following two modifications in our approach: (1) the solver here solves only a part of the dual problem and (2) the subproblem is now perturbed (Section 3). While the first modification is simply handled by considering the updates by the cores in worker k only, the second modification needs changes in each step of the proof in [7]. We give below the complete details of each of the steps.

Worker k solves subproblem (4) by applying total $R \times H$ updates, each of its R cores makes H updates. To show Θ-approximate, we need to show that sufficient progress is made between any two successive updates. However, each of the cores makes multiple atomic memory writes in an update, the updates

made by different cores are interleaved and hence it is difficult to demarcate two successive updates. Nevertheless, depending upon the order cores select a data point $\in \mathcal{I}_k$ as in step 6 of Algorithm 1 we assign a node-level counter j for each of the total $R \times H$ updates in node k and let the index $i(j)$ denote the data point selected for update $j \in \{1, \ldots, RH\}$. Fig. 3 shows an example of local updates $i(j)$.

In each update, a worker core computes step size ε using line 6 of Algorithm 1 and then applies ε in $i(j)$th axis to $\alpha_{[k]}$. However, there are a few subtle points to notice that happen due to the atomic updates. Firstly, when a core computes ε it starts with a v but by the time it reads a coordinate of v some other core might have already modified some other coordinates. So the effective v that a core uses to compute the increment ε might not be the actual v at the memory, and in fact it might not exist at all in the memory at any time. Let $\bar{v}$ denote the effective v that a core uses to compute ε, and let $\hat{v}$ denote the actual v value in the memory. Fig. 4 helps readers connect the different notation and updates used in the proofs in this section.

For all $i \in \mathcal{I}_k$, we have the following definitions:

$$
\begin{aligned}
h_i(u) &:= \frac{\phi_i^*(-u)}{n\left\|\mathbf{x}_i\right\|^2} + \frac{\lambda}{2}\left(\frac{1}{S} - \frac{1}{\sigma}\right)\frac{\|\mathbf{w}\|^2}{\left\|\mathbf{x}_i\right\|^2} \\
\operatorname{prox}_i(s) &:= \operatorname*{argmin}_u \frac{1}{2}(u-s)^2 + h_i(u) \\
T_i(\mathbf{w}, s) &:= \operatorname*{argmax}_u -\frac{1}{\sigma}\frac{\lambda}{2}\|\mathbf{w}\|^2 - \frac{1}{n}\mathbf{w}^\top \mathbf{x}_i(u-s) \\
&\quad - \frac{\lambda}{2}\sigma\left(\frac{1}{\lambda n}\mathbf{x}_i(u-s)\right)^2 \\
&\quad - \frac{1}{n}\phi_i^*(-u) - \frac{\lambda}{2}\left(\frac{1}{K}\|\mathbf{w}\|^2 - \frac{1}{\sigma}\|\mathbf{w}\|^2\right) \\
&= \operatorname*{argmax}_u -\frac{\lambda}{2}\left\|\frac{\mathbf{w}}{\sqrt{\sigma}} + \frac{\sqrt{\sigma}}{\lambda n}(u-s)\mathbf{x}_i\right\|^2 - \|\mathbf{x}_i\|^2 h_i(u) \\
&= \operatorname*{argmin}_u \frac{1}{2}\left(u - \left(s - \frac{\lambda n \mathbf{w}^\top \mathbf{x}_i}{\sigma\|\mathbf{x}_i\|^2}\right)\right)^2 + h_i(u),
\end{aligned}
$$

where $\mathbf{w} \in \mathbb{R}^d$ denotes any fixed vector, $s \in \mathbb{R}$, and $\operatorname{prox}(s)$ denotes the proximal operator. We can see the connection of the above operator to the proximal operator: $T_i(\mathbf{w}, s) = \operatorname{prox}_i\left(s - \frac{\mathbf{w}^\top \mathbf{x}_i}{\sigma\|\mathbf{x}_i\|^2}\right)$. Here both $h_i(u)$ and $T_i(\mathbf{w}, s)$ were revised from [7] to satisfy the subproblem (4).

Let $\bar{\mathbf{X}}$ denote the normalized data matrix at kth *local atomic solver* with omitted notation $_{[k]}$ where each row is $\bar{\mathbf{x}}_i^\top = \mathbf{x}_i^\top / \|\mathbf{x}_i\|$, $i \in \mathcal{I}_k$. Define $M_{[k],i} = \max_{\mathcal{D} \subseteq [d]} \| \sum_{t \in \mathcal{D}} \bar{\mathbf{X}}_{(:,t)} X_{(i,t)} \|$, $M = \max_k \max_i M_{[k],i}$ over all the *local atomic solvers*, where $[d]$ is the set of all the feature indices, and $\bar{\mathbf{X}}_{(:,t)}$ is the tth column of $\bar{\mathbf{X}}$. Moreover, R_{min} is defined as the minimum value of global data matrix, *i.e.*, $R_{min} = \min_{i=1,\ldots,n} \|\mathbf{x}_i\|^2$. Then, we define that:

Definition 2 (*Local Atomic Dual Variables*)**.** Here we omit $_{[k]}$ in the proofs of the *local atomic solver*.

$$
\begin{aligned}
\beta_t^{l+1} &= \begin{cases} T_t\left(\hat{\mathbf{w}}^l, \beta_c^l\right) & \text{if } c = i(l), \\ \beta_c^l & \text{if } c \neq i(l), \end{cases} & \varepsilon^l &= \beta_{i(l)}^{l+1} - \beta_{i(l)}^l, \\
\tilde{\beta}^{l+1} &= T\left(\hat{\mathbf{w}}^l, \boldsymbol{\beta}^l\right), & \bar{\beta}^{l+1} &= T\left(\bar{\mathbf{w}}^l, \boldsymbol{\beta}^l\right),
\end{aligned}
$$

where $\boldsymbol{\beta}^l = \boldsymbol{\alpha}_{[k]} + v\boldsymbol{\delta}_{[k]}^l$ denotes the lth sequence generated by a specific kth *local atomic solver*, $\hat{\mathbf{w}}^l$ denotes the actual values of $\mathbf{w}$ maintained at update l in the *local atomic solver*; $i(l)$ indicates the index selected at lth update; and $\bar{\mathbf{w}}^l$ refers to the "accurate" $\mathbf{w}$ if all cores are synchronously updated at iteration l. Note that, $\tilde{\beta}_{i(l)}^{l+1} = \beta_{i(l)}^{l+1}$ and $\bar{\beta}^{l+1} = \operatorname{prox}\left(\boldsymbol{\beta}^l - \frac{\lambda n}{\sigma}\bar{\mathbf{X}}\hat{\mathbf{w}}^l\right)$. Since $\mathbf{v}$ and $\boldsymbol{\alpha}_{[k]}$ will not be changed when solving the local subproblem $D_k(\boldsymbol{\delta}_{[k]}; \mathbf{v}, \boldsymbol{\alpha}_{[k]})$, we denote $Q^\sigma(\boldsymbol{\beta}^l)$ as the objective value of the dual subproblem at lth update and omit the subscripts $_{[k]}$.

Assumption 1 (*Lipschitz Continuous*)**.** The global problem objective (2) is L_{max}-Lipschitz continuous and therefore, its local subproblems objective (4) are at most L_{max}-Lipschitz continuous.

The following propositions are cited from [7], and we use these results in our proof.

Proposition 1 (*Expectation of Dual Variables*)**.**

$$
\mathbb{E}_{i(l)}\left(\left\|\boldsymbol{\beta}^{l+1} - \boldsymbol{\beta}^l\right\|^2\right) = \frac{1}{n}\left\|\tilde{\boldsymbol{\beta}}^{l+1} - \boldsymbol{\beta}^l\right\|^2, \tag{10}
$$

Proposition 2 (*Boundary of Asynchronous Variables*)**.**

$$
\left\|\bar{\mathbf{X}}\bar{\mathbf{w}}^l - \bar{\mathbf{X}}\hat{\mathbf{w}}^l\right\| \leq \frac{1}{\lambda n} M \sum_{c=l-\gamma}^{l} |\varepsilon^c|, \tag{11}
$$

Proposition 3.

$$
|T_i(\mathbf{w}_1, s_1) - T_i(\mathbf{w}_2, s_2)| \leq \left| s_1 - s2 + \frac{(\mathbf{w}_1 - \mathbf{w}_2)^\top \mathbf{x}_i}{\|\mathbf{x}_i\|^2} \right|, \tag{12}
$$

Proposition 4. *Let* $M \geq 1$, $q = \frac{6(\gamma+1)eM}{\sqrt{n}}$, $\rho = (1+q)^2$, *and* $\theta = \sum_{t=1}^{\gamma} \rho^{l/2}$. *If* $q(\gamma+1) \leq 1$ *and* $\sigma \geq 1$, *then* $\rho^{(\gamma+1)/2} \leq e$, *and*

$$
\rho^{-1} \leq 1 - \frac{4}{\sqrt{n}} - \frac{4M + 4M\theta}{\sqrt{n}} \leq 1 - \frac{4}{\sqrt{n}} - \frac{4M + 4M\theta}{\sigma\sqrt{n}}, \tag{13}
$$

Proposition 5 (*Properties of Dual Concave Function*)**.** *For all* $j > 0$, *we have*

$$
Q^\sigma\left(\boldsymbol{\beta}^l\right) \leq Q^\sigma\left(\bar{\boldsymbol{\beta}}^{l+1}\right) - \frac{\sigma\left\|\mathbf{x}_{i(l)}\right\|^2}{2}\left\|\boldsymbol{\beta}^l - \bar{\boldsymbol{\beta}}^{l+1}\right\|^2, \tag{14}
$$

$$
Q^\sigma\left(\boldsymbol{\beta}^l\right) \geq Q^\sigma\left(\bar{\boldsymbol{\beta}}^{l+1}\right) - \frac{L_{max}}{2}\left\|\boldsymbol{\beta}^l - \bar{\boldsymbol{\beta}}^{l+1}\right\|^2 \tag{15}
$$

Proof. Two properties of dual concave function are stated as follows.

- the strong convexity of $Q^\sigma(\boldsymbol{\beta}^l)$: as all conjugate functions are convex, it is clear that $Q^\sigma(\boldsymbol{\beta}^l)$ is $\sigma\left\|\mathbf{x}_{i(l)}\right\|^2$-strongly convex.
- the Lipschitz continuous gradient of $Q^\sigma(\boldsymbol{\beta}^l)$: refer to Assumption 1. □

Because of the atomic updates, the step size computation may not include all the latest updates, but we assume all the updates before the $(l-\gamma)$-th update have already been written into $\mathbf{v}$.

Assumption 2 (*Bounded Delay of Local Updates,* γ)**.**

$$
(\gamma+1)^2 \leq \frac{\sqrt{n_k}}{6eM}, \text{ where } e \text{ is the Euler's number.} \tag{16}
$$

This assumption restricts the maximum allowed local delay γ by M and n_k.

Lemma 6. *Under Assumption 2, Definition 2, and* $\rho = \left(1 + \frac{6(\gamma+1)eM}{\sqrt{n_k}}\right)^2$. *Then, the local subproblem satisfies:*

$$
\mathbb{E}\left[\left\|\boldsymbol{\beta}_{[k]}^{l-1} - \tilde{\boldsymbol{\beta}}_{[k]}^{l}\right\|^2\right] \leq \rho\mathbb{E}\left[\left\|\boldsymbol{\beta}_{[k]}^{l} - \tilde{\boldsymbol{\beta}}_{[k]}^{l+1}\right\|^2\right]. \tag{17}
$$

Not that $l \neq h$, *represents the* l*th update to* ω *in a local solver but not the* h*th iteration of one core.*

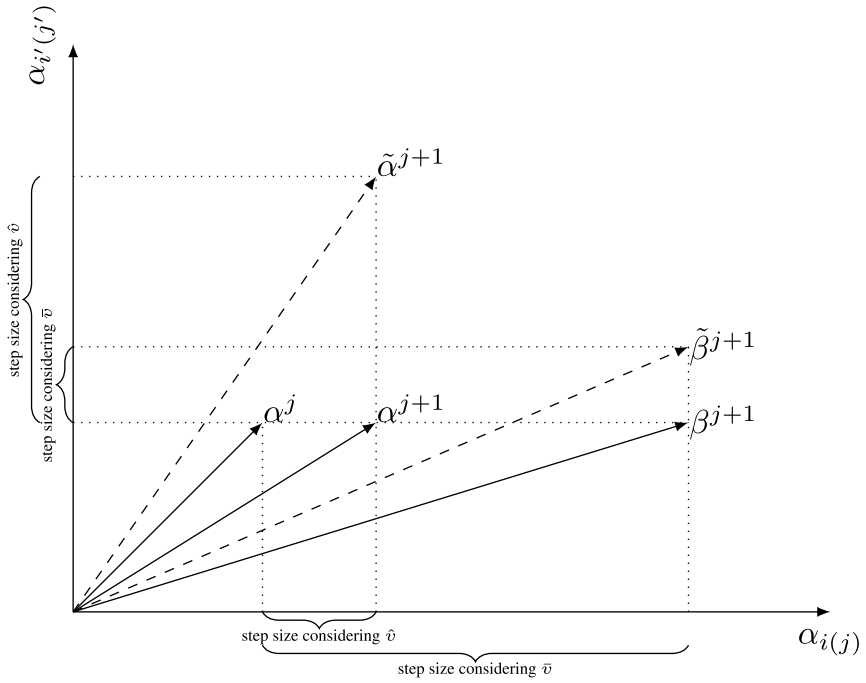


Fig. 4. Relationship among different approximations of α.

Proof. We omit the subscript $_{[k]}$ in the notations, which specifies the kth data partition, in the proof. We prove Eq. (17) by induction. As shown in [7], we have

$$\left\|\boldsymbol{\beta}^{l-1}-\tilde{\boldsymbol{\beta}}^{l}\right\|^2-\left\|\boldsymbol{\beta}^{l}-\tilde{\boldsymbol{\beta}}^{l+1}\right\|^2 \le 2\left\|\boldsymbol{\beta}^{l-1}-\tilde{\boldsymbol{\beta}}^{l}\right\|\left\|\boldsymbol{\beta}^{l}-\tilde{\boldsymbol{\beta}}^{l+1}-\boldsymbol{\beta}^{l-1}+\tilde{\boldsymbol{\beta}}^{l}\right\|. \tag{18}$$

The second factor on the r.h.s. of Eq. (18) is bounded as follows with the revisions:

$$\begin{aligned}
&\left\|\boldsymbol{\beta}^{l}-\tilde{\boldsymbol{\beta}}^{l+1}-\boldsymbol{\beta}^{l-1}+\tilde{\boldsymbol{\beta}}^{l}\right\| \\
\le&\left\|\boldsymbol{\beta}^{l}-\boldsymbol{\beta}^{l-1}\right\| \\
&+\left\|\operatorname{prox}\left(\boldsymbol{\beta}^{l}-\frac{\lambda n}{\sigma}\bar{\mathbf{X}}\hat{\mathbf{w}}^{l}\right)-\operatorname{prox}\left(\boldsymbol{\beta}^{l-1}-\frac{\lambda n}{\sigma}\bar{\mathbf{X}}\hat{\mathbf{w}}^{l-1}\right)\right\| \\
\le&\left\|\boldsymbol{\beta}^{l}-\boldsymbol{\beta}^{l-1}\right\|+\left\|\left(\boldsymbol{\beta}^{l}-\frac{\lambda n}{\sigma}\bar{\mathbf{X}}\hat{\mathbf{w}}^{l}\right)-\left(\boldsymbol{\beta}^{l-1}-\frac{\lambda n}{\sigma}\bar{\mathbf{X}}\hat{\mathbf{w}}^{l-1}\right)\right\| \\
\le&2\left\|\boldsymbol{\beta}^{l}-\boldsymbol{\beta}^{l-1}\right\|+\frac{\lambda n}{\sigma}\left\|\bar{\mathbf{X}}\hat{\mathbf{w}}^{l}-\bar{\mathbf{X}}\hat{\mathbf{w}}^{l-1}\right\| \\
=&2\left\|\boldsymbol{\beta}^{l}-\boldsymbol{\beta}^{l-1}\right\| \\
&+\frac{\lambda n}{\sigma}\left\|\bar{\mathbf{X}}\hat{\mathbf{w}}^{l}-\bar{\mathbf{X}}\bar{\mathbf{w}}^{l}+\bar{\mathbf{X}}\bar{\mathbf{w}}^{l}-\bar{\mathbf{X}}\bar{\mathbf{w}}^{l-1}+\bar{\mathbf{X}}\bar{\mathbf{w}}^{l-1}-\bar{\mathbf{X}}\hat{\mathbf{w}}^{l-1}\right\| \\
\le&2\left\|\boldsymbol{\beta}^{l}-\boldsymbol{\beta}^{l-1}\right\|+\frac{\lambda n}{\sigma}\left(\left\|\bar{\mathbf{X}}\bar{\mathbf{w}}^{l}-\bar{\mathbf{X}}\bar{\mathbf{w}}^{l-1}\right\|+\left\|\bar{\mathbf{X}}\hat{\mathbf{w}}^{l}-\bar{\mathbf{X}}\bar{\mathbf{w}}^{l}\right\|\right. \\
&\left.+\left\|\bar{\mathbf{X}}\bar{\mathbf{w}}^{l-1}-\bar{\mathbf{X}}\hat{\mathbf{w}}^{l-1}\right\|\right) \\
\le&\left(2+2\frac{\lambda n}{\sigma}\frac{M}{\lambda n}\right)\left\|\boldsymbol{\beta}^{l}-\boldsymbol{\beta}^{l-1}\right\| \\
&+2\frac{\lambda n}{\sigma}\frac{M}{\lambda n}\sum_{c=l-\gamma-1}^{l-2}|\varepsilon^{c}| \quad \text{(Proposition 2)} \qquad (19)\\
\le&\left(2+2\frac{M}{\sigma}\right)\left\|\boldsymbol{\beta}^{l}-\boldsymbol{\beta}^{l-1}\right\|+2\frac{M}{\sigma}\sum_{c=l-\gamma-1}^{l-2}|\varepsilon^{c}| \qquad (20)
\end{aligned}$$

Now we start the induction. Although some steps may be the same as the steps in [7], we still keep them here to make the proof self-contained.

Induction Hypothesis. We prove the following equivalent statement. For all j,

$$\mathbb{E}\left(\left\|\boldsymbol{\beta}^{l-1}-\tilde{\boldsymbol{\beta}}^{l}\right\|^2\right)\le\rho\mathbb{E}\left(\left\|\boldsymbol{\beta}^{l}-\tilde{\boldsymbol{\beta}}^{l+1}\right\|^2\right),$$

Induction Basis. When $l=1$,

$$\begin{aligned}
&\mathbb{E}\left(\left\|\boldsymbol{\beta}^{0}-\tilde{\boldsymbol{\beta}}^{1}\right\|^2\right)-\mathbb{E}\left(\left\|\boldsymbol{\beta}^{1}-\tilde{\boldsymbol{\beta}}^{2}\right\|^2\right) \\
\le&2E\left(\left\|\boldsymbol{\beta}^{0}-\tilde{\boldsymbol{\beta}}^{1}\right\|\left\|\boldsymbol{\beta}^{1}-\tilde{\boldsymbol{\beta}}^{2}-\boldsymbol{\beta}^{0}+\tilde{\boldsymbol{\beta}}^{1}\right\|\right) \\
\le&\left(4+4\frac{M}{2}\right)\mathbb{E}(\left\|\boldsymbol{\beta}^{0}-\tilde{\boldsymbol{\beta}}^{1}\right\|\left\|\boldsymbol{\beta}^{0}-\boldsymbol{\beta}^{1}\right\|).
\end{aligned}$$

By Proposition 1 and AM–GM inequality, which for any $b_1, b_2 > 0$ and any $c > 0$, we have

$$b_1 b_2 \le \frac{1}{2}\left(cb_1^2+c^{-1}b_2^2\right) \tag{21}$$

Therefore, we have

$$\begin{aligned}
&\mathbb{E}\left(\left\|\boldsymbol{\beta}^{0}-\tilde{\boldsymbol{\beta}}^{1}\right\|\left\|\boldsymbol{\beta}^{0}-\boldsymbol{\beta}^{1}\right\|\right) \\
\le&\frac{1}{2}\mathbb{E}\left(\sqrt{n}\left\|\boldsymbol{\beta}^{0}-\boldsymbol{\beta}^{1}\right\|^2+\frac{1}{\sqrt{n}}\left\|\boldsymbol{\beta}^{0}-\tilde{\boldsymbol{\beta}}^{1}\right\|^2\right) \\
=&\frac{1}{2}\mathbb{E}\left(\frac{1}{\sqrt{n}}\left\|\boldsymbol{\beta}^{0}-\tilde{\boldsymbol{\beta}}^{1}\right\|^2+\frac{1}{\sqrt{n}}\left\|\boldsymbol{\beta}^{0}-\tilde{\boldsymbol{\beta}}^{1}\right\|^2\right) \quad \text{(Proposition 1)} \\
=&\frac{1}{\sqrt{n}}\mathbb{E}\left(\left\|\boldsymbol{\beta}^{0}-\tilde{\boldsymbol{\beta}}^{1}\right\|^2\right)
\end{aligned}$$

Therefore,

$$\begin{aligned}
&\mathbb{E}\left(\left\|\boldsymbol{\beta}^{0}-\tilde{\boldsymbol{\beta}}^{1}\right\|^2\right)-\mathbb{E}\left(\left\|\boldsymbol{\beta}^{1}-\tilde{\boldsymbol{\beta}}^{2}\right\|^2\right) \\
\le&\left(\frac{4}{\sqrt{n}}+\frac{4M}{\sigma\sqrt{n}}\right)\mathbb{E}\left(\left\|\boldsymbol{\beta}^{0}-\tilde{\boldsymbol{\beta}}^{1}\right\|^2\right),
\end{aligned}$$

which implies

$$\mathbb{E}\left(\left\|\boldsymbol{\beta}^0-\tilde{\boldsymbol{\beta}}^1\right\|^2\right)\le\left(1-\frac{4}{\sqrt{n}}-\frac{4M}{\sigma\sqrt{n}}\right)^{-1}\mathbb{E}\left(\left\|\boldsymbol{\beta}^1-\tilde{\boldsymbol{\beta}}^2\right\|^2\right)\le\rho\mathbb{E}\left(\left\|\boldsymbol{\beta}^1-\tilde{\boldsymbol{\beta}}^2\right\|^2\right),$$

where the last inequality is based on Proposition 4 and the fact $\theta M\ge 1$.

Induction Step. By the induction hypothesis, we assume

$$\mathbb{E}\left(\left\|\boldsymbol{\beta}^{h-1}-\tilde{\boldsymbol{\beta}}^h\right\|^2\right)\le\rho\mathbb{E}\left(\left\|\boldsymbol{\beta}^h-\tilde{\boldsymbol{\beta}}^{h+1}\right\|^2\right)\ \forall h\le l-1. \tag{22}$$

To show

$$\mathbb{E}\left(\left\|\boldsymbol{\beta}^{l-1}-\tilde{\boldsymbol{\beta}}^l\right\|^2\right)\le\rho\mathbb{E}\left(\left\|\boldsymbol{\beta}^l-\tilde{\boldsymbol{\beta}}^{l+1}\right\|^2\right),$$

we firstly show that for all $h<l$,

$$\begin{aligned}
&\mathbb{E}\left(\left\|\boldsymbol{\beta}^h-\boldsymbol{\beta}^{h+1}\right\|\left\|\boldsymbol{\beta}^{l-1}-\tilde{\boldsymbol{\beta}}^l\right\|\right)\\
\le&\frac{1}{2}\mathbb{E}\left(\sqrt{n}\rho^{(h+1-l)/2}\left\|\boldsymbol{\beta}^h-\boldsymbol{\beta}^{h+1}\right\|^2\right.\\
&\left.+\frac{1}{\sqrt{n}}\rho^{(l-1-h)/2}\left\|\boldsymbol{\beta}^{l-1}-\tilde{\boldsymbol{\beta}}^l\right\|^2\right)\quad\text{(Eq. (21))}\\
=&\frac{1}{2}\mathbb{E}\left(\sqrt{n}\rho^{(h+1-l)/2}\mathbb{E}\left(\left\|\boldsymbol{\beta}^h-\boldsymbol{\beta}^{h+1}\right\|^2\right)\right.\\
&\left.+\frac{1}{\sqrt{n}}\rho^{(l-1-h)/2}\left\|\boldsymbol{\beta}^{l-1}-\tilde{\boldsymbol{\beta}}^l\right\|^2\right)\\
=&\frac{1}{2}\mathbb{E}\left(\frac{1}{\sqrt{n}}\rho^{(h+1-l)/2}\left\|\boldsymbol{\beta}^h-\tilde{\boldsymbol{\beta}}^{h+1}\right\|^2\right.\\
&\left.+\frac{1}{\sqrt{n}}\rho^{(l-1-h)/2}\left\|\boldsymbol{\beta}^{l-1}-\tilde{\boldsymbol{\beta}}^l\right\|^2\right)\quad\text{(Proposition 1)}\\
\le&\frac{1}{2}\mathbb{E}\left(\frac{1}{\sqrt{n}}\rho^{(h+1-l)/2}\rho^{l-c-1}\left\|\boldsymbol{\beta}^{l-1}-\tilde{\boldsymbol{\beta}}^l\right\|^2\right.\\
&\left.+\frac{1}{\sqrt{n}}\rho^{(l-1-h)/2}\left\|\boldsymbol{\beta}^{l-1}-\tilde{\boldsymbol{\beta}}^l\right\|^2\right)\quad\text{(Eq. (22))}\\
\le&\frac{\rho^{(l-1-h)/2}}{\sqrt{n}}\mathbb{E}\left(\left\|\boldsymbol{\beta}^{l-1}-\tilde{\boldsymbol{\beta}}^l\right\|^2\right).
\end{aligned}\tag{23}$$

Let $\theta=\sum_{h=1}^{\gamma}\rho^{h/2}$. We have

$$\begin{aligned}
&\mathbb{E}\left(\left\|\boldsymbol{\beta}^{l-1}-\tilde{\boldsymbol{\beta}}^l\right\|^2\right)-\mathbb{E}\left(\left\|\boldsymbol{\beta}^l-\tilde{\boldsymbol{\beta}}^{l+1}\right\|\right)\\
\le&\mathbb{E}\left(2\left\|\boldsymbol{\beta}^{l-1}-\tilde{\boldsymbol{\beta}}^l\right\|\left(\left(2+2\frac{M}{\sigma}\right)\left\|\boldsymbol{\beta}^{l-1}-\boldsymbol{\beta}^l\right\|\right.\right.\\
&\left.\left.+2\frac{M}{\sigma}\left\|\boldsymbol{\beta}^{h-1}-\boldsymbol{\beta}^h\right\|\right)\right)\quad\text{(Eqs. (18), (19))}\\
=&\left(4+4\frac{M}{\sigma}\right)\mathbb{E}\left(\left\|\boldsymbol{\beta}^{l-1}-\tilde{\boldsymbol{\beta}}^l\right\|\left\|\boldsymbol{\beta}^{l-1}-\boldsymbol{\beta}^l\right\|\right)\\
&+4\frac{M}{\sigma}\sum_{c=l-\gamma-1}^{l-1}\mathbb{E}\left(\left\|\boldsymbol{\beta}^{l-1}-\tilde{\boldsymbol{\beta}}^l\right\|\left\|\boldsymbol{\beta}^{c-1}-\boldsymbol{\beta}^c\right\|\right)\\
\le&\frac{4\sigma+4M}{\sigma\sqrt{n}}\mathbb{E}\left(\left\|\boldsymbol{\beta}^{l-1}-\tilde{\boldsymbol{\beta}}^l\right\|^2\right)\\
&+\frac{4M}{\sigma\sqrt{n}}\mathbb{E}\left(\left\|\boldsymbol{\beta}^{l-1}-\tilde{\boldsymbol{\beta}}^l\right\|\right)\sum_{c=l-\gamma-1}^{l-2}\rho^{(l-1-c)/2}\quad\text{(Eq. (23))}\\
\le&\frac{4\sigma+4M}{\sigma\sqrt{n}}\mathbb{E}\left(\left\|\boldsymbol{\beta}^{l-1}-\tilde{\boldsymbol{\beta}}^l\right\|^2\right)+\frac{4M}{\sigma\sqrt{n}}\theta\mathbb{E}\left(\left\|\boldsymbol{\beta}^{l-1}-\tilde{\boldsymbol{\beta}}^l\right\|\right)\\
\le&\left(\frac{4}{\sqrt{n}}+\frac{4M+4M\theta}{\sigma\sqrt{n}}\right)\mathbb{E}\left(\left\|\boldsymbol{\beta}^{l-1}-\tilde{\boldsymbol{\beta}}^l\right\|^2\right)
\end{aligned}$$

which implies that

$$\mathbb{E}\left(\left\|\boldsymbol{\beta}^{l-1}-\tilde{\boldsymbol{\beta}}^l\right\|^2\right)\le\frac{1}{1-\frac{4}{\sqrt{n}}-\frac{4M+4M\theta}{\sigma\sqrt{n}}}\mathbb{E}\left(\left\|\boldsymbol{\beta}^l-\tilde{\boldsymbol{\beta}}^{l+1}\right\|^2\right)\le\rho\mathbb{E}\left(\left\|\boldsymbol{\beta}^{l-1}-\tilde{\boldsymbol{\beta}}^l\right\|^2\right)$$

by Proposition 4. □

Lemma 6 implies that the asynchronous updates will not pull the solution away from the optimal solution too much even if the directions of the updates are wrong.

Definition 3 (*Global Error Bound*)**.** For a convex function $f:\mathbb{R}^n\to\mathbb{R}$, the optimization problem: $\min_{\boldsymbol{\beta}} f(\boldsymbol{\beta})$ admits a global error bound if there is a constant κ such that

$$\|\boldsymbol{\beta}-P_S(\boldsymbol{\beta})\|\le\kappa\|T(\boldsymbol{\beta})-\boldsymbol{\beta}\|, \tag{24}$$

where $P_S(\cdot)$ is the Euclidean projection to the set of optimal solutions, and $T:\mathbb{R}^n\to\mathbb{R}^n$ is the operator defined as

$$T_i(\boldsymbol{\beta})=\arg\min_u f\left(\boldsymbol{\beta}+(u-\beta_i)\mathbf{e}_h\right)\quad\forall i\in[n].$$

The optimization problem admits a relaxed condition called global error bound from the beginning if (24) holds for any $\boldsymbol{\beta}$ satisfying $f(\boldsymbol{\beta})\le F$ for some constant F.

Assumption 3. The local subproblem formulation (4) admits the global error bound from the beginning for $F=Q(\boldsymbol{\delta}_{[k]}^{(j)};\mathbf{v}^{(j)},\boldsymbol{\alpha}_{[k]}^{(j)})$ and any update j.

The global error bound in the local subproblem helps prove that our subproblem solver achieves significant improvement after each update. It has been shown that when the loss functions are hinge loss or squared hinge loss, the global problem formulation (2) does indeed satisfy the global error bound condition [7]. Then, for the local subproblem (4), it still satisfies the global error bound within the subset $\boldsymbol{\alpha}_{[k]}$.

Assumption 4 (*Bounded M, L_{max}*)**.**

$$2L_{max}\left(1+\frac{e^2\gamma^2M^2}{\sigma^2n_k}\right)\left(\frac{e^2\gamma^2M^2}{\sigma^2n_k}\right)\le 1$$

Lemma 7 (*Convergence for Subproblem*)**.** *When Assumptions 2–4 hold, the solutions computed in two successive updates by the local subproblem solver have a linear convergence rate in expectation,* i.e.,

$$\mathbb{E}\left[D_k(\boldsymbol{\delta}_{[k]}^*)-D_k(\boldsymbol{\delta}_{[k]}^{(j)})\right]\le\Theta\left[D_k(\boldsymbol{\delta}_{[k]}^*)-D_k(\boldsymbol{\delta}_{[k]}^{(j-1)})\right]$$

where $\boldsymbol{\delta}_{[k]}^{(j)}=\boldsymbol{\delta}_{[k]}^{(j)}+\sum_{h=1}^{H}\sum_{r=1}^{R}\boldsymbol{\beta}_{[k]}^{h,r}$ *is the* $\boldsymbol{\delta}_{[k]}$ *after the jth update,*

$$\eta=1-\frac{\kappa R_{min}}{2nL_{max}}\left(1-\frac{2L_{max}}{R_{min}}\left(1+\frac{e^2\gamma^2M^2}{\sigma^2\tilde{n}}\right)\left(\frac{e^2\gamma^2M^2}{\sigma^2\tilde{n}}\right)\right),$$

$\tilde{n}=\max_k n_k$ *is the size of the largest data part, and*

$$\Theta=\eta^{RH}. \tag{25}$$

Proof. We also omit the subscript $_{[k]}$ of the notations in the proof. We can bound the expected distance $\mathbb{E}\left(\left\|\tilde{\boldsymbol{\beta}}^{j+1}-\tilde{\boldsymbol{\beta}}^{l+1}\right\|^2\right)$ by the

following derivation.

$$
\begin{aligned}
&\mathbb{E}\left(\left\|\bar{\boldsymbol{\beta}}^{l+1}-\tilde{\boldsymbol{\beta}}^{l+1}\right\|^2\right)\\
&=\mathbb{E}\left(\sum_{t=1}^{n}\left(T_t(\bar{\mathbf{w}}^l,\beta_t^l)-T_t(\hat{\mathbf{w}}^l,\beta_t^l)\right)^2\right)\\
&\le\mathbb{E}\left(\sum_{t=1}^{n}\left(\frac{\lambda n\left(\bar{\mathbf{w}}^l-\hat{\mathbf{w}}^l\right)^\top\mathbf{x}_t}{\sigma\left\|\mathbf{x}_t\right\|^2}\right)^2\right)\quad\text{(Proposition 3)}\\
&=\frac{\lambda^2n^2}{\sigma^2}\mathbb{E}\left(\left\|\bar{\mathbf{X}}\left(\bar{\mathbf{w}}^l-\hat{\mathbf{w}}^l\right)\right\|^2\right)\\
&\le\frac{M^2}{\lambda^2n^2}\frac{\lambda^2n^2}{\sigma^2}\mathbb{E}\left(\left(\sum_{t=l-\gamma}^{l-1}\left\|\boldsymbol{\beta}^t-\boldsymbol{\beta}^{t+1}\right\|\right)^2\right)\quad\text{(Proposition 2)}\\
&\le\frac{M^2}{\sigma^2}\mathbb{E}\left(\gamma\left(\sum_{t=l-\gamma}^{l-1}\left\|\boldsymbol{\beta}^t-\boldsymbol{\beta}^{t+1}\right\|^2\right)\right)\quad\text{(Cauchy Schwarz Inequality)}\\
&\le\frac{\gamma M^2}{\sigma^2}\mathbb{E}\left(\gamma\left(\sum_{t=1}^{\gamma}\rho^t\left\|\boldsymbol{\beta}^l-\boldsymbol{\beta}^{l+1}\right\|^2\right)\right)\quad\text{(Lemma 6)}\\
&\le\frac{\gamma M^2}{\sigma^2n}\left(\sum_{t=1}^{\gamma}\rho^t\right)\mathbb{E}\left(\left\|\boldsymbol{\beta}^l-\tilde{\boldsymbol{\beta}}^{l+1}\right\|^2\right)\quad\text{(Proposition 1)}\\
&\le\frac{\gamma^2M^2}{\sigma^2n}\rho^\gamma\mathbb{E}\left(\left\|\boldsymbol{\beta}^l-\tilde{\boldsymbol{\beta}}^{l+1}\right\|^2\right)\\
&\le\frac{\gamma^2M^2e^2}{\sigma^2n}\mathbb{E}\left(\left\|\boldsymbol{\beta}^l-\tilde{\boldsymbol{\beta}}^{l+1}\right\|^2\right).\quad\text{(Proposition 4)}
\end{aligned}
\tag{26}
$$

Moreover,

$$
\begin{aligned}
&\mathbb{E}\left(\left\|\bar{\boldsymbol{\beta}}^{l}-\boldsymbol{\beta}^{l+1}\right\|^2\right)\\
&=\mathbb{E}\left(\left\|\bar{\boldsymbol{\beta}}^{l+1}-\tilde{\boldsymbol{\beta}}^{l+1}+\tilde{\boldsymbol{\beta}}^{l+1}-\boldsymbol{\beta}^{l}\right\|^2\right)\\
&\le\mathbb{E}\left(2\left(\left\|\bar{\boldsymbol{\beta}}^{l+1}-\tilde{\boldsymbol{\beta}}^{l+1}\right\|^2\right.\right.\\
&\quad\left.\left.+\left\|\tilde{\boldsymbol{\beta}}^{l+1}-\boldsymbol{\beta}^{l}\right\|^2\right)\right)\quad\text{(Cauchy–Schwarz)}\\
&\le2\left(1+\frac{\gamma^2M^2e^2}{\sigma^2n}\right)\mathbb{E}\left(\left\|\tilde{\boldsymbol{\beta}}^{l+1}-\boldsymbol{\beta}^{l}\right\|^2\right)
\end{aligned}
\tag{27}
$$

The bound of the increase of local objective function value by

$$
\begin{aligned}
&\mathbb{E}\left(Q^\sigma\left(\boldsymbol{\beta}^{l+1}\right)\right)-\mathbb{E}\left(Q^\sigma\left(\boldsymbol{\beta}^{l}\right)\right)\\
=&\mathbb{E}\left(-\left(Q^\sigma\left(\boldsymbol{\beta}^{l}\right)-Q^\sigma\left(\bar{\boldsymbol{\beta}}^{l+1}\right)\right)\right)\\
&-\mathbb{E}\left(\left(Q^\sigma\left(\bar{\boldsymbol{\beta}}^{l+1}\right)-Q^\sigma\left(\boldsymbol{\beta}^{l+1}\right)\right)\right)\\
\ge&\mathbb{E}\left(\frac{\sigma\left\|\mathbf{x}_{i(l)}\right\|^2}{2}\left\|\boldsymbol{\beta}^{l}-\bar{\boldsymbol{\beta}}^{l+1}\right\|^2\right)\\
&-\mathbb{E}\left(\frac{L_{max}}{2}\left\|\boldsymbol{\beta}^{l+1}-\bar{\boldsymbol{\beta}}^{l+1}\right\|^2\right)\quad\text{(Proposition 5)}\\
\ge&\frac{R_{min}}{2n}\mathbb{E}\left(\left\|\boldsymbol{\beta}^{l}-\bar{\boldsymbol{\beta}}^{l+1}\right\|^2\right)-\frac{L_{max}}{2n}\mathbb{E}\left(\left\|\bar{\boldsymbol{\beta}}^{l+1}-\bar{\boldsymbol{\beta}}^{l+1}\right\|^2\right)\\
\ge&\frac{R_{min}}{2n}\mathbb{E}\left(\left\|\boldsymbol{\beta}^{l}-\bar{\boldsymbol{\beta}}^{l+1}\right\|^2\right)\\
&-\frac{L_{max}}{2n}\frac{\gamma^2M^2e^2}{\sigma^2n}\mathbb{E}\left(\left\|\tilde{\boldsymbol{\beta}}^{l+1}-\boldsymbol{\beta}^{l}\right\|^2\right)\quad\text{(Eq. (26))}\\
\ge&\frac{R_{min}}{2n}\mathbb{E}\left(\left\|\boldsymbol{\beta}^{l}-\bar{\boldsymbol{\beta}}^{l+1}\right\|^2\right)\\
&-\frac{2L_{max}}{2n}\frac{\gamma^2M^2e^2}{\sigma^2n}\left(1+\frac{\gamma^2M^2e^2}{\sigma^2n}\right)\mathbb{E}\left(\left\|\bar{\boldsymbol{\beta}}^{l+1}-\boldsymbol{\beta}^{l}\right\|^2\right)\quad\text{(Eq. (27))}\\
\ge&\frac{R_{min}}{2n}\left(1-\frac{2L_{max}}{2n}\left(1+\frac{\gamma^2M^2e^2}{\sigma^2n}\right)\left(\frac{\gamma^2M^2e^2}{\sigma^2n}\right)\right)\\
&\times\ \mathbb{E}\left(\left\|\bar{\boldsymbol{\beta}}^{l+1}-\boldsymbol{\beta}^{l}\right\|^2\right)\\
\ge&\frac{\kappa R_{min}}{2n}\left(1-\frac{2L_{max}}{2n}\left(1+\frac{\gamma^2M^2e^2}{\sigma^2n}\right)\left(\frac{\gamma^2M^2e^2}{\sigma^2n}\right)\right)\\
&\times\ \mathbb{E}\left(\left\|P_S\left(\boldsymbol{\beta}^{l}\right)-\boldsymbol{\beta}^{l}\right\|^2\right)\\
\ge&\frac{\kappa R_{min}}{2nL_{max}}\left(1-\frac{2L_{max}}{2n}\left(1+\frac{\gamma^2M^2e^2}{\sigma^2n}\right)\left(\frac{\gamma^2M^2e^2}{\sigma^2n}\right)\right)\\
&\times\ \mathbb{E}\left(Q^{\sigma*}-Q^\sigma\left(\boldsymbol{\beta}^{l}\right)\right)
\end{aligned}
$$

Therefore,

$$
\begin{aligned}
&Q^{\sigma*}-\mathbb{E}\left(Q^\sigma\left(\boldsymbol{\beta}^{l+1}\right)\right)\\
&=Q^{\sigma*}-\mathbb{E}\left(Q^\sigma\left(\boldsymbol{\beta}^{l}\right)\right)-\left(\mathbb{E}\left(Q^\sigma\left(\boldsymbol{\beta}^{l+1}\right)-\mathbb{E}\left(Q^\sigma\left(\boldsymbol{\beta}^{l}\right)\right)\right)\right)\\
&\le\eta\left(Q^{\sigma*}-\mathbb{E}\left(Q^\sigma\left(\boldsymbol{\beta}^{l}\right)\right)\right)
\end{aligned}
$$

Let us assume that $\beta_{[k]}^*$ is the optimal solution of the subproblem (4) denoted as:

$$
\beta_{[k]}^*=\arg\max_{\beta_{[k]}\in\mathbb{R}^{n_k}}Q^\sigma(\beta_{[k]};\bar{\mathbf{w}}).
\tag{28}
$$

According to above proof of Lemma 7, the local atomic solver has a linear convergence rate in expectation, that is,

$$
\begin{aligned}
&Q^\sigma(\beta_{[k]}^*;\bar{\mathbf{w}})-\mathbb{E}\left(Q^\sigma(\beta_{[k]}^{j+1};\bar{\mathbf{w}})\right)\\
&\le\eta\left(\mathbb{E}\left(Q^\sigma(\beta_{[k]}^*;\bar{\mathbf{w}})-Q^\sigma(\beta_{[k]}^{j};\bar{\mathbf{w}})\right)\right)
\end{aligned}
$$

It is obvious that $\Theta=\eta^{RH}$. Thus, we can easily get the induction as

$$
\begin{aligned}
&Q^\sigma(\beta_{[k]}^*;\bar{\mathbf{w}})-\mathbb{E}\left(Q^\sigma(\beta_{[k]}^{RH};\bar{\mathbf{w}})\right)\\
&\le\eta\left(Q^\sigma(\beta_{[k]}^*;\bar{\mathbf{w}})-\mathbb{E}\left(Q^\sigma(\beta_{[k]}^{RH-1};\bar{\mathbf{w}})\right)\right)\\
&\le\eta^2\left(\mathbb{E}\left(Q^\sigma(\beta_{[k]}^*;\bar{\mathbf{w}})-Q^\sigma(\beta_{[k]}^{RH-2};\bar{\mathbf{w}})\right)\right)\\
&\le\cdots\le\Theta\left(Q^\sigma(\beta_{[k]}^*;\bar{\mathbf{w}})-\mathbb{E}\left(Q^\sigma(\beta_{[k]}^{0};\bar{\mathbf{w}})\right)\right).
\end{aligned}
$$

Notice that $\beta_{[k]}^0$ are the starting points of the local atomic solver and $\beta_{[k]}^{R,H}$ are the final results of $\beta_{[k]}$ of the local atomic solver. So the following equations hold for the global problem:

$$
\begin{aligned}
\beta_{[k]}^0&=\alpha_{[k]}\\
\beta_{[k]}^{R,H}-\beta_{[k]}^0&=\boldsymbol{\delta}_{[k]}\\
\beta_{[k]}^*-\beta_{[k]}^0&=\boldsymbol{\delta}_{[k]}^*
\end{aligned}
$$

Therefore, we have:

$$
\begin{aligned}
&\mathbb{E}\left[D_k(\boldsymbol{\delta}_{[k]}^*;\mathbf{v},\boldsymbol{\alpha}_{[k]})-D_k(\boldsymbol{\delta}_{[k]};\mathbf{v},\boldsymbol{\alpha}_{[k]})\right]\\
&\le\Theta\left[D_k(\boldsymbol{\delta}_{[k]}^*;\mathbf{v},\boldsymbol{\alpha}_{[k]})-D_k(\mathbf{0};\mathbf{v},\boldsymbol{\alpha}_{[k]})\right]
\end{aligned}
$$

with $\Theta=\eta^{RH}$. □

4.2. Convergence of global solution

Although we have shown that the local subproblem solver outputs a Θ-approximate solution, we cannot directly apply the results of [11] for the global solution because our algorithm uses updates from only a subset $S\le K$ of workers which is unlike the synchronous all-reduce of the updates from all workers used

in [11]. We need to handle this asynchronous nature of the global updates, just like we handled asynchronous updates for the local subproblem.

Let us consider the global updates in the order the master computed them (at global time t in Fig. 3). To prove convergence, it is customary to show that the global objective progresses sufficiently in each round, *i.e.,* there is sufficient change from $D(\boldsymbol{\alpha}^{(t)})$ to $D(\boldsymbol{\alpha}^{(t+1)})$ where $\boldsymbol{\alpha}^{(t)}$ denotes the value of the dual variable $\boldsymbol{\alpha}$ distributed as $\boldsymbol{\alpha}_{[k]}^{(t)}$ across all the workers k at the time master computed tth global update $\mathbf{v}^{(t)}$. For simplicity we assume that each worker updates $\boldsymbol{\alpha}_{[k]}^{(t)}$ in step 11 of Algorithm 1 as soon as it receives global update from the master. Thus, $\boldsymbol{\alpha}^{(t+1)} = \boldsymbol{\alpha}^{(t)} + \nu\boldsymbol{\delta}^{(t)}$ where $\boldsymbol{\delta}^{(t)} = \sum_k \boldsymbol{\delta}_{[k]}^{(t)}$ and $\boldsymbol{\delta}_{[k]}^{(t)}$ denotes the increment to $\boldsymbol{\alpha}_{[k]}^{(t)}$ computed by worker k if $k \in \mathcal{P}_S^{(t)}$, $\mathbf{0}$ otherwise.

If $k \in \mathcal{P}_S^{(t)}$ then the update $\boldsymbol{\delta}_{[k]}^{(t)}$ has already been included in $\mathbf{v}^{(t)}$. However, if $k \notin \mathcal{P}_S^{(t)}$ then it may not be included. Let ξ be such that for all $l \leq \xi$ and for all k, $\boldsymbol{\delta}_{[k]}^{(l)}$ has been included in $\mathbf{v}^{(t)}$. By the design of our algorithm, $t - \Gamma < \xi \leq t$. Let $\hat{\boldsymbol{\alpha}}^{(t)}$ be defined as follows: $\hat{\boldsymbol{\alpha}}_{[k]}^{(t)} = \boldsymbol{\alpha}_{[k]}^{(t)}, \forall k \in \mathcal{P}_S^{(t)}$ and $= \boldsymbol{\alpha}_{[k]}^{(\xi-1)}$ for the latest $(\xi-1)$ for which the update is already included in global $\mathbf{v}$, $\forall k \notin \mathcal{P}_S^{(t)}$. Let $\mathbf{w}^{(t)}, \hat{\mathbf{w}}^{(t)}$ be $\mathbf{w}(\boldsymbol{\alpha}^{(t)})$ and $\mathbf{w}(\hat{\boldsymbol{\alpha}}^{(t)})$ respectively. Note that $\mathbf{w}^{(t)} = \hat{\mathbf{w}}^{(t)} + \frac{\nu}{\lambda n}\sum_{l=\xi}^{t} \mathbf{X}\boldsymbol{\delta}^{(l)}$. For a vector expression (x), let $(x)_i$ represent the ith element of the vector resulting from the expression (x).

Lemma 8. *For any dual $\boldsymbol{\alpha}^{(t)}, \boldsymbol{\delta}^{(t)} \in \mathbb{R}^n$, primal $\hat{\mathbf{w}}^{(t)} = \mathbf{w}(\hat{\boldsymbol{\alpha}}^{(t)})$ and real values ν and σ satisfying (5), it holds that*

$$
\begin{aligned}
D\left(\boldsymbol{\alpha}^{(t+1)}\right) =& D\left(\boldsymbol{\alpha}^{(t)} + \nu \sum_{k\in\mathcal{P}_S^{(t)}} \boldsymbol{\delta}_{[k]}^{(t)}\right) \\
\geq& (1-\nu)D(\hat{\boldsymbol{\alpha}}) + \nu \sum_{k\in\mathcal{P}_S^{(t)}} D_k(\boldsymbol{\delta}_{[k]}^{(t)}; \boldsymbol{\alpha}_{[k]}, \hat{\mathbf{w}}) \\
&- \frac{\lambda}{2}\left(\frac{2\nu}{\lambda n}\sum_{k\notin\mathcal{P}_S^{(t)}} \mathbf{w}(\hat{\boldsymbol{\alpha}})^\top \mathbf{X}\left(\sum_{l=\xi}^{t}\boldsymbol{\delta}_{[k]}^{(l)}\right) + \left(\frac{\nu}{\lambda n}\right)^2 \left\|\sum_{k\notin\mathcal{P}_S^{(t)}}\sum_{c=\xi}^{t}\mathbf{X}\boldsymbol{\delta}_{[k]}^{(c)}\right\|^2\right) \\
&- \frac{2\nu^2}{\lambda n^2}\left(\sum_{k\notin\mathcal{P}_S^{(t)}}\sum_{l=\xi}^{t}\boldsymbol{\delta}_{[k]}^{(l)}\right)^\top \mathbf{X}^\top\mathbf{X}\sum_{k\in\mathcal{P}_S}\boldsymbol{\delta}_{[k]}^{(t)} \\
&- \frac{1}{n}\sum_{k\notin\mathcal{P}_S^{(t)}}\left(\sum_{i\in\mathcal{I}_k}\phi_i^*((-\hat{\boldsymbol{\alpha}} - \nu\sum_{c=\xi}^{t}\boldsymbol{\delta}_{[k]}^{(c)})_i)\right). \qquad (29)
\end{aligned}
$$

Proof. Assume that $\mathcal{I} = \bigcup_{k\in\mathcal{P}_S}\mathcal{I}_k$. Then, we have

$$
\begin{aligned}
& D\left(\boldsymbol{\alpha}^{(t)} + \nu\sum_{k\in\mathcal{P}_S^{(t)}}\boldsymbol{\delta}_{[k]}^{(t)}\right) \\
&= -\frac{1}{n}\sum_{i=1}^{n}\phi_i^*\left(-\hat{\alpha}_i - \nu\left(\sum_{k\notin\mathcal{P}_S^{(t)}}\sum_{c=\xi}^{t}\boldsymbol{\delta}_{[k]}^{(c)} + \sum_{k\in\mathcal{P}_S^{(t)}}\boldsymbol{\delta}_{[k]}^{(t)}\right)_i\right) \\
&- \frac{\lambda}{2}\left\|\frac{1}{\lambda n}\mathbf{X}\left(\hat{\boldsymbol{\alpha}} + \nu\sum_{k\notin\mathcal{P}_S^{(t)}}\sum_{c=\xi}^{t}\boldsymbol{\delta}_{[k]}^{(c)} + \nu\sum_{k\in\mathcal{P}_S^{(t)}}\boldsymbol{\delta}_{[k]}^{(t)}\right)\right\|^2 \\
&= -\frac{1}{n}\sum_{k\in\mathcal{P}_S^{(t)}}\left(\sum_{i\in\mathcal{I}_k}\phi_i^*(-(1-\nu)\hat{\alpha}_i - \nu(\hat{\boldsymbol{\alpha}} + \boldsymbol{\delta}_{[k]}^{(t)})_i)\right) \\
&- \frac{1}{n}\sum_{k\notin\mathcal{P}_S^{(t)}}\left(\sum_{i\in\mathcal{I}_k}\phi_i^*((-\hat{\boldsymbol{\alpha}} - \nu\sum_{c=\xi}^{t}\boldsymbol{\delta}_{[k]}^{(c)})_i)\right) \\
&- \frac{\lambda}{2}\left(\left\|\mathbf{w}(\hat{\boldsymbol{\alpha}})\right\|^2 + \frac{2\nu}{\lambda n}\sum_{k\in\mathcal{P}_S^{(t)}}\mathbf{w}(\hat{\boldsymbol{\alpha}})^\top\mathbf{X}\boldsymbol{\delta}_{[k]}^{(t)} + \left(\frac{\nu}{\lambda n}\right)^2\left\|\sum_{k\in\mathcal{P}_S^{(t)}}\mathbf{X}\boldsymbol{\delta}_{[k]}^{(t)}\right\|^2\right) \\
&- \frac{\lambda}{2}\left(\frac{2\nu}{\lambda n}\sum_{k\notin\mathcal{P}_S^{(t)}}\mathbf{w}(\hat{\boldsymbol{\alpha}})^\top\mathbf{X}\left(\sum_{l=\xi}^{t}\boldsymbol{\delta}_{[k]}^{(l)}\right) + \left(\frac{\nu}{\lambda n}\right)^2\left\|\sum_{k\notin\mathcal{P}_S^{(t)}}\sum_{c=\xi}^{t}\mathbf{X}\boldsymbol{\delta}_{[k]}^{(c)}\right\|^2\right) \\
&- \frac{2\nu^2}{\lambda n^2}\left(\sum_{k\notin\mathcal{P}_S^{(t)}}\sum_{l=\xi}^{t}\boldsymbol{\delta}_{[k]}^{(l)}\right)^\top\mathbf{X}^\top\mathbf{X}\sum_{k\in\mathcal{P}_S}\boldsymbol{\delta}_{[k]}^{(t)} \\
&\geq -\frac{1}{n}\sum_{k\in\mathcal{P}_S^{(t)}}\left(\sum_{i\in\mathcal{I}_k}((1-\nu)\phi_i^*(-\hat{\alpha}_i) + \nu\phi_i^*(-(\hat{\boldsymbol{\alpha}} + \boldsymbol{\delta}_{[k]}^{(t)})_i))\right) \\
&- \frac{\lambda}{2}\left(\left\|\hat{\mathbf{w}}\right\|^2 + \frac{2\nu}{\lambda n}\sum_{k\in\mathcal{P}_S^{(t)}}\hat{\mathbf{w}}^\top\mathbf{X}\boldsymbol{\delta}_{[k]}^{(t)} + \left(\frac{\nu}{\lambda n}\right)^2\left\|\sum_{k\in\mathcal{P}_S^{(t)}}\mathbf{X}\boldsymbol{\delta}_{[k]}^{(t)}\right\|^2\right) \\
&- \frac{\lambda}{2}\left(\frac{2\nu}{\lambda n}\sum_{k\notin\mathcal{P}_S^{(t)}}\mathbf{w}(\hat{\boldsymbol{\alpha}})^\top\mathbf{X}\left(\sum_{l=\xi}^{t}\boldsymbol{\delta}_{[k]}^{(l)}\right) + \left(\frac{\nu}{\lambda n}\right)^2\left\|\sum_{k\notin\mathcal{P}_S^{(t)}}\sum_{c=\xi}^{t}\mathbf{X}\boldsymbol{\delta}_{[k]}^{(c)}\right\|^2\right) \\
&- \frac{2\nu^2}{\lambda n^2}\left(\sum_{k\notin\mathcal{P}_S^{(t)}}\sum_{l=\xi}^{t}\boldsymbol{\delta}_{[k]}^{(l)}\right)^\top\mathbf{X}^\top\mathbf{X}\sum_{k\in\mathcal{P}_S^{(t)}}\boldsymbol{\delta}_{[k]}^{(t)} \\
&- \frac{1}{n}\sum_{k\notin\mathcal{P}_S^{(t)}}\left(\sum_{i\in\mathcal{I}_k}\phi_i^*((-\hat{\boldsymbol{\alpha}} - \nu\sum_{c=\xi}^{t}\boldsymbol{\delta}_{[k]}^{(c)})_i)\right) \\
&\text{(mean value theorem, concave)} \\
&= \underbrace{-\frac{1}{n}\sum_{k=1}^{n}\left(\sum_{i\in\mathcal{I}_k}(1-\nu)\phi_i^*(-\hat{\alpha}_i)\right) - (1-\nu)\frac{\lambda}{2}\left\|\mathbf{w}(\hat{\boldsymbol{\alpha}})\right\|^2}_{(1-\nu)D(\hat{\boldsymbol{\alpha}})} \\
&+ \frac{1}{n}\sum_{k\notin\mathcal{P}_S^{(t)}}\left(\sum_{i\in\mathcal{I}_k}(1-\nu)\phi_i^*(-\hat{\alpha}_i)\right) \\
&+ \nu\sum_{k\in\mathcal{P}_S^{(t)}}\left(-\frac{1}{n}\sum_{i\in\mathcal{I}_k}\phi_i^*(-(\hat{\boldsymbol{\alpha}} + \boldsymbol{\delta}_{[k]}^{(t)})_i)\right. \\
&\left. - \frac{1}{S}\frac{\lambda}{2}\left\|\mathbf{w}(\hat{\boldsymbol{\alpha}})\right\|^2 - \frac{1}{n}\mathbf{w}(\hat{\boldsymbol{\alpha}})^\top\mathbf{X}\boldsymbol{\delta}_{[k]}^{(t)} - \frac{\lambda}{2}\sigma\left\|\frac{1}{\lambda n}\mathbf{X}\boldsymbol{\delta}_{[k]}^{(t)}\right\|^2\right) \\
&- \frac{\lambda}{2}\left(\frac{2\nu}{\lambda n}\sum_{k\notin\mathcal{P}_S^{(t)}}\mathbf{w}(\hat{\boldsymbol{\alpha}})^\top\mathbf{X}\left(\sum_{l=\xi}^{t}\boldsymbol{\delta}_{[k]}^{(l)}\right) + \left(\frac{\nu}{\lambda n}\right)^2\left\|\sum_{k\notin\mathcal{P}_S^{(t)}}\sum_{c=\xi}^{t}\mathbf{X}\boldsymbol{\delta}_{[k]}^{(c)}\right\|^2\right) \\
&- \frac{2\nu^2}{\lambda n^2}\left(\sum_{k\notin\mathcal{P}_S^{(t)}}\sum_{l=\xi}^{t}\boldsymbol{\delta}_{[k]}^{(l)}\right)^\top\mathbf{X}^\top\mathbf{X}\sum_{k\in\mathcal{P}_S^{(t)}}\boldsymbol{\delta}_{[k]}^{(t)}
\end{aligned}
$$

$$
\begin{aligned}
&- \frac{1}{n}\sum_{k\notin\mathcal{P}_S^{(t)}}\left(\sum_{i\in\mathcal{I}_k}\phi_i^*((-\hat{\boldsymbol{\alpha}}-\nu\sum_{c=\xi}^{t}\boldsymbol{\delta}_{[k]}^{(c)})_i)\right)\\
=&(1-\nu)D(\hat{\boldsymbol{\alpha}})+\nu\sum_{k\in\mathcal{P}_S^{(t)}}D_k(\boldsymbol{\delta}_{[k]}^{(t)};\boldsymbol{\alpha}_{[k]},\hat{\mathbf{w}})+\frac{1-\nu}{n}\sum_{k\notin\mathcal{P}_S^{(t)}}\left(\sum_{i\in\mathcal{I}_k}\phi_i^*(-\hat{\alpha}_i)\right)\\
&- \frac{\lambda}{2}\left(\frac{2\nu}{\lambda n}\sum_{k\notin\mathcal{P}_S^{(t)}}\mathbf{w}(\hat{\boldsymbol{\alpha}})^\top\mathbf{X}\left(\sum_{l=\xi}^{t}\boldsymbol{\delta}_{[k]}^{(l)}\right)+\left(\frac{\nu}{\lambda n}\right)^2\left\|\sum_{k\notin\mathcal{P}_S^{(t)}}\sum_{c=\xi}^{t}\mathbf{X}\boldsymbol{\delta}_{[k]}^{(c)}\right\|^2\right)\\
&- \frac{2\nu^2}{\lambda n^2}\left(\sum_{k\notin\mathcal{P}_S^{(t)}}\sum_{l=\xi}^{t}\boldsymbol{\delta}_{[k]}^{(l)}\right)^\top\mathbf{X}^\top\mathbf{X}\sum_{k\in\mathcal{P}_S^{(t)}}\boldsymbol{\delta}_{[k]}^{(t)}\\
&- \frac{1}{n}\sum_{k\notin\mathcal{P}_S^{(t)}}\left(\sum_{i\in\mathcal{I}_k}\phi_i^*((-\hat{\boldsymbol{\alpha}}-\nu\sum_{c=\xi}^{t}\boldsymbol{\delta}_{[k]}^{(c)})_i)\right)\quad\square
\end{aligned}
$$

Assumption 5 (*Bounded Delay of Global Updates, Γ*)**.** There exists a $\varrho < e^{\frac{2}{\Gamma+1}}$ such that

$$\left\|\boldsymbol{\delta}^{(t-1)}\right\|^2 \le \varrho\left\|\boldsymbol{\delta}^{(t)}\right\|^2. \tag{30}$$

Lemma 9 (*Global Convergence at Each Iteration*)**.** *If ϕ_i^* are all $(1/\mu)$-strongly convex and Assumptions 2–5 are satisfied then for any $s\in[0,1]$, any round t of Algorithm 2 satisfies*

$$\mathbb{E}[D(\boldsymbol{\alpha}^{(t+1)})-D(\boldsymbol{\alpha}^{(t)})] \ge \Psi(1-\Theta)\left(sG(\hat{\boldsymbol{\alpha}})-\frac{\sigma}{2\lambda}\left(\frac{s}{n}\right)^2\hat{R}\right) \tag{31}$$

where

$$\Psi := \nu\left(1-\frac{(\Gamma+1)e^2ML_{max}}{\lambda n}+\frac{S\Gamma ML_{max}}{K\lambda n}\right)\le 1,\ \textit{and} \tag{32}$$

$$\hat{R} := -\frac{\lambda\mu n(1-s)}{\sigma s}\left\|\hat{\mathbf{u}}-\hat{\boldsymbol{\alpha}}\right\|^2+\sum_{k=1}^{K}\left\|\mathbf{X}(\hat{\mathbf{u}}-\hat{\boldsymbol{\alpha}})_{[k]}\right\|^2, \tag{33}$$

for $\hat{\mathbf{u}}\in\mathbb{R}^n$ with $-\hat{u}_i\in\partial\phi_i(\mathbf{w}(\hat{\boldsymbol{\alpha}})^\top\mathbf{x}_i)$.

Proof. For the sake of notation, we will write $\boldsymbol{\alpha}$ instead of $\boldsymbol{\alpha}^t$, $\mathbf{w}$ instead of $\mathbf{w}(\boldsymbol{\alpha}^t)$, $\hat{\mathbf{w}}$ instead of $\mathbf{w}(\hat{\boldsymbol{\alpha}})$, and $\boldsymbol{\delta}$ instead of $\boldsymbol{\delta}^t$.

Now, the expected change of the dual objective is

$$\begin{aligned}
\mathbb{E}[D(\boldsymbol{\alpha}^t)-D(\boldsymbol{\alpha}^{(t+1)})] =&\mathbb{E}[D(\boldsymbol{\alpha}^t)-D(\hat{\boldsymbol{\alpha}})+D(\hat{\boldsymbol{\alpha}})-D(\boldsymbol{\alpha}^{(t+1)})]\\
=&\mathbb{E}[D(\boldsymbol{\alpha}^t)-D(\hat{\boldsymbol{\alpha}})]+\mathbb{E}[D(\hat{\boldsymbol{\alpha}})-D(\boldsymbol{\alpha}^{(t+1)})]
\end{aligned}$$

Thus, it is a summation of two parts. Let us estimate both parts as follows,

$$
\begin{aligned}
&\mathbb{E}[D(\boldsymbol{\alpha}^t)-D(\hat{\boldsymbol{\alpha}})]\\
&=\mathbb{E}\left[-\frac{1}{n}\sum_{i\notin\mathcal{I}}\phi_i^*(-\hat{\alpha}_i-\nu\sum_{c=\xi}^{t}\boldsymbol{\delta}^{(c)})\right.\\
&\quad-\frac{1}{n}\sum_{k\in\mathcal{P}_S}\left(\sum_{i\in\mathcal{I}_k}\phi_i^*(-\hat{\alpha}_i)\right)\\
&\quad-\frac{\lambda}{2}\left\|\frac{1}{\lambda n}\sum_{k\notin\mathcal{P}_S}\mathbf{X}\left(\hat{\boldsymbol{\alpha}}_{[k]}+\nu\sum_{c=\xi}^{t}\boldsymbol{\delta}_{[k]}^{(c)}\right)\right\|^2\\
&\quad\left.+\frac{1}{n}\sum_{k=1}^{n}\phi^*(-\hat{\boldsymbol{\alpha}}_{[k]})+\frac{\lambda}{2}\left\|\hat{\mathbf{w}}\right\|^2\right]\\
&=\mathbb{E}\left[-\frac{1}{n}\sum_{i\notin\mathcal{I}}\phi_i^*(-\hat{\alpha}_i-\nu\sum_{c=\xi}^{t}\boldsymbol{\delta}^{(c)})+\frac{1}{n}\sum_{k\notin\mathcal{P}_S}\left(\sum_{i\in\mathcal{I}_k}\phi_i^*(-\hat{\alpha}_i)\right)\right]\\
&\quad-\frac{\lambda}{2}\mathbb{E}\left[\frac{2\nu}{\lambda n}\sum_{k\notin\mathcal{P}_S}\mathbf{w}(\hat{\boldsymbol{\alpha}})^\top\mathbf{X}\left(\sum_{c=\xi}^{t}\boldsymbol{\delta}_{[k]}^{(c)}\right)\right.\\
&\quad\left.+\left(\frac{\nu}{\lambda n}\right)^2\left\|\sum_{k\notin\mathcal{P}_S}\sum_{c=\xi}^{t}\mathbf{X}\boldsymbol{\delta}_{[k]}^{(c)}\right\|^2\right]
\end{aligned}
$$

$$
\begin{aligned}
&\mathbb{E}[D(\hat{\boldsymbol{\alpha}})-D(\boldsymbol{\alpha}^{(t+1)})]=\mathbb{E}\left[D(\hat{\boldsymbol{\alpha}})-D(\boldsymbol{\alpha}+\nu\sum_{k\in\mathcal{P}_S}\boldsymbol{\delta}_{[k]}^{(t)})\right]\\
\le&\mathbb{E}\left[D(\hat{\boldsymbol{\alpha}})-(1-\nu)D(\hat{\boldsymbol{\alpha}})-\nu\sum_{k\in\mathcal{P}_S}D_k(\boldsymbol{\delta}_{[k]}^{(t)};\boldsymbol{\alpha}_{[k]},\hat{\mathbf{w}})\right.\\
&\quad\left.+\frac{\nu}{n}\sum_{k\notin\mathcal{P}_S}\left(\sum_{i\in\mathcal{I}_k}\phi_i^*(-\hat{\alpha}_i)\right)\right]\\
&\quad-\mathbb{E}\left[-\frac{1}{n}\sum_{k\notin\mathcal{P}_S}\left(\sum_{i\in\mathcal{I}_k}\phi_i^*((-\hat{\boldsymbol{\alpha}}-\nu\sum_{c=\xi}^{t}\boldsymbol{\delta}_{[k]}^{(c)})_i)\right)\right.\\
&\quad\left.+\frac{1}{n}\sum_{k\notin\mathcal{P}_S}\left(\sum_{i\in\mathcal{I}_k}\phi_i^*(-\hat{\alpha}_i)\right)\right]\\
&\quad+\frac{\lambda}{2}\mathbb{E}\left[\frac{2\nu}{\lambda n}\sum_{k\notin\mathcal{P}_S}\mathbf{w}(\hat{\boldsymbol{\alpha}})^\top\mathbf{X}\left(\sum_{c=\xi}^{t}\boldsymbol{\delta}_{[k]}^{(c)}\right)\right.\\
&\quad\left.+\left(\frac{\nu}{\lambda n}\right)^2\left\|\sum_{k\notin\mathcal{P}_S}\sum_{c=\xi}^{t}\mathbf{X}\boldsymbol{\delta}_{[k]}^{(c)}\right\|^2\right]\\
&\quad+\mathbb{E}\left[\frac{2\nu^2}{\lambda n^2}\left(\sum_{k\notin\mathcal{P}_S}\sum_{c=\xi}^{t}\boldsymbol{\delta}_{[k]}^{(c)}\right)^\top\mathbf{X}^\top\mathbf{X}\sum_{k\in\mathcal{P}_S}\boldsymbol{\delta}_{[k]}^{(t)}\right]\quad\text{(Lemma 8)}
\end{aligned}
$$

Therefore,

$$
\begin{aligned}
&\mathbb{E}[D(\boldsymbol{\alpha}^t)-D(\boldsymbol{\alpha}^{(t+1)})]=\mathbb{E}[D(\boldsymbol{\alpha}^t)-D(\hat{\boldsymbol{\alpha}})]+\mathbb{E}[D(\hat{\boldsymbol{\alpha}})-D(\boldsymbol{\alpha}^{(t+1)})]\\
\le&\mathbb{E}\left[D(\hat{\boldsymbol{\alpha}})-(1-\nu)D(\hat{\boldsymbol{\alpha}})-\nu\sum_{k\in\mathcal{P}_S}D_k(\boldsymbol{\delta}_{[k]}^{(t)};\boldsymbol{\alpha}_{[k]},\hat{\mathbf{w}})\right.\\
&\quad\left.\frac{\nu}{n}\sum_{k\notin\mathcal{P}_S}\left(\sum_{i\in\mathcal{I}_k}\phi_i^*(-\hat{\alpha}_i)\right)\right]\\
&\quad+\mathbb{E}\left[\frac{2\nu^2}{\lambda n^2}\left(\sum_{k\notin\mathcal{P}_S}\sum_{c=\xi}^{t}\boldsymbol{\delta}_{[k]}^{(c)}\right)^\top\mathbf{X}^\top\mathbf{X}\sum_{k\in\mathcal{P}_S}\boldsymbol{\delta}_{[k]}^{(t)}\right]\\
&\text{(sum of previous two inequalities)}\\
=&\nu\mathbb{E}\left[D(\hat{\boldsymbol{\alpha}})-\sum_{k=1}^{K}D_k(\boldsymbol{\delta}_{[k]}^*;\boldsymbol{\alpha}_{[k]},\hat{\mathbf{w}})+\sum_{k=1}^{K}D_k(\boldsymbol{\delta}_{[k]}^*;\boldsymbol{\alpha}_{[k]},\hat{\mathbf{w}})\right.\\
&\quad\left.-\sum_{k=1}^{K}D_k(\mathbf{0};\hat{\boldsymbol{\alpha}}_{[k]},\hat{\mathbf{w}})\right]\\
&\quad+\mathbb{E}\left[\frac{2\nu^2}{\lambda n^2}\left(\sum_{k\notin\mathcal{P}_S}\sum_{c=\xi}^{t}\boldsymbol{\delta}_{[k]}^{(c)}\right)^\top\mathbf{X}^\top\mathbf{X}\sum_{k\in\mathcal{P}_S}\boldsymbol{\delta}_{[k]}^{(t)}\right]
\end{aligned}
$$

$$
\begin{aligned}
&\leq \nu \Bigg(\Theta \Bigg(\sum_{k=1}^{K} D_k(\boldsymbol{\delta}^*_{[k]}; \boldsymbol{\alpha}_{[k]}, \hat{\mathbf{w}}) - \underbrace{\sum_{k=1}^{K} D_k(\mathbf{0}; \hat{\boldsymbol{\alpha}}_{[k]}, \hat{\mathbf{w}})}_{D(\hat{\boldsymbol{\alpha}})} \Bigg) + D(\hat{\boldsymbol{\alpha}}) \\
&\quad - \sum_{k=1}^{K} D_k(\boldsymbol{\delta}^*_{[k]}; \boldsymbol{\alpha}_{[k]}, \hat{\mathbf{w}}) \Bigg) \\
&\quad + \mathbb{E}\left[\frac{\nu^2}{\lambda n^2} \left(\sum_{c=\xi}^{t} \boldsymbol{\delta}^{(c)}_{[k]} \right)^{\top} \mathbf{X}^{\top}\mathbf{X} \sum_{k\in\mathcal{P}_S} \boldsymbol{\delta}^{(t)}_{[k]} \right] \quad \text{(Lemma 7)} \\
&= \nu(1-\Theta) \left(D(\hat{\boldsymbol{\alpha}}) - \sum_{k=1}^{K} D_k(\boldsymbol{\delta}^*_{[k]}; \boldsymbol{\alpha}_{[k]}, \hat{\mathbf{w}}) \right) \\
&\quad + \mathbb{E}\left[\frac{2\nu^2}{\lambda n^2} \left(\sum_{k\notin\mathcal{P}_S} \sum_{c=\xi}^{t} \boldsymbol{\delta}^{(c)}_{[k]} \right)^{\top} \mathbf{X}^{\top}\mathbf{X} \sum_{k\in\mathcal{P}_S} \boldsymbol{\delta}^{(t)}_{[k]} \right] \\
&\leq \nu(1-\Theta) \left(D(\hat{\boldsymbol{\alpha}}) - \sum_{k\in\mathcal{P}_S} D_k(\boldsymbol{\delta}^*_{[k]}; \boldsymbol{\alpha}_{[k]}, \hat{\mathbf{w}}) \right) \\
&\quad + \frac{\nu}{\lambda n} \underbrace{\left(\mathbb{E}\left[\left\| \sum_{k\notin\mathcal{P}_S} \mathbf{X} \left(\sum_{c=\xi}^{t} \boldsymbol{\delta}^{(c)}_{[k]} \right) \right\|^2 \right] + \mathbb{E}\left[\left\| \sum_{k\in\mathcal{P}_S} \mathbf{X}\boldsymbol{\delta}^{(t)}_{[k]} \right\|^2 \right] \right)}_{A} \\
&\quad (a^2 + b^2 \geq 2ab)
\end{aligned}
$$

Before bounding the term A, we need the following proposition:

Proposition 10.

$$
\begin{aligned}
\mathbb{E}\left[\left\| \mathbf{X}\boldsymbol{\delta}^{(t)}_{[k]} \right\|^2 \right] &= \frac{1}{S}\mathbb{E}\left[\sum_{k\in\mathcal{P}_S} \left\| \mathbf{X}\boldsymbol{\delta}^{(t)}_{[k]} \right\|^2 \right] \\
&= \frac{1}{K}\mathbb{E}\left[\sum_{k=1}^{K} \left\| \mathbf{X}\boldsymbol{\delta}^{(t)}_{[k]} \right\|^2 \right] \quad \forall k \in \{1, \ldots, K\}.
\end{aligned}
$$

Now, let us bound the term A. We have

$$
\begin{aligned}
A =& \mathbb{E}\left[\left\| \sum_{k\notin\mathcal{P}_S} \mathbf{X} \left(\sum_{c=\xi}^{t} \boldsymbol{\delta}^{(c)}_{[k]} \right) \right\|^2 \right] + \mathbb{E}\left[\left\| \sum_{k\in\mathcal{P}_S} \mathbf{X}\boldsymbol{\delta}^{(t)}_{[k]} \right\|^2 \right] \\
\leq& \mathbb{E}\left[(\Gamma+1) \sum_{c=t-\Gamma}^{t} \left\| \sum_{k\notin\mathcal{P}_S} \mathbf{X}\boldsymbol{\delta}^{c}_{[k]} \right\|^2 \right] \\
&+ \mathbb{E}\left[S \sum_{k\in\mathcal{P}_S} \left\| \mathbf{X}\boldsymbol{\delta}^{(t)}_{[k]} \right\|^2 \right] \quad \text{(Cauchy Schwarz Inequality)} \\
\leq& \mathbb{E}\left[\frac{(K-S)(\Gamma+1)M}{K} \sum_{k=1}^{K} \sum_{c=t-\Gamma}^{t-1} \left\| \boldsymbol{\delta}^{c}_{[k]} \right\|^2 \right] \\
&+ \mathbb{E}\left[\frac{SM}{K} \sum_{k=1}^{K} \left\| \boldsymbol{\delta}^{(t)}_{[k]} \right\|^2 \right] \quad \text{(Propositions 2\&10)} \\
\leq& \mathbb{E}\left[\frac{(K-S)(\Gamma+1)M}{K} \left(\sum_{c=t-\Gamma}^{t} \varrho^c \right) \sum_{k=1}^{K} \left\| \boldsymbol{\delta}^{(t)}_{[k]} \right\|^2 \right] \\
&+ \mathbb{E}\left[\frac{SM}{K} \sum_{k=1}^{K} \left\| \boldsymbol{\delta}^{(t)}_{[k]} \right\|^2 \right] \quad \text{(By (30))} \\
\leq& \frac{(K-S)(\Gamma+1)ML_{max}}{K} \sum_{c=t-\Gamma}^{t} \varrho^c \left(D(\hat{\boldsymbol{\alpha}}) - \sum_{k=1}^{K} D_k(\boldsymbol{\delta}^{(t)}_{[k]}; \boldsymbol{\alpha}_{[k]}, \hat{\mathbf{w}}) \right) \\
&+ \frac{SML_{max}}{K} \sum_{k=1}^{K} \left(D(\hat{\boldsymbol{\alpha}}) - D_k(\boldsymbol{\delta}^{(t)}_{[k]}; \boldsymbol{\alpha}^{(t)}_{[k]}, \hat{\mathbf{w}}) \right) \quad \text{(Proposition 5)}
\end{aligned}
$$

where $D(\hat{\boldsymbol{\alpha}}) = D_k(\mathbf{0}; \boldsymbol{\alpha}_{[k]}, \hat{\mathbf{w}})$. Thus, Eq. (9) can be rewritten as,

$$
\begin{aligned}
&\mathbb{E}\left[D_k(\boldsymbol{\delta}^*_{[k]}; \boldsymbol{\alpha}_{[k]}, \hat{\mathbf{w}}) - D_k(\boldsymbol{\delta}^{(t)}_{[k]}; \boldsymbol{\alpha}_{[k]}, \hat{\mathbf{w}}) \right] \\
&\leq \Theta \left(D_k(\boldsymbol{\delta}^*_{[k]}; \boldsymbol{\alpha}_{[k]}, \hat{\mathbf{w}}) - D(\hat{\boldsymbol{\alpha}}) \right) + D(\hat{\boldsymbol{\alpha}}) - D(\hat{\boldsymbol{\alpha}}) \\
&D(\hat{\boldsymbol{\alpha}}) - D_k(\boldsymbol{\delta}^{(t)}_{[k]}; \boldsymbol{\alpha}_{[k]}, \hat{\mathbf{w}}) \\
&\leq (1-\Theta) D(\hat{\boldsymbol{\alpha}}) - (1-\Theta) D_k(\boldsymbol{\delta}^*_{[k]}; \boldsymbol{\alpha}_{[k]}, \hat{\mathbf{w}}) \\
&D(\hat{\boldsymbol{\alpha}}) - D_k(\boldsymbol{\delta}^{(t)}_{[k]}; \boldsymbol{\alpha}_{[k]}, \hat{\mathbf{w}}) \\
&\leq -(1-\Theta) \left(D_k(\boldsymbol{\delta}^*_{[k]}; \boldsymbol{\alpha}_{[k]}, \hat{\mathbf{w}}) - D(\hat{\boldsymbol{\alpha}}) \right) \qquad (34)
\end{aligned}
$$

Then, A can be bounded as,

$$
\begin{aligned}
A \leq& -\frac{(K-S)(\Gamma+1)ML_{max}}{K} \varrho^{\Gamma+1}(1-\Theta) \sum_{k=1}^{K} \left(D(\hat{\boldsymbol{\alpha}}) \right. \\
&\left. - D_k(\boldsymbol{\delta}^*_{[k]}; \boldsymbol{\alpha}_{[k]}, \hat{\mathbf{w}}) \right) \\
&- \frac{SML_{max}}{K}(1-\Theta) \sum_{k=1}^{K} \left(D(\hat{\boldsymbol{\alpha}}) - D_k(\boldsymbol{\delta}^*_{[k]}; \boldsymbol{\alpha}_{[k]}, \hat{\mathbf{w}}) \right) \quad \text{(By (34))} \\
\leq& -\frac{(K-S)(\Gamma+1)e^2ML_{max}}{K}(1-\Theta) \sum_{k=1}^{K} \left(D(\hat{\boldsymbol{\alpha}}) \right. \\
&\left. - D_k(\boldsymbol{\delta}^*_{[k]}; \boldsymbol{\alpha}_{[k]}, \hat{\mathbf{w}}) \right) \\
&- \frac{SML_{max}}{K}(1-\Theta) \sum_{k=1}^{K} \left(D(\hat{\boldsymbol{\alpha}}) \right. \\
&\left. - D_k(\boldsymbol{\delta}^*_{[k]}; \boldsymbol{\alpha}_{[k]}, \hat{\mathbf{w}}) \right) \quad \text{(Assumption 5)}
\end{aligned}
$$

By substituting A, we have

$$
\begin{aligned}
\mathbb{E}[D(\boldsymbol{\alpha}^t) - D(\boldsymbol{\alpha}^{(t+1)})] \leq& \nu \left(1 - \frac{(\Gamma+1)e^2ML_{max}}{\lambda n} + \frac{S\Gamma ML_{max}}{K\lambda n} \right) \\
&(1-\Theta) \sum_{k=1}^{K} \left(D(\hat{\boldsymbol{\alpha}}) - D_k(\boldsymbol{\delta}^*_{[k]}) \right) \\
\mathbb{E}[D(\boldsymbol{\alpha}^t) - D(\boldsymbol{\alpha}^{(t+1)})] \leq& \Psi(1-\Theta) \sum_{k=1}^{K} \left(D(\hat{\boldsymbol{\alpha}}) - D_k(\boldsymbol{\delta}^*_{[k]}) \right)
\end{aligned}
$$

Using Eq. C in the proof of Lemma 5 in [11], we can show that

$$
\begin{aligned}
&\mathbb{E}[D(\boldsymbol{\alpha}^t) - D(\boldsymbol{\alpha}^{(t+1)})] \\
&\leq \Psi(1-\Theta) \left(-sG(\hat{\boldsymbol{\alpha}}) - \frac{1}{2\mu}(1-s)s\frac{1}{n} \left\| \hat{\mathbf{u}} - \hat{\boldsymbol{\alpha}} \right\|^2 \right. \\
&\quad \left. + \frac{\sigma}{2\lambda} \left(\frac{s}{n} \right)^2 \sum_{k=1}^{K} \left\| \mathbf{X}(\hat{\mathbf{u}} - \hat{\boldsymbol{\alpha}})_{[k]} \right\| \right) \\
&= \Psi(1-\Theta) \left(-sG(\hat{\boldsymbol{\alpha}}) + \frac{\sigma}{2\lambda} \left(\frac{s}{n} \right)^2 \hat{R} \right) \quad \square
\end{aligned}
$$

Remark. When S (the minimal number of workers required to update before a global update is communicated) is fixed in (32), Ψ will approach 0 when Γ (the maximal delay allowed for the slowest worker) becomes larger and larger since $(\Gamma+1)e^2 >$

Γ. In other words, the improvement from $D(\boldsymbol{\alpha}^{(t)})$ to $D(\boldsymbol{\alpha}^{(t+1)})$ at iteration t becomes smaller when a larger delay is allowed among the workers.

When Γ is fixed in (32), Ψ will be larger when S is set larger. The improvement from $D(\boldsymbol{\alpha}^{(t)})$ to $D(\boldsymbol{\alpha}^{(t+1)})$ at iteration t will be more significant when more updates from different workers are taken into account. However, when $\Gamma = 0$, Ψ will be independent from S because all updates have to be gathered by the master.

Using the main results in [11] and combining Lemma 7 with Lemma 9 we get the following two convergence results for two types of loss functions: (i) smooth and (ii) Lipschitz continuous. The theorems use the following quantities: $\sigma_{max} = \max_k \sigma_k$, and $\sigma_{sum} = \sum_k \sigma_k n_k$ where $\forall k$, $\sigma_k = \max_{\alpha_{[k]} \in \mathbb{R}^n} \left\|\mathbf{X}\boldsymbol{\alpha}_{[k]}\right\|^2 / \left\|\boldsymbol{\alpha}_{[k]}\right\|^2$.

Theorem 11 (*Global Convergence for* $(1/\mu)$*-Smooth Functions*)**.** *If the loss functions* ϕ_i *are all* $(1/\mu)$*-smooth, then in* T_1 *iterations Algorithm 2 finds a solution with objective at most* ϵ_D *from the optimal,* i.e., $\mathbb{E}[D(\alpha^*) - D(\alpha^{(T_1)})] \leq \epsilon_D$ *whenever* $T_1 \geq C_1 \log \frac{1}{\epsilon_D}$ *where* $C_1 = \frac{1}{\Psi(1-\Theta)}(1 + \frac{\sigma_{max}\sigma}{\nu\lambda n})$ *and* Θ *is given by (25). Furthermore, in* T_2 *iterations, it finds a solution with duality gap at most* ϵ_{gap}, i.e., $\mathbb{E}[P(\mathbf{w}(\alpha^{(T_2)})) - D(\alpha^{(T_2)})] \leq \epsilon_{gap}$ *whenever* $T_2 \geq C_1 \log \frac{C_1}{\epsilon_D}$.

Theorem 12 (*Global Convergence for L-Lipschitz Functions*)**.** *If the loss functions* ϕ_i *are all L-Lipschitz, then in* T_1 *iterations Algorithm 2 finds a solution with duality gap at most* ϵ_{gap}, i.e., $\mathbb{E}[P(\mathbf{w}(\bar{\alpha})) - D(\bar{\alpha})] \leq \epsilon_{gap}$ *for the average iterate* $\bar{\alpha} = \frac{1}{T_1 - T_0}\sum_{t=T_0+1}^{T_1-1} \alpha^{(t)}$ *whenever* $T_1 \geq T_0 + \max\{\lceil \frac{1}{\Psi(1-\Theta)} \rceil, \frac{4L^2\sigma_{sum}\sigma}{\lambda n^2 \epsilon_{gap} \Psi(1-\Theta)}\}$, *and* $T_0 \geq \max\{0, \lceil \frac{1}{\Psi(1-\Theta)} \log \frac{2\lambda n^2 (D(\alpha^*)) - D(\alpha^{(0)})}{4L^2\sigma_{sum}\sigma} \rceil\} + \max\{0, \frac{2}{\Psi(1-\Theta)}(\frac{8L^2\sigma_{sum}\sigma}{\lambda n^2 \epsilon_{gap}} - 1)\}$ *and* Θ *is given by (25).*

Theorem 12 establishes the convergence for L-Lipschitz continuous loss functions, and Theorem 11 proves a linear convergence rate for smooth convex loss functions.

5. Experimental results

We implemented our algorithm in C++ using Open MPI and OpenMP. All our experiments were conducted on the Biowulf cluster at the National Institutes of Health, USA, using up to $K = 16$ nodes where each node has 58 GB main memory and $R = 16$ cores each with a 2.6 GHz Xeon E5-2650v2 processor and 20 MB secondary cache. Though the cores in the cluster are hyperthreading enabled, we did not use hyperthreading mode for our experiments. Each node in the cluster runs exactly one MPI task corresponding to a worker which in turn runs one OpenMP thread on each core available within the node. The main thread in a worker handles the inter-node communication and the root MPI task works as the master. For MPI node scheduling we used the `slurm` task scheduler with the settings `--ntasks-per-node=1` and `--threads-per-core=1` whereas for OpenMP thread scheduling we used a simple CPU affinity scheduler that always assigned thread i to physical core i where $i \in \{1, \ldots, 16\}$.

5.1. Datasets

We evaluated our algorithm against three other algorithms on four binary classification datasets, `rcv1`, `webspam`, `kddb` and `splicesite` from the LIBSVM [3] website (https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html) as shown in Table 1. For datasets `rcv1` and `kddb` we used separate training and test data files downloaded directly from the website. As there was no separate test data file for `webspam`, we divided the data file into two parts and used the first part containing 80% of the datapoints as the training set and the remaining as the test set. For `splicesite` we used the test file on the website as our training set (to better test the scalability of our algorithm as the test file was bigger). We chose the datasets in such a way that we had representatives of several scenarios: data sample-heavy `rcv1` where $n \gg d$, feature-heavy `webspam` where $n \ll d$, both sample and feature heavy `kddb` where both $n \approx d$ are high, and the big dataset `splicesite` which was more dense in both ways.

5.2. Comparison of algorithms

We experimented with the following four algorithms:

- *Baseline*: a sequential implementation of stochastic dual coordinate ascent (DCA) [6] which runs on a single core of a single node
- *CoCoA+*: a MPI based distributed implementation of stochastic DCA [11] which runs on multiple nodes, however, each node uses a single core
- *PassCoDe*: an OpenMP based parallel implementation of stochastic DCA [7] which runs on a single node, however, the node uses multiple cores
- *Hybrid-DCA*: an OpenMP + MPI implementation of our hybrid parallel distributed approach for stochastic DCA which runs on multiple nodes and each node uses multiple cores.

5.3. Parameter settings

We evaluated all the four algorithms for the hinge loss, though other loss functions could be tested too, with the regularization parameter λ. In our experiments with three values $\lambda \in \{10^{-3}, 10^{-4}, 10^{-5}\}$, we observed similar patterns of results and we reported the results for $\lambda = 10^{-4}$ only. All the three parallel/distributed algorithms, namely *PassCoDe*, *CoCoA+*, *Hybrid-DCA*, have a global parameter G denoting the number of basic updates to the dual variable $\boldsymbol{\alpha}$ that are made in each global round. The parameter G acts as a tradeoff between the progress on the dual objective and the time taken in each global round. In the original implementation in [7] *PassCoDe* sets G equal to the number of datapoints n in the dataset whereas *CoCoA* implementation in [11] uses a smaller G than n.

We experimented with different values of G and found $G = 40000, 30000, 2000000$ for the datasets `rcv1`, `webspam`, `kddb`, respectively, gave the best results in our empirical study. In our implementation, for *PassCoDe* on t cores each thread made about $H = G/t$ local updates, for *PassCoDe* on p nodes, each MPI task made $H = G/p$ local updates and for *Hybrid-DCA* on p nodes each with t cores, each thread within an MPI task made $H = G/(p \times t)$ local updates so that all the three algorithms made total G updates in a global round. Though the sequential algorithm *Baseline* did not have any local iteration parameter, for better comparison we computed performance metrics such as time taken after every G update and treated such G updates as a global round. We set aggregation parameters $\nu = 1$, and the scaling parameter $\sigma = K$ for both *CoCoA+*, *Hybrid-DCA* as recommended in [11].

Our *Hybrid-DCA* algorithm has additional parameters, the bounded barrier S and bounded delay Γ so that updates from only S out of p workers are incorporated in each global round with a maximum delay of Γ rounds for any update from the workers. In our implementation of *Hybrid-DCA*, we treated the two cases slightly differently: synchronous *Hybrid-DCA* with $S = p$ where the updates from the workers were merged using the collective operation `MPI_Allreduce`, and asynchronous *Hybrid-DCA* with $S < p$ where the updates from a subset of S out of p

Table 1
Datasets.

Dataset details	rcv1	webspam	kddb	splicesite
Training set	rcv1_train.binary	webspam_*_trigram	kddb	splice_site.t
File size	1.2 GB	20 GB	5.1 GB	280 GB
Training size n	677,399	280,000	19,264,097	4,627,840
Number of features d	47,236	16,609,143	29,890,095	11,725,480
Non-zero entries nnz	49,556,258	1,045,051,224	566,345,888	15,383,587,858
Test set	rcv1_test.binary	webspam_*_trigram	kddb.t	–
Test size	677,399	69,632	748,401	–

workers were merged and distributed using basic MPI "send and receive" commands. For all our experiments, we ran algorithms 10 times for each setting and reported the average of measured values.

5.4. Optimization performance

Fig. 5 shows the progress of duality gap achieved by the four algorithms on the three relatively smaller datasets rcv1, webspam and kddb. We chose the number of nodes ($p \leq K$) and the number of cores ($t \leq R$) per node such that the total number of worker cores ($p \times t$) was the same (16) for all algorithms except *Baseline*. For *Hybrid-DCA* we set the parameters as the bounding barrier $S = p$ and the delay $\Gamma = 1$ so that updates from all workers were incorporated in each global round. However, for $p = t = 4$ we also experimented with $S = 3$ and $\Gamma = 4$ to compare how *Hybrid-DCA* performed when updates from only 3 out of 4 workers were incorporated in each round with a maximum delay of 4 rounds for any update from the workers. The duality gap was measured as $P(\mathbf{w}) - D(\boldsymbol{\alpha})$ where the primal estimate $\mathbf{w}$ was computed as ($\mathbf{w} = \mathbf{v}$) $\mathbf{w} = \mathbf{w}(\boldsymbol{\alpha})$ at the end of each global round. However, when $S < K$ it was not possible for the master in *Hybrid-DCA* to gather the parts of $P(\mathbf{w})$ from all workers at the end of each global round. To workaround in such a case, we let the master temporarily store $\mathbf{w}$ in disk after each round and at the end of all stipulated rounds, the workers computed the respective parts of $P(\mathbf{w})$ from the stored $\mathbf{w}$ and the master computed the duality gap using a series of synchronous all-reduce computations from all the workers.

The bottom row of Fig. 5 shows the progress of the duality gap over time, while the top row shows the progress after each global round. In terms of progress across global rounds, the algorithms performed somewhat equally except for $S < p$ where *Hybrid-DCA* had slightly slower progress on duality gap as the updates from one of the workers was missing in each round. In terms of time, there was no clear winner of the three parallel/distributed algorithms. *CoCoA+* showed an advantage over *PassCoDe* on datasets with a smaller number of datapoints n, such as rcv1 and webspam, because the costly inter-node communication complexity was $O(n)$ per round. For kddb where the dataset had more non-zero data elements, *PassCoDe* performed better. For all the three datasets the performance of *Hybrid-DCA* came in between *CoCoA+* and *PassCoDe* by balancing inter-core and inter-node communications. Since the asynchronous *Hybrid-DCA* with $S < p$ missed updates from some of the workers, took longer time as expected than synchronous *Hybrid-DCA* with $S = p$.

5.5. Test accuracy

To compare the quality of the solution obtained by each algorithm we used test datasets from LIBSVM binary repository the details of which are given in Table 1 and in Section 5.1. Note that the webspam dataset in LIBSVM did not have a separate test dataset. We divided it into two parts and took the first 280,000 datapoints in training and the remaining 69,632 datapoints for test. For each of the empirical datatsets, we computed the accuracy, i.e., the fraction of test datapoints that were classified correctly by each of the algorithms and plotted the results in Fig. 6. In terms of the progress on accuracy across rounds, all the algorithms performed somewhat equally except for *Hybrid-DCA* with $S < p$ which missed updates from one of the workers. Though, *CoCoA+* apparently worked better on kddb, the margin was about 0.0005. However, in the long run all algorithms reached the same accuracy level. The progress on accuracy across time can be explained by the progress on duality gap across time as explained in Section 5.4.

5.6. Speedup

Speedup evaluates the improvement in performance of an algorithm as a function of the number of cores used. However, the exact definition of speedup varies in the literature. For example, *PassCoDe* computes speedup as the improvement in runtime to execute a fixed number of rounds, whereas *CoCoA+* uses a more refined notion of speedup for an optimization problem like RRM defined in Eq. (1) and computes speedup as the improvement in runtime to achieve a fixed level of duality gap. In our experiments, we evaluated the algorithms using both notions of speedup. Let $TR_A(p, t, r)$ and $TD_A(p, t, \epsilon)$ denote the time taken by an algorithm A using p nodes each with t cores to complete r rounds and to achieve duality gap ϵ, respectively. We formally define the two notions of speedup as follows[2]:

$$\text{Speedup}_A(p, t, r) = \frac{TR_A(p, t, r)}{TR_{Baseline}(1, 1, r)}, \quad \text{and}$$

$$\text{Proficiency}_A(p, t, \epsilon) = \frac{TD_A(p, t, \epsilon)}{TD_{Baseline}(1, 1, \epsilon)}$$

We ran sufficient rounds ($\geq$ 100) of each of the four algorithms and computed $\text{Speedup}_A(p, t, 100)$ and $\text{Proficiency}_A(p, t, 10^{-3})$ for all algorithms A except *Baseline*, as shown in the top two rows of Fig. 7. *PassCoDe* can be run only on a single node; so we varied only the number of cores. Because *CoCoA+* could use only 1 core per node. We ran *CoCoA+* and *Hybrid-DCA* with $t = 1$ cores on $p \in \{1, 2, 4, 8, 12, 16, 20, 24, 28, 31\}$ nodes and plotted the results separately. We also ran synchronous *Hybrid-DCA* on $p \in \{2, 4, 8\}$ nodes each with $t \in \{2, 4, 6, 8, 10, 12, 13, 14, 15, 16\}$ cores, $S = p, \gamma = 1$ and plotted the results separately for each p fixed and varying t. For $p = 8$ nodes and the same set of possible number of cores, we additionally plotted the results for asynchronous *Hybrid-DCA* with $S = 6, \Gamma = 4$.

Our first observation on the results of speedup experiments was that though proficiency and speedup were computed differently, they turned out to follow a similar trend. In fact, speedup was slightly higher than proficiency as the merging of parallel/distributed updates in every global round reduced the quality of the merged update in comparison with the pure updates of *Baseline*. We also observed that the speedup and proficiency of

[2] We use the term 'proficiency' to differentiate with a related term 'efficiency' widely used in the literature.

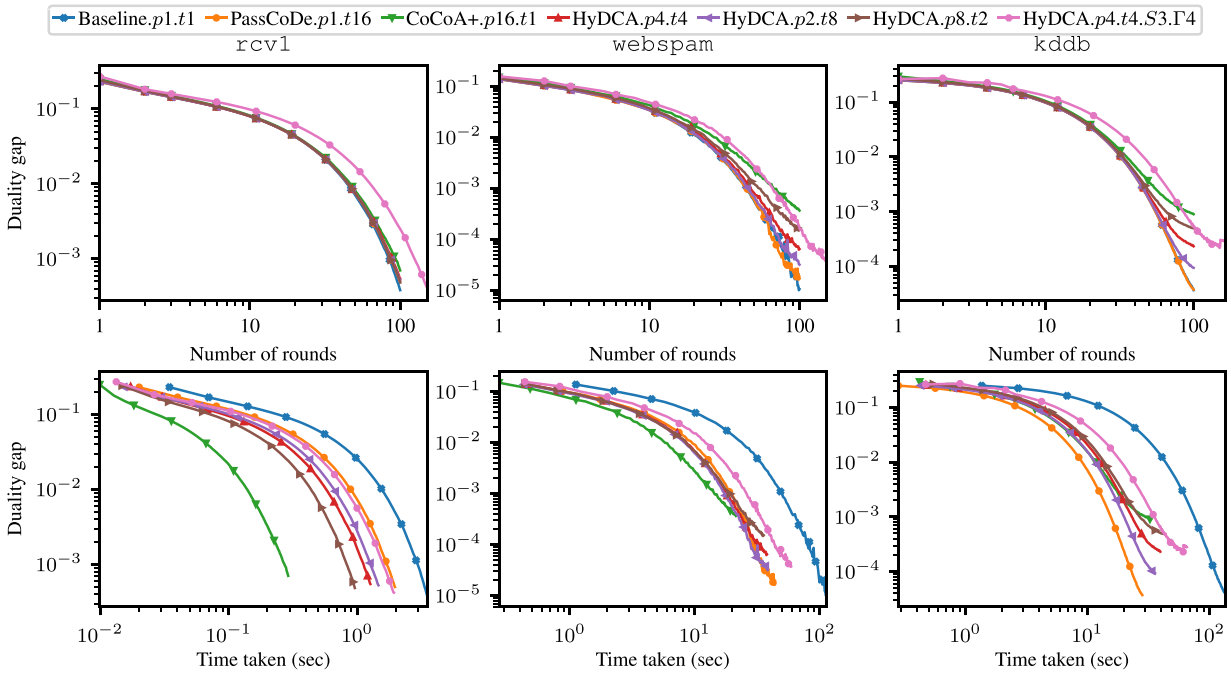


Fig. 5. Performance of different solvers on three datasets, `rcv1` (left column), `webspam` (middle column), and `kddb` (right column), in terms of the progress of the duality gap across the number of rounds (top row) and across the wall time taken (bottom row).

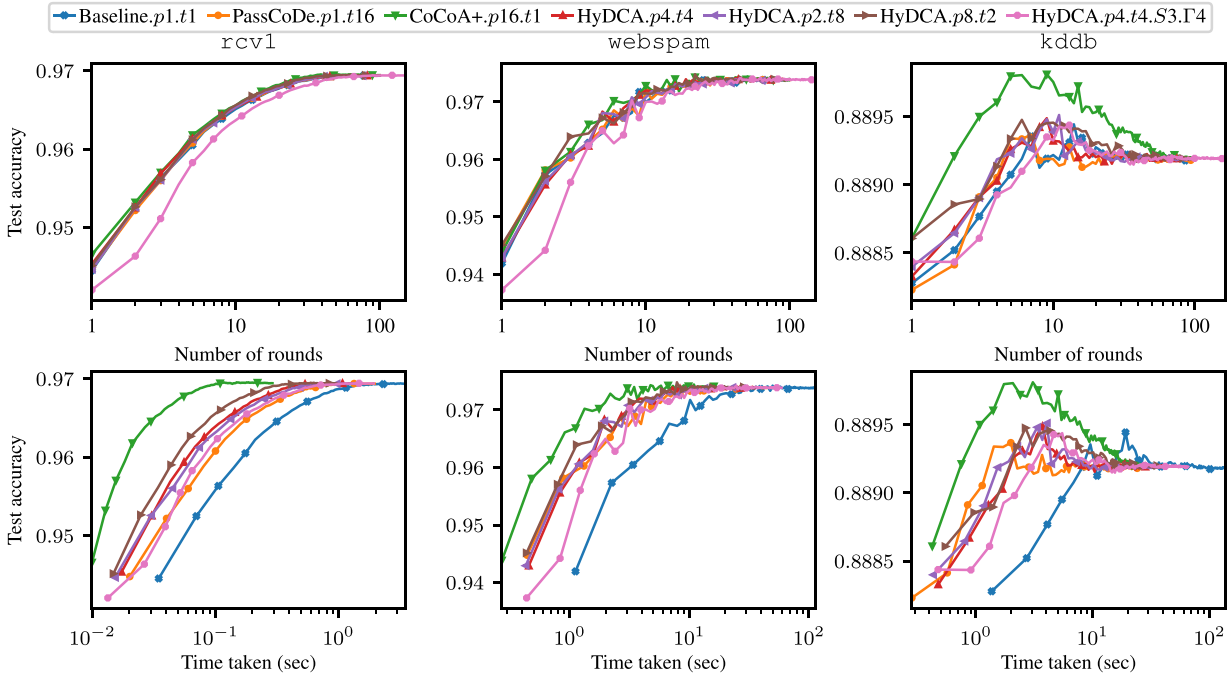


Fig. 6. Performance of different solvers on three datasets, `rcv1` (left column), `webspam` (middle column), and `kddb` (right column), in terms of the progress of the duality gap across the number of rounds (top row) and across the wall time taken (bottom row).

the algorithms followed a trend similar to the performance seen in Section 5.4, *i.e.*,*CoCoA+* performed better on datasets that were either sample-heavy or feature heavy but not both, *PassCoDe* performed better on dataset that was both sample-heavy and feature heavy, and *Hybrid-DCA* performed in between. Furthermore, asynchronous *Hybrid-DCA* ran slower than synchronous *Hybrid-DCA*.

While investigating why *Hybrid-DCA* was slower than our expectation, we observed that the overhead of OpenMP in maintaining the parallel threads was significant on the datasets `rcv1` and `webspam` as evident from the performance of *CoCoA.t*1 and *Hybrid-DCA.t*1 where their only difference was the overhead of maintaining a single OpenMP thread. However, this overhead was relatively small in comparison with the overall computation in the sample and feature heavy `kddb` dataset. However, we also

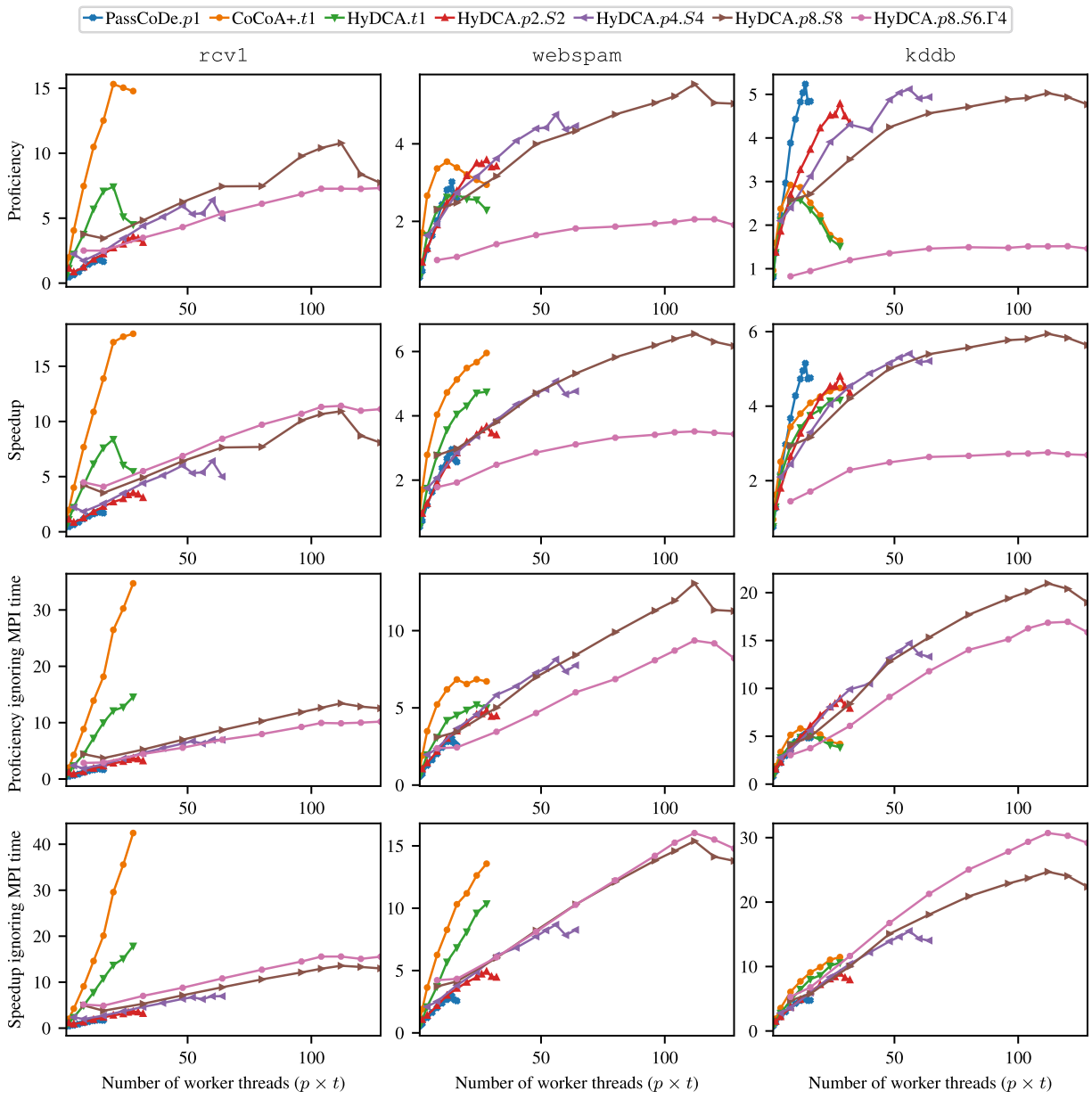


Fig. 7. Speedup of different parallel or distributed solvers with respect to the sequential implementation *Baseline*.

noticed that *CoCoA+* and *PassCoDe* did not scale well beyond 20 nodes and 14 cores, respectively.

Further investigation revealed two drawbacks of our implementation of MPI based inter-node communication. Firstly, MPI was inherently single threaded. Even for $t > 1$, for the whole duration when the workers in *Hybrid-DCA* sent their local updates to the master and received back the merged global update from the master, only the main thread in the workers was active and all other threads were idle. Though there has been academic research such as [21] on making MPI communications multi-threaded taking advantage of OpenMP threads available within the same task, the research has not been incorporated in the standard MPI implementations. This inherent drawback hindered *Hybrid-DCA* to take full advantage of all the cores available. The second drawback was the way we implemented asynchronous inter-node data transfers in *Hybrid-DCA* for $S < p$. In the absence of a collective operation for receiving updates from only a subset of all workers in the standard MPI implementation, we implemented such a collective operation using primitive MPI "send and receive" operations that lacked the performance improvement of *CoCoA+* that utilized the optimized MPI collectives such as `MPI_Allreduce`.

To mask the effect of the two drawbacks in the MPI, we drew the plots for speedup and proficiency, as shown in the bottom two rows of Fig. 7, after ignoring the time involved in MPI communications. Though *CoCoA+* also received the advantage of ignoring MPI time and showed better performance for datasets `rcv1` and `webspam`, it became totally outperformed by *Hybrid-DCA* on the sample-heavy `kddb` dataset. We also observed better performance of asynchronous *Hybrid-DCA* than synchronous *Hybrid-DCA*. In summary, as expected theoretically, we see almost the same speedup for a fixed number of total $p \times t$ cores irrespective of individual values of p and t. The drop in performance of *Hybrid-DCA* on $t > 14$ cores could be due to (1) the increase of time taken in an atomic memory update as the number of threads increase, and (2) for higher number of threads the delay bound of local updates may violate the assumption in (16). Both the aspects of atomic memory writes for higher number of threads have

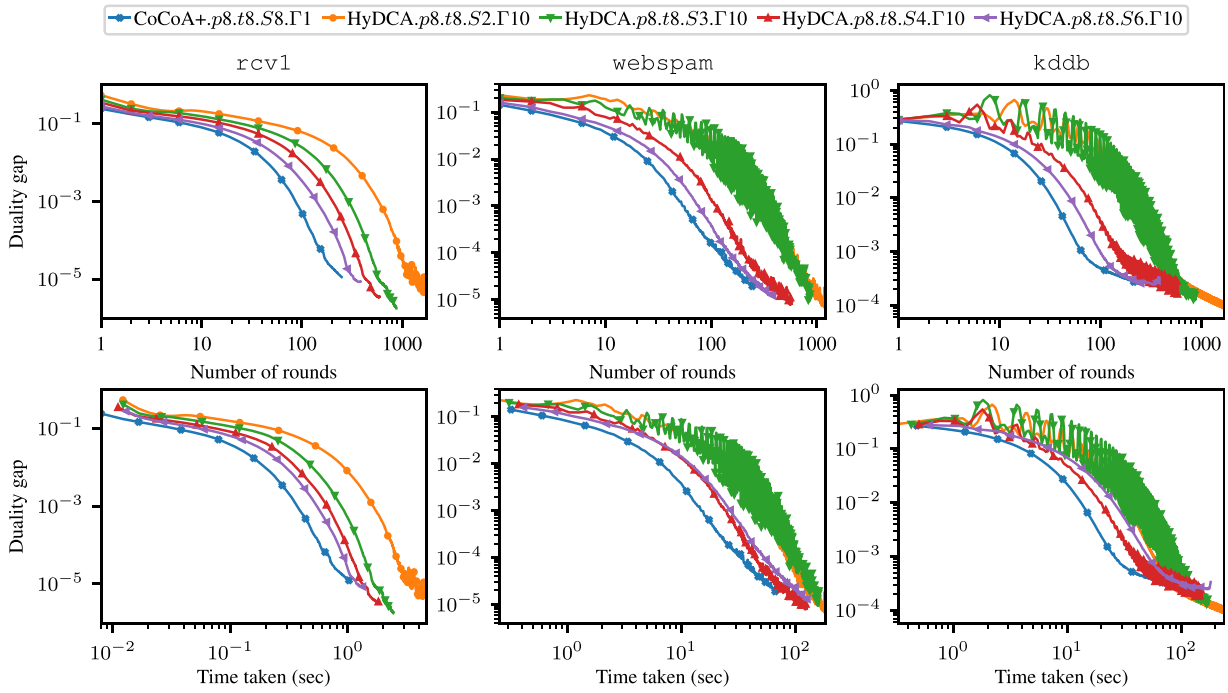


Fig. 8. Effect of varying S on $p = 8$ worker nodes, with Γ fixed at 10.

been investigated and worked around in a follow-up paper [26] of original *PassCoDe* paper [7], however, the incorporation of a similar fix in our implementation is out of scope for this paper.

We ignored MPI time for the computation of speedup and proficiency only for the experiments described in the section, for the experiments elsewhere we do include MPI time while measuring the wall time.

5.7. Effects of the parameter S

Fig. 8 shows the results of varying $S \in \{2, 3, 4, 6, 8\}$ with fixed $\Gamma = 10$ on $p = 8$ nodes each with $t = 8$ cores. When $S < p/2$, only a minority of the workers contributed in a round and the duality gap did not progress smoothly below some certain level. On the other hand, when at least half of the workers contributed in each round, it was possible to achieve the same duality level obtained using all the workers. However, the reduction in time per round was eventually eaten by the larger number of rounds required to achieve the same duality gap. Nevertheless, we will see in a later section that the approach was useful for HPC platforms with heterogeneous nodes, unlike ours, where the waiting for updates from all workers had larger penalty per round, or for the case, where the need was to run for a specified number of rounds and quickly achieved a reasonably good duality gap.

5.8. Effects of the parameter Γ

Fig. 9 shows the results of varying $\Gamma \in \{1, 2, 3, 4, 10\}$ with fixed $S = 6$ on $p = 8$ nodes each with $t = 8$ cores. We did not see much effect of Γ as the HPC platform used for our experiments had homogeneous nodes. Our experimentation showed that even if we used $\Gamma = 10$, the stale value at any worker was for at most 4 rounds. We expect to see a larger variance of staleness in case of heterogeneous nodes.

5.9. Effects of workload and processing power

When the HPC system had homogeneous nodes, our experimental results showed that asynchronous *Hybrid-DCA* did not change much when varying S and Γ. To show usefulness of asynchronous *Hybrid-DCA* on heterogeneous systems, we introduced imbalance in the setup for an experiment on $p = 4$ nodes each with $t = 4$ cores in two different ways: (1) by varying workload and (2) independently varying processing power. To vary workload, instead of distributing the datapoints equally on all 4 nodes, we loaded one of the nodes heavily by distributing the datapoints in the ratios 1:1:1:10. Similarly, to see the effect of heterogeneous processing speed, we introduced 10 s of delay in one of the nodes.

We compared the performance of synchronous *Hybrid-DCA* ($S = 4$) and asynchronous *Hybrid-DCA* ($S = 3, \Gamma = 10$) in the two imbalanced scenarios as well as on the usual balanced-load, homogeneous-speed scenario by plotting the progress on duality gap across rounds and wall time as shown in Fig. 10. For both scenarios of imbalanced load and heterogeneous speed the performance of synchronous *Hybrid-DCA* degraded significantly in terms of both across round and across time. However, the asynchronous *Hybrid-DCA* was able to mitigate the effects of imbalance completely in terms of performance across rounds in both scenarios and across wall time in heterogeneous speed scenario. For imbalanced load, asynchronous *Hybrid-DCA* was not able to improve performance as the time taken by the heavily loaded node dominated the overall time.

5.10. Performance on a big dataset

We experimented our hybrid algorithm on the big dataset `splicesite` of size about 280 GB and compared with the previous best algorithm *CoCoA+*. Because of the enormous size, the dataset could not be accommodated on a single node and hence *PassCoDe* could not be run on this dataset. In this experiment, we used the number of global iterations $G = 1000,000$. The results are shown in Fig. 11 where the progress of duality gap

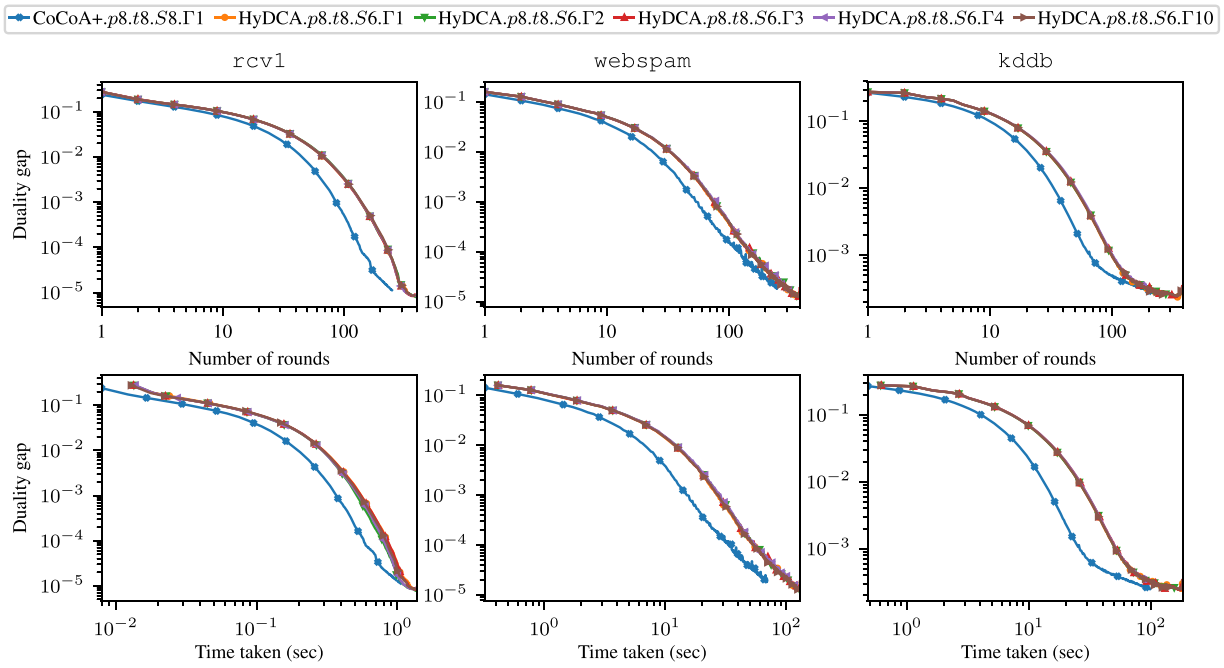


Fig. 9. Effect of varying Γ on $p = 8$ worker nodes, with S fixed at 8.

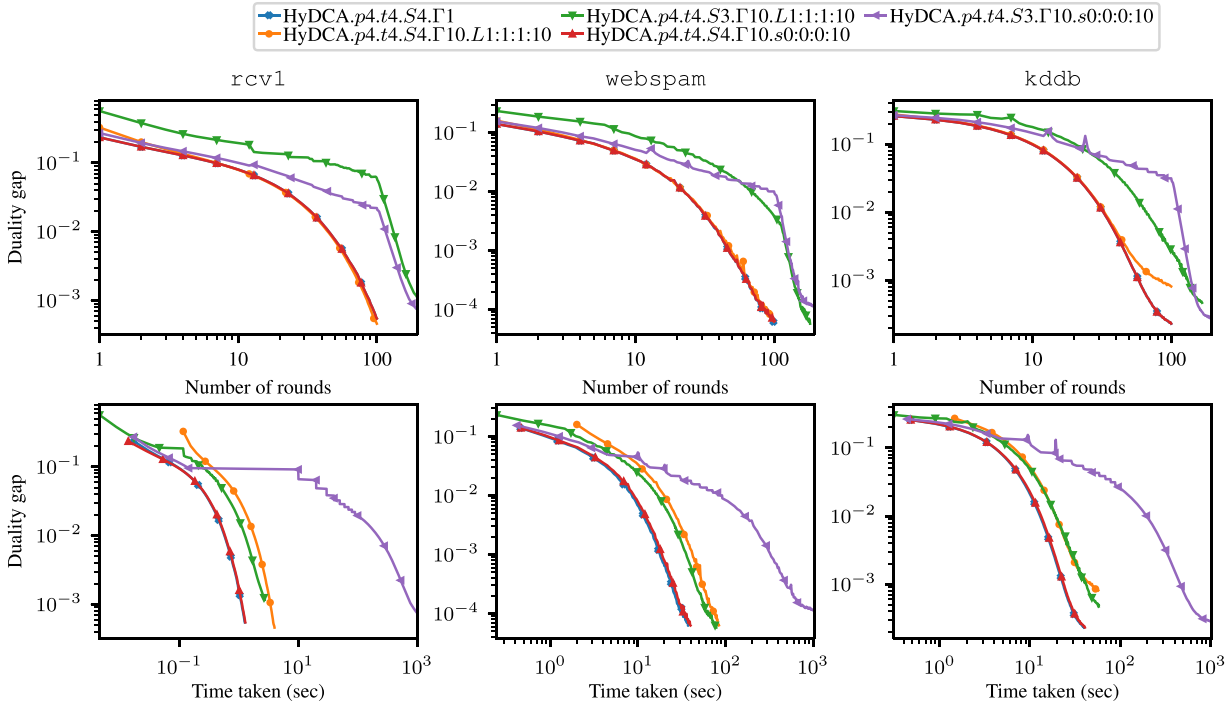


Fig. 10. Effect of varying workload and processing power on $p = 4$ worker nodes.

across the rounds of communication is shown on the left and across the wall time on the right. To achieve a duality gap of at least 10^{-5} on 16 nodes, *CoCoA+* took about 165 s. On the other hand *Hybrid-DCA* on 16 nodes each using 12 cores took about 29 seconds to achieve the same duality gap giving approximately 6-fold improvement, showing enough evidence about the scalability of our algorithm. One could also argue that *CoCoA+* can be run on all these $16 \times 12 = 192$ cores, treating each core as a distributed node. However, when we experimented with this mode of *CoCoA+*, we found that *CoCoA+* did not reach the duality gap 10^{-5} in a stipulated 100 rounds. We also experimented *CoCoA+* on $16 \times 8 = 128$ cores treating each core as a node and found out that though it performed better than 16×1 cores in the initial few rounds but worse in the later rounds. Moreover, it

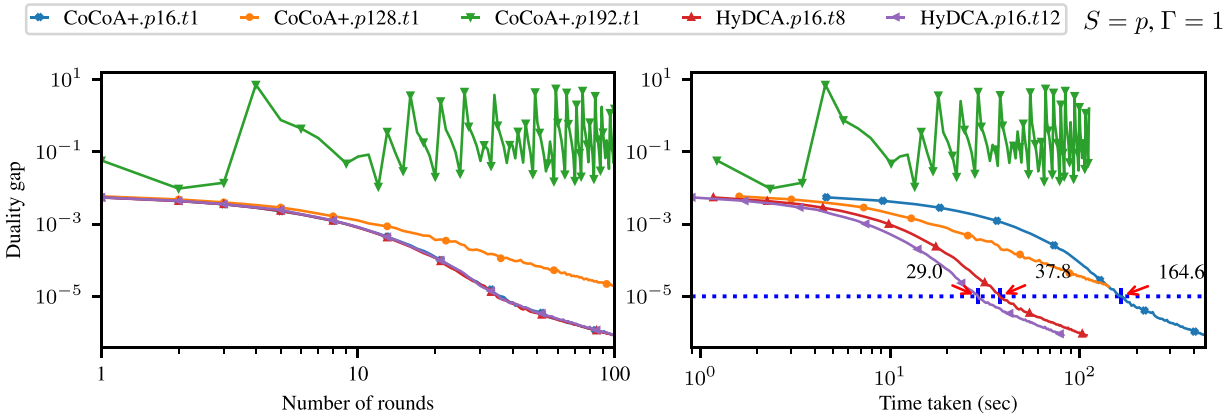


Fig. 11. Performance of *Hybrid-DCA* on big dataset `splicesite`.

was outperformed by *Hybrid-DCA* in terms of both the number of rounds and the time taken even on 16 nodes each using 8 cores.

6. Conclusions

In this paper, we have presented a hybrid parallel and distributed asynchronous stochastic dual coordinate ascent algorithm utilizing modern HPC platforms with many nodes of multi-core shared-memory systems. We analyze the convergence properties of this novel algorithm which uses asynchronous updates at two cascading levels: inter-cores and inter-nodes. Experimental results show that our algorithm is faster than the state-of-the-art distributed algorithms and scales better than the state-of-the-art parallel algorithms. The effectiveness of our approach in practical implementations can be increased further by a combination of (1) optimizing overhead of OpenMP threads, (2) incorporating multi-threaded implementation [21] of MPI operation, and (3) fixing the issues of larger delays in atomic memory writes for larger number of threads as given in [26].

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was partially supported by a National Science Foundation grant CCF-1514357, and a grant from National Institutes of Health 5K02DA043063-04 to J. Bi and two National Science Foundation grants NSF-1743418 and NSF-1843025 to S. Rajasekaran. This work utilized the computational resources of the NIH HPC Biowulf cluster (http://hpc.nih.gov).

References

[1] Alekh Agarwal, Olivier Chapelle, Miroslav Dudík, John Langford, A reliable effective terascale linear learning system, J. Mach. Learn. Res. 15 (1) (2014) 1111–1133.

[2] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, Distributed optimization and statistical learning via the alternating direction method of multipliers, Found. Trends Mach. Learn. 3 (1) (2011) 1–122.

[3] Chih-Chung Chang, Chih-Jen Lin, LIBSVM: A library for support vector machines, ACM Trans. Intell. Syst. Technol. (TIST) 2 (3) (2011) 27.

[4] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, Chih-Jen Lin, LIBLINEAR: A library for large linear classification, J. Mach. Learn. Res. 9 (2008) 1871–1874.

[5] Christina Heinze, Brian McWilliams, Nicolai Meinshausen, DUAL-LOCO: Distributing statistical estimation using random projections, in: Arthur Gretton, Christian C. Robert (Eds.), Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, in: Proceedings of Machine Learning Research, vol. 51, PMLR, Cadiz, Spain, 2016, pp. 875–883.

[6] Cho-Jui Hsieh, Kai-Wei Chang, Chih-Jen Lin, S. Sathiya Keerthi, Sellamanickam Sundararajan, A dual coordinate descent method for large-scale linear SVM, in: Proceedings of the 25th International Conference on Machine Learning, ICML, 2008, pp. 408–415.

[7] Cho-Jui Hsieh, Hsiang-Fu Yu, Inderjit S. Dhillon, PASSCoDe: Parallel asynchronous stochastic dual co-ordinate descent, in: Proceedings of the 32nd International Conference on Machine Learning, ICML, 2015.

[8] Martin Jaggi, Virginia Smith, Martin Takáč, Jonathan Terhorst, Sanjay Krishnan, Thomas Hofmann, Michael I Jordan, Communication-efficient distributed dual coordinate ascent, in: Advances in Neural Information Processing Systems, NIPS, 2014, pp. 3068–3076.

[9] Jason D. Lee, Qihang Lin, Tengyu Ma, Tianbao Yang, Distributed stochastic variance reduced gradient methods by sampling extra data with replacement, J. Mach. Learn. Res. 18 (122) (2017) 1–43, URL http://jmlr.org/papers/v18/16-640.html.

[10] Ji Liu, Stephen J. Wright, Christopher Ré, Victor Bittorf, Srikrishna Sridhar, An asynchronous parallel stochastic coordinate descent algorithm, J. Mach. Learn. Res. 16 (2015) 285–322.

[11] Chenxin Ma, Jakub Konečný, Martin Jaggi, Virginia Smith, Michael I. Jordan, Peter Richtárik, Martin Takáč, Distributed optimization with arbitrary local solvers, Optim. Methods Softw. 32 (4) (2017) 813–848, http://dx.doi.org/10.1080/10556788.2016.1278445.

[12] Ryan Mcdonald, Mehryar Mohri, Nathan Silberman, Dan Walker, Gideon S. Mann, Efficient large-scale distributed training of conditional maximum entropy models, in: Advances in Neural Information Processing Systems, 2009, pp. 1231–1239.

[13] Brian McWilliams, Christina Heinze, Nicolai Meinshausen, Gabriel Krummenacher, Hastagiri P. Vanchinathan, LOCO: Distributing ridge regression with random projections, 2014, arXiv preprint arXiv:1406.3469.

[14] Eric Moulines, Francis R. Bach, Non-asymptotic analysis of stochastic approximation algorithms for machine learning, in: Advances in Neural Information Processing Systems, 2011, pp. 451–459.

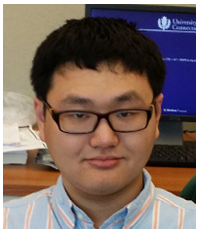
[15] Deanna Needell, Rachel Ward, Nati Srebro, Stochastic gradient descent, weighted sampling, and the randomized kaczmarz algorithm, in: Advances in Neural Information Processing Systems, 2014, pp. 1017–1025.

[16] Zhimin Peng, Yangyang Xu, Ming Yan, Wotao Yin, ARock: An algorithmic framework for asynchronous parallel coordinate updates, SIAM J. Sci. Comput. 38 (5) (2016) A2851–A2879, http://dx.doi.org/10.1137/15M1024950.

[17] Peter Richtárik, Martin Takáč, Distributed coordinate descent method for learning with big data, J. Mach. Learn. Res. 17 (1) (2016) 2657–2681, URL http://dl.acm.org/citation.cfm?id=2946645.3007028.

[18] Shai Shalev-Shwartz, Tong Zhang, Stochastic dual coordinate ascent methods for regularized loss minimization, J. Mach. Learn. Res. 14 (1) (2013) 567–599.
[19] Shai Shalev-Shwartz, Tong Zhang, Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization, Math. Program. 1 (155) (2016) 105–145.
[20] Ohad Shamir, Nathan Srebro, Tong Zhang, Communication-Efficient Distributed Optimization using an Approximate Newton-type Method, in: Proceedings of the 31th International Conference on Machine Learning, ICML, Beijing, China, 21-26 June 2014, 2014, pp. 1000–1008.
[21] Min Si, Antonio J. Peña, Pavan Balaji, Masamichi Takagi, Yutaka Ishikawa, MT-MPI: Multithreaded MPI for many-core environments, in: Proceedings of the 28th ACM International Conference on Supercomputing, ACM, 2014, pp. 125–134.
[22] Martin Takáč, Avleen Bijral, Peter Richtarik, Nati Srebro, Mini-Batch primal and dual methods for SVMs, in: Proceedings of the 30th International Conference on Machine Learning, ICML-13, 2013, pp. 1022–1030.
[23] Tianbao Yang, Trading computation for communication: Distributed stochastic dual coordinate ascent, in: Advances in Neural Information Processing Systems, 2013, pp. 629–637.
[24] Hsiang-Fu Yu, Fang-Lan Huang, Chih-Jen Lin, Dual coordinate descent methods for logistic regression and maximum entropy models, Mach. Learn. 85 (1) (2011) 41–75.
[25] Tong Zhang, Solving large scale linear prediction problems using stochastic gradient descent algorithms, in: Proceedings of the Twenty-First International Conference on Machine Learning, ACM, 2004, p. 116.
[26] Huan Zhang, Cho-Jui Hsieh, Fixing the convergence problems in parallel asynchronous dual coordinate descent, in: Sixteenth IEEE International Conference on Data Mining, ICDM, IEEE, 2016, pp. 619–628.
[27] Ruiliang Zhang, James Kwok, Asynchronous distributed ADMM for consensus optimization, in: Proceedings of the 31st International Conference on Machine Learning, ICML, 2014, pp. 1701–1709.
[28] Yuchen Zhang, Lin Xiao, Communication-efficient distributed optimization of self-concordant empirical loss, in: Large-Scale and Distributed Optimization, Springer, 2018, pp. 289–341.

Soumitra Pal received a B.E.degree in Computer Science and Technology from Bengal Engineering and Science University, Howrah, India in 2000 and M.Tech. and Ph.D. degrees in Computer Science and Engineering from IIT Bombay in 2007 and 2013, respectively. He was a postdoctoral research fellow at the University of Connecticut, USA during 2014–6. He also worked with Texas Instruments, India as a Software Design Engineer during 2000–5. His research interests are in algorithm design, optimization, bioinformatics and parallel and distributed systems.



Tingyang Xu is a Ph.D. student in the Department of Computer Science and Engineering, the University of Connecticut. Since 2011, he is a member of the Laboratory of Machine Learning and Health Informatics led by Professor Jinbo Bi. His main areas of research interests are machine learning in longitudinal and time series data and high performance computing for optimization problems. He is also a student member of the Institute of Electrical and Electronics Engineers (IEEE).

Tianbao Yang is an Assistant Professor of the Computer Science Department at the University of Iowa. He received the Ph.D. degree in Computer Science from Michigan State University in 2012. He worked as a researcher in GE Global Research from 2012 to 2013 and in NEC Laboratories America, Inc. from 2013 to 2014. He has board interests in machine learning and has focused on several research topics, including social network analysis and large scale optimization in machine learning. He has won the Mark Fulk Best student paper award at 25th Conference on Learning Theory (COLT) in 2012. He also served as program committee for several conferences, including AAAI'15, AAAI'12, CIKM'12, '13, IJCAI'13, ACML'12.

Sanguthevar Rajasekaran received his M.E. degree in Automation from the Indian Institute of Science (Bangalore) in 1983, and his Ph.D. degree in Computer Science from Harvard University in 1988. Currently he is the UTC Chair Professor of Computer Science and Engineering at the University of Connecticut and the Director of Booth Engineering Center for Advanced Technologies (BECAT). Before joining UConn, he has served as a faculty member in the CISE Department of the University of Florida and in the CIS Department of the University of Pennsylvania. During 2000–2002 he was the Chief Scientist for Arcot Systems. His research interests include Parallel Algorithms, Bioinformatics, Data Mining, Randomized Computing, Computer Simulations and Combinatorial Optimization. He has published over 150 articles in journals and conferences. He has co-authored two texts on algorithms and co-edited four books on algorithms and related topics. He is an IEEE Fellow and an elected member of the Connecticut Academy of Science and Engineering.

Jinbo Bi received a Ph.D. degree in mathematics from Rensselaer Polytechnic Institute, USA, and a master degree in Electrical Engineering and Automatic Control from Beijing Institute of Technology, China. She is an associate professor of Computer Science and Engineering at the University of Connecticut. Prior to her current appointment, she worked with Siemens Medical Solutions on computer aided diagnosis research and Partners Healthcare on clinical decision support systems. Her research interests include machine learning, data mining, bioinformatics and biomedical informatics.