Distributed Online Optimization over a Heterogeneous Network with Any-Batch Mirror Descent

Nima Eshraghi [1] **Ben Liang** [1]

Abstract

In distributed online optimization over a computing network with heterogeneous nodes, slow nodes can adversely affect the progress of fast nodes, leading to drastic slowdown of the overall convergence process. To address this issue, we consider a new algorithm termed Distributed Any-Batch Mirror Descent (DABMD), which is based on distributed Mirror Descent but uses a fixed per-round computing time to limit the waiting by fast nodes to receive information updates from slow nodes. DABMD is characterized by varying minibatch sizes across nodes. It is applicable to a broader range of problems compared with existing distributed online optimization methods such as those based on dual averaging, and it accommodates time-varying network topology. We study two versions of DABMD, depending on whether the computing nodes average their primal variables via single or multiple consensus iterations. We show that both versions provide strong theoretical performance guarantee, by deriving upperbounds on their expected dynamic regret, which capture the variability in minibatch sizes. Our experimental results show substantial reduction in cost and acceleration in convergence compared with the known best alternative.

1. Introduction

In many practical applications, system parameters and cost functions vary over time. For example, in online machine learning, data samples arrive dynamically, so that the learning model is progressively updated. Therefore, online optimization has recently attracted significant attention as an important tool for modeling various classes of problems with uncertainty, including cloud computing (Lin et al., 2013), networking (Shi et al., 2018), and machine learning (Shalev-Shwartz, 2012).

Furthermore, emerging computing and machine learning applications often demand huge amounts of data and computation, which exacerbates the need for distributed computation, to avoid overburdening a single computing node and to provide robustness to failures. Yet, a central challenge to distributed computation is how to handle the uncertainty involved in the processing time of heterogeneous computing nodes. In a large distributed computing network, the finishing times of different computing nodes can vary greatly across the network and time, and they can be highly random due to the variation in available computing resource and dynamics of the workload. In the design of these networks, it is difficult to capture such uncertainty in an offline setting, while online optimization techniques naturally accommodate unpredictable variations in the cost functions.

In this paper, we study the problem of online optimization over a distributed computing network with time-varying topology. The network consists of heterogeneous computing nodes, and the processing speed varies among the nodes and over time. At each time instant, there is a global cost function in the network, which is generally defined and is time-varying. An example of such a global cost is the prediction loss. Individual nodes communicate locally to collaboratively supplement their incomplete knowledge about the time-varying global cost. Our aim is to minimize the total accumulated cost over a finite time interval.

The above computing model can be viewed as an abstraction for several important practical cases. For instance, consider a parallel processing system with unknown processing times. The computing nodes may be parallel processors or virtual machine instances in a public cloud or mobile edge servers. A number of factors contribute to not knowing the processing time on these processors. For example, an edge server may be shared among offloaded user tasks and network service tasks, so that only a fraction of the processing resource may be available to serve the user tasks. Another example is networked sensors and mobile devices in a machine learning application, where the goal is to fit a model to a large dataset. Different sensors and devices have differing

[1] Department of Electrical and Computer Engineering, University of Toronto, Toronto, Canada. Correspondence to: Nima Eshraghi <nima.eshraghi@mail.utoronto.ca>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020. Copyright 2020 by the author(s).

computing speed, and thus, slow nodes can cause significant delay that adversely affects the overall performance. Closely related to this case is the straggler problem in distributed learning networks (Tandon et al., 2017). A careful design is required to handle slow learners and stragglers.

Prior studies on distributed online optimization utilize a variety of solution methods, including gradient descent (Mateos-Nunez & Cortés, 2014), and mirror descent (Shahrampour & Jadbabaie, 2017). Mirror descent uses Bregman divergence, which generalizes the Euclidean norm used in the projection step of gradient descent, thus acquiring expanded applicability to a broader range of problems. In addition, Bregman divergence is mildly dependent on the dimension of decision variables (Beck & Teboulle, 2003). Thus, mirror descent is optimal among first-order methods when the decision variables' dimension is high (Duchi et al., 2010). In this work we focus on the mirror descent approach.

Most prior studies assume homogeneous computing nodes, so that the amount of data processed in each round is fixed (Hosseini et al., 2013; Mateos-Nunez & Cortés, 2014; Nedich et al., 2015; Shahrampour & Jadbabaie, 2017; Tsianos & Rabbat, 2016; Xiao, 2010). However, as explained above, heterogeneity is often unavoidable in modern computing networks. Therefore, in this work we instead consider an any-batch approach, fixing the per-round processing time and allowing each computing node to complete as much work as it can in each round. This forces all nodes to report their accomplished work when the fixed time interval expires, thus preventing slow computing nodes from holding up the progress of faster ones. However, fixing the processing time results in varying workload among the heterogeneous nodes, which complicates system design and performance analysis. As far as we are aware, this work is the first to consider this approach for mirror descent, with a theoretical analysis on its regret bound.

Contributions. In this paper, we consider the problem of distributed online optimization over a computing network, taking into account heterogeneity among computing nodes in terms of processing speed. We present an online optimization approach termed Distributed Any-Batch Mirror Descent (DABMD), which uses fixed per-round processing time to limit the waiting time by fast nodes to receive information update from slow nodes. The term any-batch is due to the fact that our approach allows varying batch sizes of data processed by computing nodes within each round. We study two versions of DABMD, depending on whether the computing nodes average their primal variables via single or multiple consensus iterations. Our main contribution is in analyzing the performance of DABMD in terms of regret, which measures the difference between the accumulated cost of the online algorithm and that of an optimal offline counterpart. In contrast to prior studies that focus on the static regret (Ferdinand et al., 2019; Hosseini et al., 2013; Mateos-Nunez & Cortés, 2014; Nedich et al., 2015; Tsianos & Rabbat, 2016; Xiao, 2010), where the offline comparison target is static and hence highly suboptimal, we establish upper bounds on the dynamic regret, which provides a stronger performance guarantee. Furthermore, in contrast to earlier studies (Ferdinand et al., 2019; Hosseini et al., 2013; Nedich et al., 2015; Shahrampour & Jadbabaie, 2017; Tsianos & Rabbat, 2016; Xiao, 2010) where a fixed network topology is considered, we allow a time-varying network topology. Experimental results show a substantially accelerated convergence speed compared with the known best alternative from (Shahrampour & Jadbabaie, 2017)

2. Related Works

The present paper lies at the intersection between two important bodies of works, distributed computation and online optimization. Here we review the most relevant studies in these two areas.

2.1. Distributed Computation

The problem of distributed computation over a network has been extensively studied since the seminal work of (Tsitsiklis et al., 1986). Distributed optimization techniques can be categorized based on their computing network topology, either master-worker (Boyd et al., 2011) or fully distributed, which are characterized by the lack of centralized access to information. Our main focus is on methods in the second category. These methods are particularly attractive since they are robust to communication and node failures and are simple to implement.

Of particular interest is distributed gradient-based methods. In (Nedic & Ozdaglar, 2009), the distributed subgradient method is presented and its convergence rate is analyzed for a time-varying network. In (Duchi et al., 2011), the dual averaging method for distributed optimization is proposed and the effect of network parameters on the convergence behavior is investigated. Distributed dual averaging under various scenarios of delayed subgradient information and quantized communication error is studied in (Tsianos & Rabbat, 2012; Wang et al., 2014; Yuan et al., 2012). All the aforementioned methods are first-order algorithms requiring only the calculation of subgradients and projections. This is especially important in some applications such as large-scale learning where first-order methods are preferable to higher order approaches (Bottou & Bousquet, 2008). However, despite the benefit of simplicity of first-order methods, it is challenging to generate projections for some common objective functions. An example of such objective functions is the entropy-based loss with the constraint set of unit simplex (Beck & Teboulle, 2003).

The mirror descent algorithm (Beck & Teboulle, 2003; Nemirovsky & Yudin, 1983) generalizes the projection step by using Bregman divergence, consequently gaining applicability to a wide range of problems. Distributed versions of mirror descent under various settings are presented in (Li et al., 2016; Rabbat, 2015; Raginsky & Bouvrie, 2012; Shahrampour & Jadbabaie, 2017; Wang et al., 2018; Xi et al., 2014; Yuan et al., 2018). However, all of these works, except (Shahrampour & Jadbabaie, 2017), consider only time-invariant cost functions in an *offline* setting.

2.2. Onilne Optimization

Online optimization techniques take into account the dynamics in a system and allow the cost function to be time-varying (Shalev-Shwartz, 2012; Zinkevich, 2003). In (Mateos-Nunez & Cortés, 2014), distributed online optimization over a time-varying network topology is studied using the subgradients of local functions. The subgradient approach uses Euclidean norms in the projection step. In contrast, our work based on mirror descent considers a Bregman divergence function, and thus is applicable to a broader range of problems.

For distributed online optimization, the dual averaging approach is studied in (Hosseini et al., 2013; Xiao, 2010). Extensions include approximate dual averaging (Tsianos & Rabbat, 2016), and a primal-dual approach (Nedich et al., 2015). However, these works assume homogeneous computing nodes. The authors in (Ferdinand et al., 2019), consider the straggler issue in learning networks, and propose any-batch dual averaging approach. However, the network topology is fixed over time, which limits its applicability to modern computing systems. Finally, all of these works analyze only the static regret, while we provide a bound on the dynamic regret, which takes into account the time variant nature of the minimizer.

The work most closely related to ours is (Shahrampour & Jadbabaie, 2017), which studies distributed mirror descent for online optimization over a network with static topology and homogeneous nodes in terms of processing capability. Our work may be viewed as a generalization of (Shahrampour & Jadbabaie, 2017) in two directions. First, we account for the heterogeneity among the computing nodes, which motivates an any-batch design that is robust to slow computing nodes, to prevent long waiting time by fast nodes until slow nodes finish their work. Second, we allow a time-varying network topology. Thus, our approach is more suited to practical computing networks, e.g., mobile cloud systems and networked sensors, where heterogeneity and time-varying topology are unavoidable.

3. Network Model and Problem Formulation

3.1. Heterogeneous Time-Varying Computing Network

We consider a distributed computing network with heterogeneous nodes in a time-slotted setting. The communication topology is assumed to be time-varying, and it is modeled by an undirected graph $\mathcal{G}_t = (\mathcal{V}, \mathcal{E}^t)$, where $\mathcal{V} = \{1, \ldots, n\}$ and $\mathcal{E}^t \subset \mathcal{V} \times \mathcal{V}$ denote the set of nodes and edges at time t, respectively. Each node $i \in V$ can generally represent a computing unit, e.g., data center, or mobile device.

Computing nodes exchange information according to the topology graph $\mathcal{G}_t$ at time t. Each computing node i assigns a non-negative weight P_{ij}^t to the received information from node j. To ensure individual nodes sufficiently communicate with each other, we impose several standard assumptions on topology graph $\mathcal{G}_t$ and the weighting rule (Nedic & Ozdaglar, 2009):

(a) (*Weight Rule*) There exists a scalar η with $0 < \eta < 1$ such that for all $i, j \in \mathcal{V}$, and all $t \geq 0$, $P_{ij}^t \geq \eta$, if $(i,j) \in \mathcal{E}^t$, and $P_{ij}^t = 0$, otherwise.

(b) (*Doubly Stochastic*) The weight matrix P^t is doubly stochastic, i.e.,

$$\sum_{j=1}^{n} P_{ij}^t = \sum_{i=1}^{n} P_{ij}^t = 1, \quad \forall t \geq 0. \tag{1}$$

(c) (*Connectivity*) The graph $(\mathcal{V}, \bigcup_{t \geq k} \mathcal{E}^t)$ is connected for all k. Furthermore, there exists an integer $B \geq 1$ such that $\forall (i,j) \in \mathcal{V} \times \mathcal{V}$, we have

$$(i,j) \in \mathcal{E}^t \cup \mathcal{E}^{t+1} \cup \cdots \cup \mathcal{E}^{t+B-1}, \quad \forall t \geq 0. \tag{2}$$

This guarantees that every node i receives the share of information processed by any other node j at least once every B consecutive time-slots.

3.2. Distributed Online Optimization

We consider the case where data samples arrive at the system over time. Each data sample i is represented by (ω_i, z_i), where $\omega_i \in \Omega \subseteq \mathbb{R}^d$ and $z_i \in \mathbb{R}$. There is a cost function $f(x, (\omega, z))$ associated with every data sample, where $x \in \mathcal{X}$ denotes a decision variable taken from a compact and convex set $\mathcal{X}$. The cost function is generally defined and may represent e.g., a measure of prediction accuracy to fit a learning model. For example, logistic regression is obtained by setting $f(x, (\omega, z)) = \log(1 + \exp(-z x^T \omega))$. For brevity, we use $f(x, \omega)$ instead of $f(x, (\omega, z))$ in the rest of the paper. The cost function $f(x, \omega)$ is convex in x, for all $\omega \in \mathbb{R}^d$. In addition, we assume $f(x, \omega)$ is Lipschitz continuous on $\mathcal{X}$ with constant L, i.e.,

$$|f(x_1, \omega) - f(x_2, \omega)| \leq L \parallel x_1 - x_2 \parallel, \forall x_1, x_2 \in \mathcal{X}. \tag{3}$$

This implies that $f(x, \omega)$ has bounded gradients, i.e., $\|\nabla f(x, \omega)\|_* \leq L$ for all $\omega \in \Omega$ and $x \in \mathcal{X}$, where $\|\cdot\|_*$ is the dual norm of $\|\cdot\|$.

In distributed online optimization, the computing nodes individually process the data samples to minimize some global cost function in a round-by-round manner. Each round t corresponds to time slot t and consists of computation and communication phases. Let $b_{i,t}$ be the local minibatch size, i.e., the number of samples that node $i \in V$ is able to process in round t. The value of $b_{i,t}$ depends on the processing speed of node i within the computation time.

The local cost function is defined as the sum of the costs incurred for the locally computed samples. We denote the local cost function of node i at round t by $f_{i,t} : \mathcal{X} \to \mathbb{R}$, i.e.,

$$f_{i,t}(x) = \sum_{s=1}^{b_{i,t}} f(x, \omega_{i,t}^s), \tag{4}$$

where the data samples $\omega_{i,t}^s$, $s = 1, \ldots, b_{i,t}$ are computed locally, and collectively form the local minibatch of node i at round t.

At each round t, after each node i submits a local decision $x_{i,t}$, a minibatch of $b_{i,t}$ samples are processed, and the node suffers a corresponding cost of $f_{i,t}(x_{i,t}) = \sum_{s=1}^{b_{i,t}} f(x_{i,t}, \omega_{i,t}^s)$. Thus, these cost functions are time varying. The random samples are assumed to be independent and identically distributed with an unknown and fixed probability distribution.

We are interested in an optimization problem with a global cost function, represented by $f_t : \mathcal{X} \to \mathbb{R}$ at round t. It is based on the local functions that are distributed over the entire network, i.e.,

$$f_t(x) = \sum_{i=1}^{n} f_{i,t}(x). \tag{5}$$

For distributed minimization of the global cost, every computing node i maintains a local estimate of the global minimizer $y_{i,t}$ as well as a local decision vector $x_{i,t}$. Individual nodes perform local averaging of the current decision collected from their neighbors, using the weight matrix P^t. Local estimates $y_{i,t}$ are updated accordingly, which are then used together with the next minibatch $b_{i,t+1}$ to compute the local decision $x_{i,t+1}$ in the next round.

The goal of an online algorithm is to minimize the total accrued cost over a finite number of rounds T. The quality of decisions is measured in terms of a popular metric, called regret, which is the difference between the total cost incurred by the online algorithm and that of an optimal offline solution, where all cost functions are known in advance. A successful online algorithm closes the gap between the online decisions and the offline counterpart, when normalized by T. In other words, a successful online algorithm sustains a regret sublinear in T.

Studies on online learning often focus on the static regret, where offline solution is fixed over time, i.e., $x^* = \text{argmin}_{x \in \mathcal{X}} \sum_{t=1}^{T} f_t(x)$. Unfortunately, this offline comparison target is often highly suboptimal in most practical streaming settings, e.g., x_t^* may correspond to video frames in a dynamic environment which by nature are highly variable over time (Hall & Willett, 2015). Furthermore, in this paper we study a fully dynamic scenario, where both the cost functions and the network topology are time-varying. Therefore, to capture the full potential of dynamics in the network, we focus on the notion of dynamic regret (Zinkevich, 2003), which is defined by

$$\text{Reg}_T^d = \frac{1}{n} \sum_{i=1}^{n} \sum_{t=1}^{T} f_t(x_{i,t}) - \sum_{t=1}^{T} f_t(x_t^*). \tag{6}$$

where $x_t^* = \text{argmin}_{x \in \mathcal{X}} f_t(x)$ is a minimizer of the global cost at round t.

It is well known that the in a dynamic setting, the problem may be intractable due to arbitrary fluctuation in the functions (Zhang et al., 2017). However, it is possible to upper bound the dynamic regret in terms of certain regularity of the sequence of minimizers. A popular quantity to represent such regularity is

$$A_T = \sum_{t=1}^{T} \parallel x_{t+1}^* - x_t^* \parallel, \tag{7}$$

which represents the path length of the sequence. Thus, A_T serves as a complexity measure which assesses the hardness of an online problem. For instance, it is shown in (Shahrampour & Jadbabaie, 2017) that the dynamic regret of distributed online mirror descent for a convex function in a homogeneous network can be bounded by $O(\sqrt{T(A_T + 1)})$.

For the analysis of dynamic regret of DABMD, we face multiple challenges that need to be carefully considered. In particular, heterogeneity adds substantial challenge to the algorithm design. Since the computation time is fixed for all nodes, the minibatch size $b_{i,t}$ is a random variable. This is in contrast to the traditional methods, where the minibatch size is fixed among the computing nodes and over time.

Before delving into the details of DABMD, we proceed by stating several standard definitions. Further discussion on some preliminary information is given in App. A.

Definition 1: A subdifferentiable function $r : \mathbb{R}^d \to \mathbb{R}$ is σ-strongly convex with respect to a norm $\parallel . \parallel$, if there exists a positive constant σ such that

$$r(x) \geq r(y) + \langle \nabla r(y), x - y \rangle + \frac{\sigma}{2} \parallel x - y \parallel^2, \forall x, y, \tag{8}$$

where $\nabla r(.)$ stands for the subgradient of function $r(.)$.

Definition 2: Bregman divergence with respect to the function $r(.)$ is defined as

$$D_r(x, y) = r(x) - r(y) - \langle \nabla r(y), x - y \rangle. \tag{9}$$

Bregman divergences are a general class of distance-measuring functions, which contains Euclidean norms and Kullback-Leibler divergence as two special cases.

4. Distributed Any-Batch Mirror Descent

Now we present DABMD, a distributed online optimization approach for an evolving dynamic networks with heterogeneous computing nodes. We bound the performance of the proposed DABMD algorithm in terms of its expected dynamic regret.

4.1. Algorithm Description

DABMD uses mirror descent in its core as an optimization engine, powered by averaging consensus, enabling nodes to collaboratively approximate a globally optimal solution through local interactions. In contrast to traditional distributed methods, where a classical but impractical requirement is that each computing node processes an equal amount of workload in a given time, DABMD is designed to accommodate heterogeneity in a computing system. The computing time for all nodes is fixed, and thus, faster nodes would no longer need to wait for slower ones to report their accomplished work. As a result, the performance of DABMD is no longer limited by the variability in the batch sizes. This is particularly important in a large network consisting of many nodes with differing processing capabilities, where slow nodes hold up the network, often at a great cost to the overall system performance. Next, we describe DABMD in detail and establish a bound on its performance through regret analysis. Every round of DABMD is comprised of three major steps:

Any-Batch Computation: The computation time in each round is set to a fixed value for all nodes. Individual nodes compute the gradient of as many samples as they can within the fixed computation time. In particular, every computing node i computes $b_{i,t}$ gradients of $f(x, \omega)$, evaluated at the local estimate of the global minimizer $x = y_{i,t}$. Resulting from heterogeneity, the computing nodes process varying workload within each round. This implies that the batch size $b_{i,t}$ is a random variable, which is in contrast to classical approaches based on a fixed minibatch size over time and across the network. By the end of each round, each node computes the local minibatch gradient, i.e.,

$$g_{i,t} = \frac{1}{b_{i,t}} \sum_{s=1}^{b_{i,t}} \nabla_y f(y_{i,t}, \omega_{i,t}^s). \tag{10}$$

Local Decision Update: After computing the local minibatch gradient, each computing node i performs the following update on the local decision:

$$x_{i,t+1} = \operatorname*{argmin}_{x \in \mathcal{X}} \Big\{ \langle x, g_{i,t} \rangle + \frac{1}{\alpha_t} D_r(x, y_{i,t}) \Big\}, \tag{11}$$

where $\{\alpha_t\}$ is a sequence of positive and non-increasing step sizes, and $D_r(.,.)$ is the Bregman divergence corresponding to the regularization function. Recall that $x_{i,t}$ and $y_{i,t}$ are respectively the local decision and the local estimate of the global minimizer at round t. Thus, (11) suggests that every node i aims to stay close to the locally approximated global minimizer $y_{i,t}$ as measured by Bregman divergence, while taking a step in a direction close to $g_{i,t}$ to reduce the local cost at the current round.

Consensus Averaging: Consensus averaging is a mechanism to coordinate and synchronize the decision variables at different computing nodes. After each node has processed its share of local information, and updated its local decision, it sends a message to the neighboring nodes. At round t, after receiving the messages from all neighbors, computing node i updates its estimate of the global minimizer by

$$y_{i,t} = \sum_{j=1}^{n} P_{ij}^t x_{j,t}. \tag{12}$$

The objective of this step is to provide an opportunity for each node to obtain some information about the global cost. It allows nodes to cooperatively approximate the global cost function. In particular, as opposed to the approaches built on distributed dual averaging, where the subgradient information are exchanged, here, each node updates its local primal variable according to the local mini-batch gradients, and then it shares the primal information within its local neighborhood.

4.2. Theoretical Results

We begin analyzing the performance of DABMD with a convergence result on the local decision variables.

We define $\Phi(k, s) = P^s P^{s+1} \dots P^k$ as the transition matrix that records the weights history from round s to k. The convergence properties of this matrix is extensively analyzed in (Nedic & Ozdaglar, 2009). In this work, we use the following result

$$|\Phi(k, s)_{ij} - \frac{1}{n}| \leq \gamma \Gamma^{k-s}, \tag{13}$$

where $\gamma = \frac{2(1+\eta^{-B_0})}{1-\eta^{B_0}}$ and $\Gamma = (1 - \eta^{B_0})^{1/B_0}$, and $B_0 = (n-1)B$ is a positive integer. We refer the reader to *Proposition* 1 in (Nedic & Ozdaglar, 2009) for further discussion and proof.

Now, we are ready to present an upperbound on the deviation of the local decisions from the exact average in the following lemma.

Lemma 1 *Let $x_{i,t}$ be the sequence of local decisions generated by DABMD, with all initial local decisions set to zero vectors. Suppose every node uses a σ-strongly convex regularizer $r(.)$ and a non-increasing step size sequence*

$\{\alpha_t\}$. *Then, the following bound holds on* $x_{i,t+1}$:

$$\|x_{i,t+1} - \bar{x}_{t+1}\| \leq \frac{\alpha_0 L}{\sigma}\Big(2 + \frac{n\gamma}{1-\Gamma}\Big),$$

where $\bar{x}_{t+1} = \frac{1}{n}\sum_{i=1}^{n} x_{i,t+1}$.

Lemma 1 is proved in App. C of the supplementary material.

We observe that the error bound above relates to the network graph through parameters γ and Γ. Smaller values of Γ and γ result in local decisions moving closer to their average. In other words, a shorter communication interval B and a larger minimum link weight η would help individual nodes to collect the information of all other computing nodes across the network at a faster pace. Thus, each computing node can approximate $\bar{x}_{t+1}$ with a higher accuracy.

Remark 1. Similar to Lemma 1, the convergence of local decisions are investigated in Lemma 1 of (Shahrampour & Jadbabaie, 2017). However, that analysis is based on the assumption that an oracle provides a common knowledge matrix to all computing nodes and requires the weight matrix to be fixed. Hence, their work cannot be extended to the case of a time-varying network, such as in mobile cloud computing. In contrast, our analysis obviates the common knowledge requirement and is applicable to a general time-varying network, where in addition to the weight matrix, even the connections can be time-varying.

The succeeding theorem provides an upper bound on the expected dynamic regret and it is followed by a corollary characterizing the regret bound for the optimized, fixed step size.

Theorem 2 *Let* $x_{i,t}$ *be the sequence of local decisions generated by DABMD. Let* $b_{\text{avg}} = \mathbb{E}[b_{i,t}]$, $b_{\min} = \min_{i,t}\{b_{i,t}\}$ *and* $b_{\max} = \max_{i,t}\{b_{i,t}\}$ *be the mean value, minimum, and maximum of the minibatch sizes across the nodes and over time. The expected dynamic regret satisfies*

$$\begin{aligned}\mathbb{E}\Big[\mathrm{Reg}_T^d\Big] &= \frac{1}{n}\sum_{i=1}^{n}\sum_{t=1}^{T}\mathbb{E}\Big[f_t(x_{i,t}) - f_t(x_t^*)\Big] \\ &\leq \sum_{t=1}^{T}\frac{2nb_{\text{avg}}L^2\alpha_0}{\sigma}\Big(2+\frac{n\gamma}{1-\Gamma}\Big) + \sum_{t=1}^{T}\frac{V_b L^2 n}{2\sigma b_{\min}}\alpha_t \\ &\quad + \sum_{t=1}^{T}\frac{Mnb_{\max}\mathbb{E}[\| x_{t+1}^* - x_t^* \|]}{\alpha_{T+1}} + \frac{2nb_{\max}R^2}{\alpha_{T+1}}, \quad (14)\end{aligned}$$

where $R^2 = \sup_{x,y\in\mathcal{X}} D_r(x,y)$, *and* $V_b = \mathbb{E}\left[b_{i,t}^2\right]$.

The proof of Theorem 2 is given in App. D.

Remark 2. The regret bound obtained above depends on the heterogeneity in processing speed only through parameter statistics $b_{\max}$, $b_{\min}$, b_{avg}, and V_b. In other words, these parameters capture and summarize the effect of varying processing speed among the computing nodes. Moreover, the impact of time-varying network graph with an imperfect averaging protocol is summarized in the first component. More specifically, it is through this term that the network connectivity interval B, minimum link weight η, and the precision of local estimates of the global minimizer appear in the result. Furthermore, the impact of the dynamic offline comparator variable x_t^* is reflected in the third term.

Corollary 3 *Under the same condition stated in Theorem 2, with fixed step size*

$$\alpha \in O\left(\sqrt{b_{\max}(\bar{A}_T+1)/\left[T\left(\frac{2n\gamma b_{\text{avg}}}{1-\Gamma}+\frac{V_b}{b_{\min}}\right)\right]}\right),$$

we have (15)

$$\mathbb{E}\left[\mathrm{Reg}_T^d\right] \leq O\left(\sqrt{T(\bar{A}_T+1)b_{\max}\left(\frac{2n\gamma b_{\text{avg}}}{1-\Gamma}+\frac{V_b}{b_{\min}}\right)}\right),$$

where $\bar{A}_T = \sum_{t=1}^{T}\mathbb{E}[\| x_{t+1}^* - x_t^* \|]$ *denotes the expected value of the total variation of the dynamic minimizer.*

This corollary directly follows from substituting the fixed step size α into the bound in (14).

Remark 3. We remark that our result recovers the regret rates previously derived for two special cases. When computing nodes are homogeneous, $b_{i,t}$ is no longer a random variable. Instead, the minibatch size is fixed across the network and over time. In this case, the terms involving the distribution characteristics disappear. Thus, we have a regret bound of $O(\sqrt{T(A_T+1)})$, which recovers the result of (Shahrampour & Jadbabaie, 2017), in which distributed online mirror descent was studied. In addition, if the offline minimizers are fixed, the problem is reduced to minimizing the static regret. In this case, the path length A_T is equal to zero and we obtain a regret bound of $O(\sqrt{T})$, which recovers the result of (Tsianos & Rabbat, 2016) on distributed online dual averaging.

Remark 4. The regret bound scales with the expected path length $\bar{A}_T$, which collects the mismatch errors between successive offline minimizers. When the minimizer sequence drifts slowly, the overall error is small and $\bar{A}_T$ can be of a constant order. On the other hand, severe fluctuation in the objective function or network topology can result in large mismatch errors, which can cause $\bar{A}_T$ to become linear in time. Such behavior is natural since even for centralized online optimization, the problem is generally intractable under worst-case system fluctuation (Zhang et al., 2017). Nevertheless, our goal is to consider $\bar{A}_T$ as a complexity measure of the problem environment and derive the regret bound in term of this quantity. We observe that if $\bar{A}_T$ is sublinear then the dynamic regret is also sublinear.

5. Multiple Consensus Iterations

In this section, we study a more general version of DABMD, which uses multiple consensus averaging iterations. Here we remark that the analysis of regret bound for this general case requires new techniques different from those used in the previous section. As a result, the obtained regret bound for k consensus averaging iterations is different from our previous result even for $k = 1$, therefore necessitating presentation in a separate section.

Similar to the previous case, this version also contains three steps. However, in the previous case, computing nodes perform only a single consensus iteration. Thus, in estimating the global minimizer each node only receives the message of the immediate neighbors that are one hop away, and is oblivious about the decision of the nodes that are farther away. This issue can be resolved via multiple consensus averaging iterations. In particular, with k iterations of consensus averaging, computing nodes receive the message of nodes that are k hops away. Therefore, they can more precisely approximate the global minimizer.

Let $y_{i,t}^{(0)} = x_{i,t}$ denote the initial message vector before beginning the consensus averaging step. Let $y_{i,t}^{(l)}$ denote the vector output at computing node i and round t, after l consensus iterations using matrix P^t. At each iteration l, after receiving messages from all its neighbors, node i updates its estimate of the global minimizer by

$$y_{i,t}^{(l)} = \sum_{j=1}^{n} P_{ij}^{t} y_{j,t}^{(l-1)} = \sum_{j=1}^{n} [P^t]_{ij}^{l} y_{j,t}^{(0)}. \tag{16}$$

As long as graph $\mathcal{G}_t$, corresponding to the weight matrix P^t, is connected and the second-largest eigen value of P^t is strictly less than unity, the consensus iterations are guaranteed to converge to the exact average. However, for a finite number of iterations each node will have an error in its estimation. The following lemma bounds the consensus errors after k iterations.

Lemma 4 *Let $y_{i,t}^{(k)}$ represent the output of the consensus averaging after k iterations with updates (16). Under the same condition stated in Lemma 1, the following bound holds:*

$$\left\| y_{i,t}^{(k)} - \bar{x}_t \right\| \leq \frac{2\sqrt{n}}{\exp\left[kn^{-3}\right]} \frac{\alpha_0 L}{\sigma} \left(2 + \frac{n\gamma^{(k)}}{1-\Gamma^{(k)}} \right),$$

where $\bar{x}_t = \frac{1}{n}\sum_i x_{i,t}$, $\gamma^{(k)} = 2(1+\eta_k^{-B_0})/(1-\eta_k^{B_0})$, $\Gamma^{(k)} = (1-\eta_k^{B_0})^{1/B_0}$, and $\eta_k = \min_{i,j}[P^t]_{i,j}^{k}$.

Lemma 4 is proved in App. E of the supplementary material.

The following theorem establishes an upper bound on the expected dynamic regret, followed by a corollary characterizing the regret rate for a fixed step size.

Theorem 5 *Let $x_{i,t}$ be the sequence of local decisions generated by DABMD with k consensus averaging iterations. The expected dynamic regret satisfies*

$$\begin{aligned}
&\mathbb{E}[\mathrm{Reg}_T^d] \leq \\
&\sum_{t=1}^{T} \frac{b_{\mathrm{avg}} L^2 \alpha_0 n}{\sigma} \left(2 + \frac{n\gamma^{(k)}}{1-\Gamma^{(k)}} \right) \left(1 + \frac{2\sqrt{n}}{\exp[kn^{-3}]} \right) \\
&+ \sum_{t=1}^{T} \frac{V_b L^2 n \alpha_t}{2\sigma b_{\min}} + \sum_{t=1}^{T} \frac{M n b_{\max} \mathbb{E}[\| x_{t+1}^* - x_t^* \|]}{\alpha_t} \\
&+ \frac{2nR^2 b_{\max}}{\alpha_{T+1}}.
\end{aligned} \tag{17}$$

The proof of Theorem 5 is given in App. F.

Corollary 6 *Under the same condition stated in Theorem 5, with fixed step size*

$$\alpha \in O\left(\sqrt{ \frac{b_{\max}(\bar{A}_T + 1)}{T\left(\frac{2n\sqrt{n}\gamma^{(k)} b_{\mathrm{avg}}}{(1-\Gamma^{(k)})\exp[kn^{-3}]} + \frac{V_b}{b_{\min}} \right)} } \right),$$

we have

$$\begin{aligned}
&\mathbb{E}\left[\mathrm{Reg}_T^d\right] \leq \qquad (18) \\
&O\left(\sqrt{ T(\bar{A}_T + 1) b_{\max} \left(\frac{2n\sqrt{n}\gamma^{(k)} b_{\mathrm{avg}}}{(1-\Gamma^{(k)})\exp[kn^{-3}]} + \frac{V_b}{b_{\min}} \right)} \right).
\end{aligned}$$

This corollary directly follows from substituting the fixed step size α into the bound in (17).

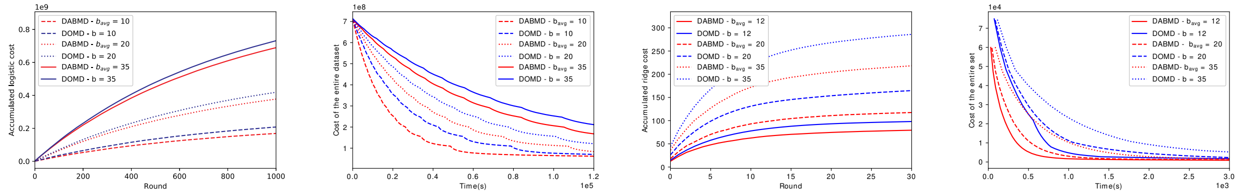
Remark 5. We remark that the regret bound of DABMD with multiple consensus iterations in (18) is of the same order of that of DABMD with a single consensus iteration in (15). The effect of multiple consensus iterations are mainly summarized in $\gamma^{(k)}$, $\Gamma^{(k)}$, and $\exp\left[kn^{-3}\right]$, where both $\gamma^{(k)}$ and $\Gamma^{(k)}$ are functions of the transition matrix, and $\exp\left[kn^{-3}\right]$ is resulted from bounding the second largest eigen value of a doubly stochastic matrix (Landau & Odlyzko, 1981).

6. Experiments

We investigate the performance of DABMD via numerical experiments on two different machine learning tasks using real-world datasets. We compare DABMD with the method proposed in (Shahrampour & Jadbabaie, 2017), which was termed Distributed Online Mirror Descent (DOMD). Due to page limitation, here we focus on the case of single consensus iteration. Additional experiments are presented in App. G.

6.1. Logistic Regression

In the first experiment, we consider multi-class classification with logistic regression. In this task, learner nodes observe a sequence of labeled examples (ω, z), where $\omega \in \mathbb{R}^d$, and



(a) Accumulated logistic cost of MNIST dataset.

(b) Logistic cost of MNIST dataset.

(c) Accumulated ridge cost of YouTube transcoding dataset.

(d) Ridge cost of YouTube transcoding dataset.

Figure 1. Performance comparison between DOMD and DABMD on MNIST and YouTube transcoding datasets.

the label z, denoting the class of the data example, is drawn from a discrete space $\mathcal{Z} = \{1, 2, \dots, c\}$. We use the well-known MNIST digits dataset, where every data sample ω is an image of size 28×28 pixel that can be represented by a 784-dimensional vector, i.e., $d = 784$. Each sample corresponds to one of the digits in $\{0, 1, \dots, 9\}$, and hence, there are $c = 10$ classes.

We consider a network consisting of $n = 10$ distributed nodes. The underlying network topology switches sequentially in a round robin manner among some doubly stochastic generated graphs. In addition, to model the processing speed of distributed nodes, similar to studies on stragglers in (Dutta et al., 2018) and (Lee et al., 2017), we adopt a shifted exponential distribution. This captures a minimum computing time, denoted by ϵ, to process a sample, while the remaining computing time is memoryless. More specifically, the computing time of each sample is modeled as a random variable, which follows the probability density function of the form $P(r) = \lambda \exp(-\lambda(r - \epsilon))$. We set $\lambda = 0.5$ and $\epsilon = 1$. For fair comparison, we extend DOMD to have a fixed minibatch size close to the value of b_{avg}, which is empirically found.

For logistic regression, the cost function associated with each data point is given by

$$f(x, (\omega_i, z_i)) = \log(1 + \exp(-z_i x^T \omega_i)).$$

The goal of the computing nodes is to classify streaming images online by tracking the unknown optimal parameter x_t^*.

In Fig. 1, we compare the performance of DABMD with DOMD, for different minibatch sizes b. From Fig. 1(a) and Fig. 1(b), we see that DABMD can save up to 10% in accumulated cost after 1000 rounds, and it reduces the convergence time up to 30% compared with DOMD.

6.2. Ridge Regression

In our next experiment, we consider the ridge regression problem. The cost function for each data sample is given by

$$f(x, (\omega_i, z_i)) = (x^T \omega_i - z_i)^2.$$

We use YouTube transcoding measurements from a publicly available dataset (Deneke et al., 2014). It contains 6×10^4 data samples, representing the input and output characteristics of a video transcoding application. The input data space has a dimension of $d = 20$, corresponding to video transcoding attributes such as resolution, bit rate, and required memory. Using this dataset, the goal is to predict the transcoding time based on the video features.

Similar to (Champati & Liang, 2017), we consider a Pareto distribution to model the processing time required to finish a transcoding task. By adjusting the Pareto tail index, we can control how unevenly the processing times are distributed, and thus, can capture the heterogeneity of the network in terms of processing speed. We set the Pareto tail index and scale parameter to 1.6 and 1.0, respectively. The network topology is time-varyig and is generated in the same way as the previous experiment.

From Figs. 1(c) and 1(d), we observe that the performance advantage of DABMD is more pronounced when the computing time has a heavy tail distribution. This is because with a heavy tail distribution, it is more likely to observe some relatively long computing time in DOMD, which causes a significant delay and leads to inefficiency. DOMD performs particularly poorly when b is large. In comparison, DABMD saves up to 30% in accumulated cost after 30 rounds, and reduces the convergence time up to 20%.

7. Conclusion

We have studied the problem of distributed online optimization over a heterogeneous time-varying network. The proposed DABMD algorithm imposes a fixed per-round processing time interval for all computing nodes. Thus, a key feature of DABMD is that the optimization progress does not solely depend on the slowest computing node. We discussed two versions of DABMD, depending on whether the computing nodes average their primal variables via single or multiple consensus iterations. We guarantee the performance of DABMD and provide upper bounds on the expected dynamic regret. Experimental results on MNIST and YouTube transcoding datasets show substantial improvement on both the accumulated cost and convergence speed compared with existing alternatives. We observe that the performance advantage of using a fixed round time is more significant when the sample processing time is taken from a heavy tail distribution.

References

Bauschke, H. H. and Borwein, J. M. Joint and separate convexity of the bregman distance. *Studies in Computational Mathematics*, 8:22–36, 2001.

Beck, A. and Teboulle, M. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, 2003.

Bottou, L. and Bousquet, O. The tradeoffs of large scale learning. In *Proc. of Advances in Neural Information Processing Systems*, pp. 161–168, 2008.

Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J., et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine learning*, 3(1):1–122, 2011.

Champati, J. P. and Liang, B. Single restart with time stamps for computational offloading in a semi-online setting. In *Proc. IEEE Conference on Computer Communications (INFOCOM)*, 2017.

Deneke, T., Haile, H., Lafond, S., and Lilius, J. Video transcoding time prediction for proactive load balancing. In *Proc. IEEE International Conference on Multimedia and Expo (ICME)*, 2014.

Duchi, J. C., Shalev-Shwartz, S., Singer, Y., and Tewari, A. Composite objective mirror descent. In *Proc. Conference on Learning Theory (COLT)*, 2010.

Duchi, J. C., Agarwal, A., and Wainwright, M. J. Dual averaging for distributed optimization: Convergence analysis and network scaling. *IEEE Transactions on Automatic control*, 57(3):592–606, 2011.

Dutta, S., Joshi, G., Ghosh, S., Dube, P., and Nagpurkar, P. Slow and stale gradients can win the race: Error-runtime trade-offs in distributed sgd. In *Proc. International Conference on Artificial Intelligence and Statistics*, 2018.

Ferdinand, N., Al-Lawati, H., Draper, S., and Nokleby, M. Anytime minibatch: Exploting stragglers in online distributed optimization. In *Proc. International Conference on Learning Representations (ICLR)*, 2019.

Hall, E. C. and Willett, R. M. Online convex optimization in dynamic environments. *IEEE Journal of Selected Topics in Signal Processing*, 9(4):647–662, 2015.

Hosseini, S., Chapman, A., and Mesbahi, M. Online distributed optimization via dual averaging. In *Proc. IEEE Conference on Decision and Control (CDC)*, pp. 1484–1489, 2013.

Landau, H. and Odlyzko, A. Bounds for eigenvalues of certain stochastic matrices. *Linear algebra and its Applications*, 38:5–15, 1981.

Lee, K., Lam, M., Pedarsani, R., Papailiopoulos, D., and Ramchandran, K. Speeding up distributed machine learning using codes. *IEEE Transactions on Information Theory*, 64(3):1514–1529, 2017.

Li, J., Chen, G., Dong, Z., and Wu, Z. Distributed mirror descent method for multi-agent optimization with delay. *Neurocomputing*, 177:643–650, February 2016.

Lin, M., Wierman, A., Andrew, L., and Thereska, E. Dynamic right-sizing for power-proportional data centers. *IEEE/ACM Transactions on Networking*, 21(5):1378–1391, 2013.

Mateos-Nunez, D. and Cortés, J. Distributed online convex optimization over jointly connected digraphs. *IEEE Transactions on Network Science and Engineering*, 1(1): 23–37, 2014.

Nedic, A. and Ozdaglar, A. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1):48, 2009.

Nedich, A., Lee, S., and Raginsky, M. Decentralized online optimization with global objectives and local communication. In *Proc. American Control Conference (ACC)*, 2015.

Nemirovsky, A. S. and Yudin, D. B. Problem complexity and method efficiency in optimization. *Wiley*, 1983.

Rabbat, M. Multi-agent mirror descent for decentralized stochastic optimization. In *Proc. IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, 2015.

Raginsky, M. and Bouvrie, J. Continuous-time stochastic mirror descent on a network: Variance reduction, consensus, convergence. In *Proc. IEEE Conference on Decision and Control (CDC)*, 2012.

Shahrampour, S. and Jadbabaie, A. Distributed online optimization in dynamic environments using mirror descent. *IEEE Transactions on Automatic Control*, 63(3):714–725, 2017.

Shalev-Shwartz, S. Online learning and online convex optimization. *Foundations and Trends in Machine Learning*, 4(2):107–194, 2012.

Shi, M., Lin, X., Fahmy, S., and Shin, D.-H. Competitive online convex optimization with switching costs and ramp constraints. In *Proc. IEEE Conference on Computer Communications (INFOCOM)*, 2018.

Tandon, R., Lei, Q., Dimakis, A. G., and Karampatziakis, N. Gradient coding: Avoiding stragglers in distributed learning. In *Proc. International Conference on Machine Learning (ICML)*, 2017.

Tsianos, K. I. and Rabbat, M. G. Distributed dual averaging for convex optimization under communication delays. In *Proc. American Control Conference (ACC)*, 2012.

Tsianos, K. I. and Rabbat, M. G. Efficient distributed online prediction and stochastic optimization with approximate distributed averaging. *IEEE Transactions on Signal and Information Processing over Networks*, 2(4):489–506, 2016.

Tsitsiklis, J., Bertsekas, D., and Athans, M. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE Transactions on Automatic Control*, 31(9):803–812, 1986.

Wang, H., Liao, X., Huang, T., and Li, C. Cooperative distributed optimization in multiagent networks with delays. *IEEE Transactions on Systems, Man, and Cybernetics*, 45(2):363–369, 2014.

Wang, Y., Zhou, H., and Hong, Y. Distributed stochastic mirror descent algorithm over time-varying network. In *Proc. IEEE Conference on Control and Automation (ICCA)*, 2018.

Xi, C., Wu, Q., and Khan, U. A. Distributed mirror descent over directed graphs. *arXiv preprint, arXiv:1412.5526*, 2014.

Xiao, L. Dual averaging methods for regularized stochastic learning and online optimization. *Journal of Machine Learning Research*, 11:2543–2596, 2010.

Yuan, D., Xu, S., Zhao, H., and Rong, L. Distributed dual averaging method for multi-agent optimization with quantized communication. *Systems & Control Letters*, 61(11): 1053–1061, 2012.

Yuan, D., Hong, Y., Ho, D. W., and Jiang, G. Optimal distributed stochastic mirror descent for strongly convex optimization. *Automatica*, 90:196–203, 2018.

Zhang, L., Yang, T., Yi, J., Jin, R., and Zhou, Z.-H. Improved dynamic regret for non-degenerate functions. In *Proc. Advances in Neural Information Processing Systems*, pp. 732–741, 2017.

Zinkevich, M. Online convex programming and generalized infinitesimal gradient ascent. In *Proc. International Conference on Machine Learning (ICML)*, 2003.

A. Additional Preliminary Definitions

Combining the definitions of strong convexity in (8) and Bergman divergence in (9) results in an important property of the Bregman divergence associated with a σ-strongly convex function, i.e.,

$$D_r(x, y) \geq \frac{\sigma}{2} \parallel x - y \parallel^2 . \tag{19}$$

We assume that the Bregman divergence satisfies the following conditions (Bauschke & Borwein, 2001).

(a) (*Separate Convexity*) Let x and y_k, $k = 1, \dots, K$, be arbitrary vectors in $\mathbb{R}^d$. Then, the following inequality holds:

$$D_r\Big(x, \sum_{k=1}^{K} \mu_k y_k\Big) \leq \sum_{k=1}^{K} \mu_k D_r(x, y_k), \tag{20}$$

where μ_k are non-negative constants with $\sum_k \mu_k = 1$.

(b) (*Lipschitz Continuity*) The Bregman divergence is Lipschitz continuous of the following form:

$$|D_r(x, z) - D_r(y, z)| \leq M \parallel x - y \parallel, \forall x, y, z \in \mathcal{X}, \tag{21}$$

where M is a positive constant.

Many commonly used functions satisfy (20), e.g., Euclidean distance and Kullback-Leibler divergence (refer to (Bauschke & Borwein, 2001) for further discussion and proof). Moreover, the condition in (21) is directly satisfied with a proper choice of Lipschitz continuous function $r(.)$ on the compact set $\mathcal{X}$.

B. Pseudocode of DABMD Algorithm

The psuedocode of DABMD operating over a heterogeneous network is given in Algorithm 1. Lines $3 - 9$ correspond to the any-batch computation step, in which every node i computes $b_{i,t}$ gradients during the fixed computation time interval, denoted by T_r. Line 10 corresponds to the update step, where nodes update their decisions using the mirror descent method. Line 11 represents the consensus averaging step, in which each node shares the local decision with its neighbors. Upon receiving the messages of neighboring nodes, every node i updates its estimate of the global minimizer $y_{i,t+1}$.

C. Proof of Lemma 1

In the proof of Lemma 1, we make use of another technical lemma. The following lemma presents an upper bound on the deviation of the local decisions from their approximate average.

Algorithm 1 DABMD algorithm

```
Input: initial points: $\{x_{i,0}, y_{i,0}\}$; step size $\alpha_t$; time horizon $T$.
Output: sequence of decisions $\{x_{i,t}, y_{i,t} : 1 \leq t \leq T\}$.
1: for $t = 1, 2, \dots, T$ do
2:    Initialize $b_{i,t} = 0, g_{i,t} = 0$
3:    $T_0 = current\_time$
4:    while $current\_time - T_0 \leq T_r$ do
5:       Receive input $\omega_{i,t}^s$ sampled i.i.d from $\Omega$
6:       Compute gradient: $g_{i,t} \leftarrow g_{i,t} + \nabla f(y_{i,t}, \omega_{i,t}^s)$
7:       $b_{i,t} \leftarrow b_{i,t} + 1$
8:    end while
9:    Normalize gradients: $g_{i,t} \leftarrow \frac{1}{b_{i,t}} g_{i,t}$
10:   Update local decisions:
        $x_{i,t+1} = \operatorname*{argmin}_{x \in \mathcal{X}} \Big\{ \langle x, g_{i,t} \rangle + \frac{1}{\alpha_t} D_r(x, y_{i,t}) \Big\}$
11:   Average decisions by consensus iteration
        single consensus iteration:
           $y_{i,t+1} \leftarrow \sum_{j=1}^{n} P_{ij}^{t+1} x_{j,t+1}$
        multiple consensus iteration:
           $y_{i,t+1} \leftarrow \sum_{j=1}^{n} [P^{t+1}]_{ij}^{k} x_{j,t+1}$
12: end for
```

Lemma 7 *If every node uses a σ-strongly convex regularization function $r(.)$, we have*

$$\parallel x_{i,t+1} - y_{i,t} \parallel \leq \frac{L\alpha_t}{\sigma}, \forall i \in \mathcal{V}, \forall t \geq 0.$$

Proof. By applying the first-order optimality condition to the update (11), we get

$$\langle x - x_{i,t+1}, \alpha_t g_{i,t} + \nabla r(x_{i,t+1}) - \nabla r(y_{i,t}) \rangle \geq 0, \forall x \in \mathcal{X}. \tag{22}$$

Then, setting $x = y_{i,t}$ in (22) yields

$$\begin{aligned} &\langle y_{i,t} - x_{i,t+1}, \alpha_t g_{i,t} \rangle \\ &\qquad \geq \langle x_{i,t+1} - y_{i,t}, \nabla r(x_{i,t+1}) - \nabla r(y_{i,t}) \rangle. \end{aligned} \tag{23}$$

Next, we exploit the strong convexity of regularizer $r(.)$, i.e.,

$$\begin{aligned} r(x_{i,t+1}) - r(y_{i,t}) - \nabla r(y_{i,t})^T (x_{i,t+1} - y_{i,t}) \\ \geq \frac{\sigma}{2} \parallel x_{i,t+1} - y_{i,t} \parallel^2 . \end{aligned}$$

Taking gradient with respect to $x_{i,t+1}$ yields

$$\nabla r(x_{i,t+1}) - \nabla r(y_{i,t}) \geq \sigma \parallel x_{i,t+1} - y_{i,t} \parallel .$$

Combining the above with (23), and using the Lipschitzness property, we obtain

$$\alpha_t L \parallel x_{i,t+1} - y_{i,t} \parallel \geq \sigma \parallel x_{i,t+1} - y_{i,t} \parallel^2 . \tag{24}$$

Dividing (24) by $\sigma \parallel x_{i,t+1} - y_{i,t} \parallel$ completes the proof. □

Now, we are ready to present an upperbound on the deviation of the local decisions from the exact average. So, we present the proof of Lemma 1.

Let $e_{i,t} = x_{i,t+1} - y_{i,t}$ denote the error between the local decision and local estimate between two consecutive time slots. Then, the update (12) can be recursively written as

$$\begin{aligned} x_{i,t+1} = y_{i,t} + e_{i,t} &= \sum_{j=1}^{n} P_{ij}^{t} x_{j}^{t} + e_{i,t} \\ &= \sum_{j=1}^{n} \Phi(t,0)_{ij} x_{j,0} + \sum_{s=1}^{t} \sum_{j=1}^{n} \Phi(t,s)_{ij} e_{j,s-1} + e_{i,t}. \end{aligned} \tag{25}$$

Averaging the above over the entire network yields

$$\begin{aligned} \bar{x}_{t+1} = \frac{1}{n} \sum_{i=1}^{n} x_{i,t+1} &= \frac{1}{n} \sum_{j=1}^{n} \sum_{i=1}^{n} \Phi(t,0)_{ij} x_{j,0} \\ &+ \frac{1}{n} \sum_{s=1}^{t} \sum_{j=1}^{n} \sum_{i=1}^{n} \Phi(t,s)_{ij} e_{j,s-1} + \frac{1}{n} \sum_{i=1}^{n} e_{i,t} \\ &= \frac{1}{n} \sum_{j=1}^{n} x_{j,0} + \frac{1}{n} \sum_{s=1}^{t+1} \sum_{j=1}^{n} e_{j,s-1}, \end{aligned} \tag{26}$$

where in the last line, we have used the fact that $\Phi(t,0)$ and $\Phi(t,s)$ are doubly stochastic matrices, and thus $\sum_i \Phi(t,0) = \sum_i \Phi(t,s)_{ij} = 1$. By combining (25) and (26), we obtain

$$\begin{aligned} \parallel x_{i,t+1} - \bar{x}_{t+1} \parallel \leq & \sum_{j=1}^{n} \parallel \Phi(t,0)_{ij} - \frac{1}{n} \parallel \parallel x_{j,0} \parallel \\ &+ \sum_{s=1}^{t} \sum_{j=1}^{n} \parallel \Phi(t,s)_{ij} - \frac{1}{n} \parallel \parallel e_{j,s-1} \parallel \\ &+ \frac{1}{n} \sum_{j=1}^{n} \parallel e_{j,t} \parallel + \parallel e_{i,t} \parallel . \end{aligned}$$

We next use the result of Lemma 7 and (13) to bound the error and transition matrix errors, respectively. Noting that all initial local decisions are zero vectors, we have

$$\begin{aligned} \parallel x_{i,t+1} - & \bar{x}_{t+1} \parallel \\ &\leq \sum_{s=1}^{t} \sum_{j=1}^{n} \gamma \Gamma^{t-s} \frac{L\alpha_{s-1}}{\sigma} + \frac{1}{n} \sum_{j=1}^{n} \frac{\alpha_t L}{\sigma} + \frac{\alpha_t L}{\sigma} \\ &\leq \sum_{s=1}^{t} \frac{\alpha_0 L n}{\sigma} \gamma \Gamma^{t-s} + \frac{2\alpha_t L}{\sigma} \\ &\leq \frac{\alpha_0 L}{\sigma} \Big(2 + \frac{n\gamma}{1-\Gamma} \Big). \end{aligned} \tag{27}$$

where α_0 denotes the initial step size. To obtain the right-hand side above we used the fact that $\{\alpha_t\}$ is a non-increasing sequence. □

D. Proof of Theorem 2

D.1. Key Lemmas

The following two lemmas pave the way for our regret analysis provided in Theorem 2. Lemma 8 shows the impact of the dynamic minimizers on the regret bound.

Lemma 8 *For any non-increasing step size sequence $\{\alpha_t\}$ it holds that*

$$\begin{aligned} \sum_{i=1}^{n} \sum_{t=1}^{T} \Big(\frac{1}{\alpha_t} D_r(x_t^*, y_{i,t}) - \frac{1}{\alpha_t} D_r(x_t^*, x_{i,t+1}) \Big) \\ \leq \frac{2nR^2}{\alpha_{T+1}} + \sum_{t=1}^{T} \frac{Mn \parallel x_{t+1}^* - x_t^* \parallel}{\alpha_{t+1}}, \end{aligned} \tag{28}$$

where $R^2 = \sup_{x,y \in \mathcal{X}} D_r(x,y)$.

Proof. We begin by adding and subtracting several terms as follows:

$$\begin{aligned} \frac{1}{\alpha_t} D_r(x_t^*, y_{i,t}) - \frac{1}{\alpha_t} D_r(x_t^*, x_{i,t+1}) = \\ + \frac{1}{\alpha_t} D_r(x_t^*, y_{i,t}) - \frac{1}{\alpha_{t+1}} D_r(x_{t+1}^*, y_{i,t+1}) \\ + \frac{1}{\alpha_{t+1}} D_r(x_{t+1}^*, y_{i,t+1}) - \frac{1}{\alpha_{t+1}} D_r(x_t^*, y_{i,t+1}) \\ + \frac{1}{\alpha_{t+1}} D_r(x_t^*, y_{i,t+1}) - \frac{1}{\alpha_{t+1}} D_r(x_t^*, x_{i,t+1}) \\ + \frac{1}{\alpha_{t+1}} D_r(x_t^*, x_{i,t+1}) - \frac{1}{\alpha_t} D_r(x_t^*, x_{i,t+1}). \end{aligned} \tag{29}$$

We proceed by bounding every pair of terms on the right-hand side of (29). The first pair telescopes when summed over time t. For the second pair, by the Lipschitz condition on the Bregman divergence (21), we have

$$\begin{aligned} \frac{1}{\alpha_{t+1}} D_r(x_{t+1}^*, y_{i,t+1}) - \frac{1}{\alpha_{t+1}} D_r(x_t^*, y_{i,t+1}) \\ \leq \frac{M \parallel x_{t+1}^* - x_t^* \parallel}{\alpha_{t+1}}. \end{aligned} \tag{30}$$

Furthermore, using the separate convexity of the Bregman divergence given in (20), the third pair is bounded as follows:

$$\begin{aligned}
&\sum_{i=1}^{n}\Big(D_r(x_t^*, y_{i,t+1}) - D_r(x_t^*, x_{i,t+1})\Big)\\
&= \sum_{i=1}^{n}\left(D_r(x_t^*, \sum_{j=1}^{n} P_{ij}^t x_{j,t+1}) - D_r(x_t^*, x_{i,t+1})\right)\\
&\leq \sum_{j=1}^{n}\left(\sum_{i=1}^{n} P_{ij}^t\right) D_r(x_t^*, x_{j,t+1})\\
&\quad - \sum_{i=1}^{n} D_r(x_t^*, x_{i,t+1}) = 0, \qquad (31)
\end{aligned}$$

where the separate convexity of Bregman divergence and the doubly stochastic property of P^t are used in the last line. Summing (29) over time and nodes, we obtain

$$\begin{aligned}
&\sum_{i=1}^{n}\sum_{t=1}^{T}\left(\frac{1}{\alpha_t} D_r(x_t^*, y_{i,t}) - \frac{1}{\alpha_t} D_r(x_t^*, x_{i,t+1})\right)\\
&\quad\leq \frac{nR^2}{\alpha_1} + \sum_{t=1}^{T} \frac{Mn \parallel x_{t+1}^* - x_t^* \parallel}{\alpha_{t+1}}\\
&\qquad + nR^2 \sum_{t=1}^{T}\left(\frac{1}{\alpha_{t+1}} - \frac{1}{\alpha_t}\right)\\
&\quad\leq \frac{2nR^2}{\alpha_{T+1}} + \sum_{t=1}^{T} \frac{Mn \parallel x_{t+1}^* - x_t^* \parallel}{\alpha_{t+1}}, \qquad (32)
\end{aligned}$$

where the fact that $\{\alpha_t\}$ is a positive and non-increasing sequence is used in the last line. □

Lemma 9 *Let* $b_{\min} = \min_{i,t}\{b_{i,t}\}$ *and* $b_{\max} = \max_{i,t}\{b_{i,t}\}$ *be the minimum and maximum of the minibatch sizes across nodes and over time. The sequence* $y_{i,t}$ *generated by* (12) *satisfies*

$$\begin{aligned}
&\sum_{i=1}^{n}\sum_{t=1}^{T}\mathbb{E}\Big[f_{i,t}(y_{i,t}) - f_{i,t}(x_t^*)\Big]\\
&\leq \sum_{t=1}^{T}\frac{V_b L^2 n}{2\sigma b_{\min}}\alpha_t + \sum_{t=1}^{T}\frac{Mnb_{\max}\mathbb{E}\left[\parallel x_{t+1}^* - x_t^* \parallel\right]}{\alpha_{T+1}}\\
&\quad + \frac{2nb_{\max}R^2}{\alpha^{T+1}},
\end{aligned}$$

where $V_b = \mathbb{E}\left[b_{i,t}^2\right]$ *and the expectation is taken with respect to the variability in the minibatch sizes.*

Proof. We start by adding and subtracting $f_{i,t}(y_{i,t})$ as follows:

$$\begin{aligned}
f_{i,t}(y_{i,t}) - f_{i,t}(x_t^*) &= \sum_{s=1}^{b_{i,t}}\left(f(y_{i,t}, \omega_{i,t}^s) - f(x_t^*, \omega_{i,t}^s)\right)\\
&\leq b_{i,t}\langle g_{i,t}, y_{i,t} - x_t^*\rangle\\
&= b_{i,t}\langle g_{i,t}, y_{i,t} - x_{i,t+1} + x_{i,t+1} - x_t^*\rangle\\
&\leq b_{i,t}L \parallel y_{i,t} - x_{i,t+1} \parallel + b_{i,t}\langle g_{i,t}, x_{i,t+1} - x_t^*\rangle, \qquad (33)
\end{aligned}$$

where we have used the convexity and Lipschitz continuity of $f_{i,t}(.)$ in the last line of (33). Next, we bound the last term of (33) as follows:

$$\begin{aligned}
&b_{i,t}\langle g_{i,t}, x_{i,t+1} - x_t^*\rangle\\
&\quad\leq \frac{b_{i,t}}{\alpha_t}\Big[D_r(x_t^*, y_{i,t}) - D_r(x_t^*, x_{i,t}) - D_r(x_{i,t+1}, y_{i,t})\Big]\\
&\quad\leq \frac{b_{i,t}}{\alpha_t}\Big[D_r(x_t^*, y_{i,t}) - D_r(x_t^*, x_{i,t})\Big]\\
&\qquad - \frac{\sigma b_{\min}}{2\alpha_t} \parallel y_{i,t} - x_{i,t+1} \parallel^2, \qquad (34)
\end{aligned}$$

where we have used a simple algebra of Bregman divergences (see (Beck & Teboulle, 2003)) to derive the first inequality in (34). Also, the last line of (34) is obtained by the strong convexity of $r(.)$, as presented in (19).

Finally, by substituting (34) into (33), we get

$$\begin{aligned}
&f_{i,t}(y_{i,t}) - f_{i,t}(x_t^*)\\
&\quad\leq b_{i,t}L \parallel y_{i,t} - x_{i,t+1} \parallel - \frac{\sigma b_{\min}}{2\alpha_t} \parallel y_{i,t} - x_{i,t+1} \parallel^2\\
&\qquad + \frac{b_{i,t}}{\alpha_t}\Big[D_r(x_t^*, y_{i,t}) - D_r(x_t^*, x_{i,t})\Big]\\
&\quad\leq \frac{b_{i,t}^2 L^2 \alpha_t}{2\sigma b_{\min}} + \frac{b_{\max}}{\alpha_t}\Big[D_r(x_t^*, y_{i,t}) - D_r(x_t^*, x_{i,t})\Big], \qquad (35)
\end{aligned}$$

where in the last line above we have used the fact that $cu - q\frac{u^2}{2} \leq \frac{c^2}{2q}$, with $c = b_{i,t}L$ and $q = \sigma b_{\min}/\alpha_t$. We sum (35) across computing nodes and over time and take expectation, and apply Lemma 8 to the last term to achieve the result. □

D.2. Proof of the Theorem

Now, we are ready to present the proof of Theorem 2.

To bound the dynamic regret, we begin by adding and subtracting $f_t(\bar{x}_t)$ as follows:

$$\begin{aligned} & f_t(x_{i,t}) - f_t(x_t^*) \\ &\quad = f_t(x_{i,t}) - f_t(\bar{x}_t) + f_t(\bar{x}_t) - f_t(x_t^*) \\ &\quad = \sum_{j=1}^{n} \Big(f_{j,t}(x_{i,t}) - f_{j,t}(\bar{x}_t) \Big) \\ &\qquad + \sum_{i=1}^{n} \Big(f_{i,t}(\bar{x}_t) - f_{i,t}(y_{i,t}) + f_{i,t}(y_{i,t}) - f_{i,t}(x_t^*) \Big) \\ &\quad \leq L \sum_{j=1}^{n} b_{j,t} \parallel x_{i,t} - \bar{x}_t \parallel + L \sum_{i=1}^{n} b_{i,t} \parallel y_{i,t} - \bar{x}_t \parallel \\ &\qquad + \sum_{i=1}^{n} \Big(f_{i,t}(y_{i,t}) - f_{i,t}(x_t^*) \Big), \end{aligned} \tag{36}$$

where we have used the Lipschitz continuity of $f_{i,t}(.)$ to derive the last inequality in (36). We next need to bound the three terms in the last line of (36). The first term can be bounded using the result of Lemma 1. The second term is bounded as follows:

$$\begin{aligned} L \sum_{i=1}^{n} b_{i,t} \parallel y_{i,t} - \bar{x}_t \parallel &= L \sum_{i=1}^{n} b_{i,t} \parallel \sum_{j=1}^{n} P_{ij}^t x_{j,t} - \bar{x}_t \parallel \\ &\leq L \sum_{i=1}^{n} b_{i,t} \sum_{j=1}^{n} P_{ij}^t \parallel x_{j,t} - \bar{x}_t \parallel \\ &\leq \frac{L^2}{\sigma} \alpha_0 \Big(2 + \frac{n\gamma}{1-\Gamma} \Big) \sum_{i=1}^{n} b_{i,t}, \end{aligned} \tag{37}$$

where we have used the result of Lemma 1 and doubly stochastic property of matrix P^t to obtain the last line. Substituting (37) into (36), taking expectation and summing over time, and combining that with the previous result in Lemma 9 completes the proof. □

E. Proof of Lemma 4

We use the following result on consensus averaging, which is presented in Lemma 1 of (Tsianos & Rabbat, 2016).

Let $\delta > 0$ be a given scalar and $\lambda_2(P^t)$ denote the second-largest eigen value of the doubly stochastic matrix P^t. If the number of consensus iterations satisfies

$$k \geq \frac{\log(\frac{1}{\delta} 2\sqrt{n} \max_j \parallel y_{j,t}^{(0)} - \bar{y}_t \parallel)}{1 - \lambda_2(P^t)}, \tag{38}$$

the following bound holds on the output after k consensus iterations:

$$\parallel y_{i,t}^{(k)} - \bar{y}_t \parallel \leq \delta. \tag{39}$$

Recall that the initial message $y_{i,t}^{(0)}$ is set to $x_{i,t}$, and thus the average $\bar{y}_t = \frac{1}{n}\sum_{i=1}^{n} y_{i,t}^{(0)}$ is equal to $\bar{x}_t$. Therefore, we can replace $\parallel y_{j,t}^{(0)} - \bar{y}_t \parallel$ by $\parallel x_{i,t} - \bar{x}_t \parallel$, and rewrite (38) as follows:

$$\delta \geq \frac{2\sqrt{n} \max_j \parallel y_{j,t}^{(0)} - \bar{y}_t \parallel}{\exp\Big[k \Big(1 - \lambda_2(P^t) \Big) \Big]}. \tag{40}$$

Similar to Lemma 1, using the updates (16), the following bound can be established on the consensus error

$$\parallel x_{i,t} - \bar{x}_t \parallel \leq \frac{\alpha_0 L}{\sigma} \Big(2 + \frac{n\gamma^{(k)}}{1 - \Gamma^{(k)}} \Big). \tag{41}$$

where $\gamma^{(k)}$ and $\Gamma^{(k)}$ are corresponding parameters of weight matrix $[P^t]^k$.

In addition, if D is an $n \times n$ doubly stochastic matrix, the second-largest eigen value is upper bounded by $\lambda_2(D) \leq 1 - n^{-3}$ (Landau & Odlyzko, 1981). Therefore, for every weight matrix P^t, we can conclude that

$$1 - \lambda_2(P^t) \leq n^{-3}. \tag{42}$$

Finally, combining (40), (41), and (42), we can set $\delta = \frac{2\sqrt{n}\alpha_0 L}{\exp[kn^{-3}]\sigma} (2 + \frac{n\gamma^{(k)}}{1-\Gamma^{(k)}})$. This value of δ satisfies (38), and hence, we get

$$\parallel y_{i,t}^{(k)} - \bar{x}_t \parallel \leq \frac{2\sqrt{n}\alpha_0 L}{\exp[kn^{-3}]\sigma} \Big(2 + \frac{n\gamma^{(k)}}{1 - \Gamma^{(k)}} \Big). \tag{43}$$

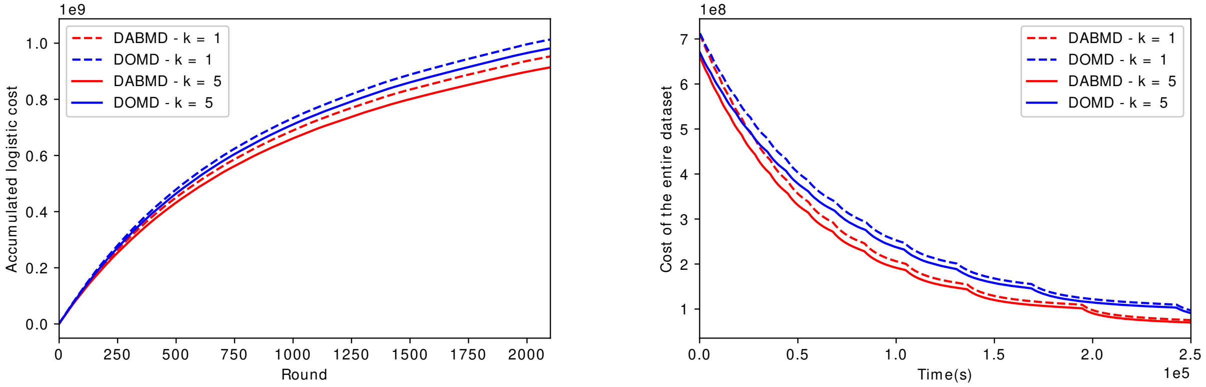
□

F. Proof of Theorem 5

To bound the dynamic regret, we begin by adding and subtracting $f_t(\bar{x}_t)$ as follows:

$$\begin{aligned} & f_t(x_{i,t}) - f_t(x_t^*) \\ &\quad = f_t(x_{i,t}) - f_t(\bar{x}_t) + f_t(\bar{x}_t) - f_t(x_t^*) \\ &\quad = \sum_{j=1}^{n} \Big(f_{j,t}(x_{i,t}) - f_{j,t}(\bar{x}_t) \Big) \\ &\qquad + \sum_{i=1}^{n} \Big(f_{i,t}(\bar{x}_t) - f_{i,t}(y_{i,t}) + f_{i,t}(y_{i,t}) - f_{i,t}(x_t^*) \Big) \\ &\quad \leq L \sum_{j=1}^{n} b_{j,t} \parallel x_{i,t} - \bar{x}_t \parallel + L \sum_{i=1}^{n} b_{i,t} \parallel y_{i,t} - \bar{x}_t \parallel \\ &\qquad + \sum_{i=1}^{n} \Big(f_{i,t}(y_{i,t}) - f_{i,t}(x_t^*) \Big), \end{aligned} \tag{44}$$

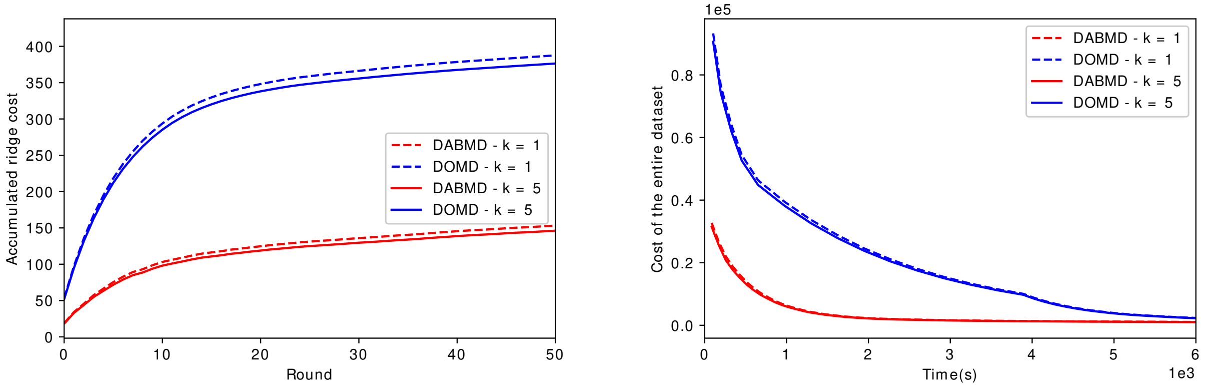
where we have used the Lipschitz continuity of $f_{i,t}(.)$. The first term on the right-hand side of (44) can be bounded by (41). The second term can be bounded using the result of



(a) Accumulated logistic cost of MNIST dataset.

(b) Logistic cost of MNIST dataset.

Figure 2. Performance comparison between DOMD and DABMD on MNIST dataset.



(a) Accumulated ridge cost of YouTube transcoding dataset.

(b) Ridge cost of YouTube transcoding dataset.

Figure 3. Performance comparison between DOMD and DABMD on YouTube transcoding dataset.

Lemma 4. By substituting (41) into (44), taking expectation and summing over time, and combining that with the previous result in Lemma 9, we obtain

$$\begin{aligned}
&\mathbb{E}[Reg_T^d] \leq \\
&\quad \sum_{t=1}^{T} \frac{b_{\text{avg}} L^2 \alpha_0 n}{\sigma} \left(2 + \frac{n\gamma^{(k)}}{1-\Gamma^{(k)}}\right)\left(1 + \frac{2\sqrt{n}}{\exp[kn^{-3}]}\right) \\
&\quad + \sum_{t=1}^{T} \frac{V_b L^2 \alpha_t}{2\sigma b_{\min}} + \sum_{t=1}^{T} \frac{M n b_{\text{avg}} \mathbb{E}[\| \, x_{t+1}^* - x_t^* \, \|]}{\alpha_t} \\
&\quad + \frac{2nR^2 b_{\text{avg}}}{\alpha_{T+1}}. \qquad (45)
\end{aligned}$$

□

G. Additional Experiments

In this section, we present additional experiments regarding the performance of DABMD. Here we investigate the performance of DABMD with multiple consensus iterations using MNIST and YouTube transcoding datasets (Deneke et al., 2014). All system and experiment parameter settings are the same as those presented in Section 6, unless otherwise specified. In particular, we consider a time-varying network, similar to the one used in Yuan et. al. in ICLR'20, shown in Fig. 4. Note that the union graph of any two consecutive rounds is strongly connected.

G.1. Logistic Regression

We consider the MNIST logistic regression problem, where the computing nodes perform k iterations of consensus averaging.

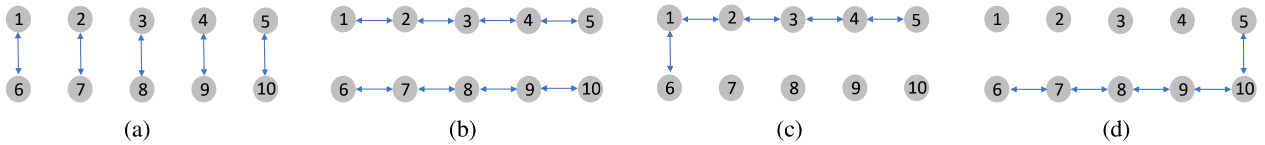


Figure 4. The network switches sequentially in a round robin manner among (a), (b), (c), and (d).

Fig. 2 illustrates the performance of DABMD versus DOMD on the MNIST logistic regression problem. We compare the results for $k = 1$ and $k = 5$ consensus averaging iterations. We observe that DABMD with $k = 5$ consensus averaging iterations incures up to 5% lower accumulated cost compared to the single consensus case. In addition, in both cases DABMD outperforms DOMD.

G.2. Ridge Regression

W also study the ridge regression problem on YouTube transcoding dataset under multiple consensus averaging settings.

We compare the performance of DABMD with DOMD for varying number of consensus iterations k in Fig. 3. We observe that the performance of DABMD improves with multiple consensus iterations, since computing nodes can more accurately approximate the global minimizer.