Distributed Optimization and Learning with Automated Stepsizes

Ziqin Chen and Yongqiang Wang, *Senior Member, IEEE*

Abstract— **The selection of stepsizes has always been an elusive task in distributed optimization and learning. Although some stepsize-automation approaches have been proposed in centralized optimization, these approaches are inapplicable in the distributed setting. This is because in distributed optimization/learning, letting individual agents adapt their own stepsizes unavoidably results in stepsize heterogeneity, which can easily lead to algorithmic divergence. To solve this issue, we propose an approach that enables agents to adapt their individual stepsizes without any manual adjustments or global knowledge of the objective function. To the best of our knowledge, this is the first algorithm to successfully automate stepsize selection in distributed optimization/learning. Its performance is validated using several machine learning applications, including logistic regression, matrix factorization, and image classification.**

I. INTRODUCTION

We consider a group of m agents communicating through a network to cooperatively solve the following optimization problem:

$$\min_{x\in\mathbb{R}^n} f(x) = \frac{1}{m}\sum_{i=1}^{m} f_i(x), \tag{1}$$

where $f_i(x):\mathbb{R}^n\to\mathbb{R}, i\in\{1,\cdots,m\}$ is a local objective function known to agent i only.

Nowadays, substantial progress has been made in solving the above distributed optimization/learning problem [1]–[8]. In all existing results, the stepsize is a crucial parameter that determines the performance of convergence. However, determining a good stepsize usually incurs tedious manual adjustments [9]–[11]. The distributed setting further complicates stepsize selection because in this case, each agent independently adapts its stepsize using its partial view of the objective function, which leads to stepsize heterogeneity that can easily lead to algorithmic divergence. Our experimental results in Fig. 1 show that the divergence issue becomes even more acute when the data distribution is heterogeneous across the agents. Moreover, existing distributed optimization and learning algorithms typically choose a constant or diminishing stepsize, whose selection requires knowledge of the communication graph or the Lipschitz constant of the global objective function, which is generally hard to obtain in practical distributed applications.

Recently, some stepsize-adaptation approaches have been proposed in distributed optimization and learning. For example, [12] introduced a backtracking line-search method to adapt stepsizes in distributed optimization. However, it requires execution of another subroutine with additional evaluation of gradients, and hence incurs heavy computational overheads. [13] used the Barzilai-Borwein stepsize in a gradient-tracking-based algorithm, which, unfortunately, requires knowledge of the global Lipschitz constant and the strongly convex coefficient. Several adaptive gradient methods have also been proposed to automate stepsize selection in distributed learning [14]–[16]. Nevertheless, these methods still require laborious tuning for learning-rate parameters. To the best of our knowledge, we still lack a solution which is capable of automating stepsize selection in distributed optimization and learning without any manual adjustments.

In this paper, we propose an approach that can automate stepsize selection in distributed optimization and learning without any manual adjustments or global knowledge of the objective function. Our basic idea is inspired by a recently proposed stepsize-automation approach for centralized optimization [17]. It is worth noting that directly applying the centralized stepsize-automation method in [17] to the distributed setting will unavoidably make the stepsize heterogeneous across the agents, which can easily lead to divergence. Hence, we propose to adjust the centralized stepsize automation approach in [17] and then combine the adjusted version with gradient tracking to automate stepsize selection in distributed optimization and learning. To the best of our knowledge, this is the first algorithm that successfully automates stepsize selection in distributed optimization and learning without any manual adjustments. Our numerical experiments with logistic regression, matrix factorization, and image classification confirm the effectiveness of the proposed approach in real-world machine learning problems. In fact, the proposed algorithm was shown to have better learning and test accuracies than existing popular algorithms for distributed optimization and learning.

Notation: We use $\mathbb{R}$, $\mathbb{N}$, and $\mathbb{N}^+$ to represent the sets of real numbers, nonnegative integers, and positive integers, respectively. We denote an m-dimensional column vectors whose elements are 0 and 1 as $\mathbf{0}_m$ and $\mathbf{1}_m$, respectively. We write an n-dimensional vector as $x\in\mathbb{R}^n$, with $\|x\|$ denoting its Euclidean norm. For vectors $x_1,\cdots,x_m$, we denote their stacked column vector by $\boldsymbol{x}=\mathrm{col}\{x_1,\cdots,x_m\}$. We add an overbar to a variable to denote the averaged version of all agents, e.g., $\bar{x}=\frac{1}{m}\sum_{i=1}^{m} x_i$.

II. Problem Statement

We assume that a network of m agents interact on an undirected graph $\mathcal{G}([m],\mathcal{E})$, where $[m]$ is the set of agents and $\mathcal{E}$ is the set of edges. The interaction strength is described by a weight matrix $W=\{w_{ij}\}\in\mathbb{R}^{m\times m}$, in which $w_{ij}>0$ if the edge $(i,j)\in\mathcal{E}$ exists, and $w_{ij}=0$ otherwise. The

The work was supported in part by the National Science Foundation under Grants ECCS-1912702, CCF-2106293, CCF-2215088, CNS-2219487, and CCF-2334449.

Ziqin Chen and Yongqiang Wang are with the Department of Electrical and Computer Engineering, Clemson University, Clemson, SC 29634 USA (e-mail: yongqiw@clemson.edu).

neighbor set $\mathcal{N}_i$ of agent i is defined as the set of agents $\{j|w_{ij} > 0\}$, which always includes itself. We make the following standard assumption on the interaction:

Assumption 1: $W = w_{ij} \in \mathbb{R}^{m\times m}$ satisfies $\mathbf{1}_m^T W = \mathbf{1}_m^T$, $W\mathbf{1}_m = \mathbf{1}_m$, and $\rho \triangleq \max\{|\lambda_2|, |\lambda_m|\} < 1$, where $\lambda_m \leq \lambda_{m-1} \leq \cdots < \lambda_1 = 1$ denote the eigenvalues of W.

In most existing distributed optimization and learning algorithms [1]–[6], [18]–[23], the stepsize is selected as a constant value (usually denoted as η) or a decaying sequence like $\frac{1}{t^v}$, where t is the iteration index and v is some positive constant. However, the selection of such η and v requires laborious manual adjustments to achieve effective convergence. Generally, to ensure provable convergence, the stepsize has to be below a threshold value which is determined by the interaction graph or the Lipschitz constant of the global objective function, which, in general, is difficult to obtain in the distributed setting. To make things worse, even if a good stepsize is obtained after tedious adjustments, it is usually highly dependent on the network size, topology, and datasets, making it hard to migrate to other applications or even datasets.

III. Distributed Stepsize-Automation Algorithm

To avoid tedious and repetitive stepsize tuning, we propose a stepsize-automation approach for distributed optimization and learning, as summarized in Algorithm 1.

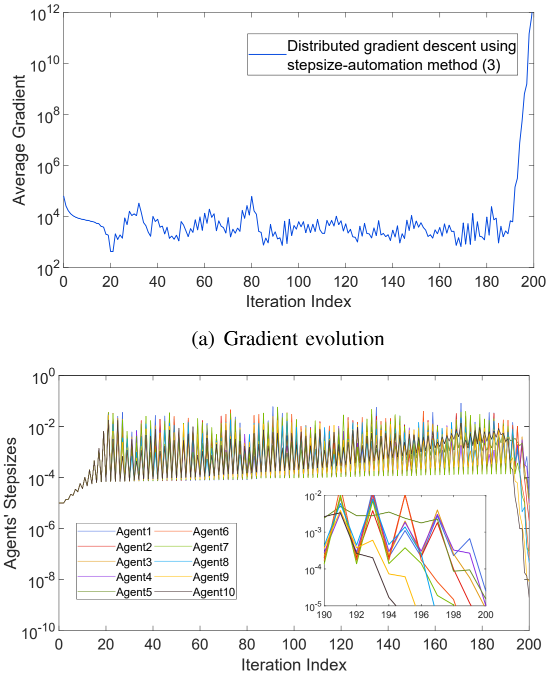
Algorithm 1 Stepsize-Automated Distributed Optimization and Learning (from agent i's perspective)

```
**Input:** Random initialization $x_{i,0} \in \mathbb{R}^n$ and $\eta_{i,0} > 0$; $y_{i,0} = \nabla f_i(x_{i,0})$.
**for** $t = 0, \cdots, T-1$ **do**
    $x_{i,\frac{t}{2}} = x_{i,t} - \eta_{i,t} y_{i,t}$
    $x_{i,t+1} = \sum_{j=1}^m w_{ij} x_{j,\frac{t}{2}}$
    $y_{i,\frac{t}{2}} = y_{i,t} + \nabla f_i(x_{i,t+1}) - \nabla f_i(x_{i,t})$
    $y_{i,t+1} = \sum_{j=1}^m w_{ij} y_{j,\frac{t}{2}}$
    $\eta_{i,t+1} = \min\left\{\sqrt{2}\eta_{i,t}, \frac{1}{\|y_{i,t+1}\|}, \frac{\|x_{i,t+1} - x_{i,t}\|}{2\|\nabla f_i(x_{i,t+1}) - \nabla f_i(x_{i,t})\|}\right\}$
**end for**
```

In Algorithm 1, the stepsize for each agent $i \in [m]$ is updated locally according to the following rule:

$$\eta_{i,t+1} = \min\left\{\sqrt{2}\eta_{i,t}, \frac{1}{\|y_{i,t+1}\|}, \frac{\|x_{i,t+1} - x_{i,t}\|}{2\|\nabla f_i(x_{i,t+1}) - \nabla f_i(x_{i,t})\|}\right\}. \tag{2}$$

In (2), the first argument of the min function is used to guarantee that the stepsize does not increase too fast. The second argument is used to estimate the inverse of the average gradient of all agents, and the third argument is used to approximate the inverse of the Lipschitz constant of the local gradient. It can be seen that the stepsize strategy in (2) only uses local gradients and variables, and is tuning-free. This approach differs from existing distributed adaptive stepsize approaches in [13]–[16], which require knowledge of the global Lipschitz constant. Moreover, unlike distributed optimization algorithms [18]–[22] that restrict the stepsize to be below the reciprocal of the global Lipschitz constant, the stepsize strategy in (2) is not subject to this limitation, thereby offering potentially fast convergence, as evidenced in our experimental results in Fig. 2-Fig. 4.



(a) Gradient evolution

(b) Stepsize evolution of all agents

Fig. 1. Matrix factorization using distributed gradient descent [1] with 10 agents. Each agent runs the centralized stepsize-automation approach in [17]. We use the "MovieLens 100k" dataset under heterogeneous data distribution.

Our stepsize-automation approach in (2) is inspired by the centralized stepsize automating approach in [17]:

$$\eta_{t+1} = \min\left\{\sqrt{1 + \frac{\eta_t}{\eta_{t-1}}}\eta_t, \frac{\|x_{t+1} - x_t\|}{2\left\|\nabla f\left(x_{t+1}\right) - \nabla f\left(x_t\right)\right\|}\right\}, \tag{3}$$

However, directly applying the centralized stepsize-automation approach (3) to the distributed setting can easily lead to algorithmic divergence. The reason lies in that allowing each agent to implement (3) using its own local gradients and optimization variables will lead to heterogeneous stepsizes across the agents. What's even worse is that this stepsize heterogeneity is time-varying as agents adapt their stepsizes over iterations, which may easily lead to divergence, as confirmed in our numerical experimental results in Fig. 1.

IV. Main Results

In this section, we prove that Algorithm 1 can indeed avoid algorithmic divergence even when individual agents' stepsizes are heterogeneous and time-varying. To this end, we make the following standard assumption, which is commonly used in distributed optimization under heterogeneous stepsizes [18]–[21]:

Assumption 2: Each local objective function $f_i(x)$ is differentiable, l_i-smooth with some constant $l_i > 0$, and μ_i-strongly convex with some constant $\mu_i > 0$.

Lemma 1: [17] Under Assumption 2, the stepsize $\eta_{i,t}$ in Algorithm 1 satisfies $\frac{1}{2l_i} \leq \eta_{i,t} \leq \frac{1}{2\mu_i}$.

Let us define $\eta_{\max} \triangleq \max_{t\in\mathbb{N}^+, i\in[m]}\{\eta_{i,t}\} \leq \frac{1}{2\mu_{\min}}$ with $\mu_{\min} \triangleq \min_{i\in[m]}\{\mu_i\}$ and $\eta_{\min} \triangleq \min_{t\in\mathbb{N}^+, i\in[m]}\{\eta_{i,t}\} \geq \frac{1}{2l_{\max}}$ with $l_{\max} \triangleq \max_{i\in[m]}\{l_i\}$, respectively.

Lemma 2: Under Assumption 1 and Assumption 2, the following inequalities always hold for Algorithm 1:

$$\lim_{t\to\infty}\|\boldsymbol{x}_t - \mathbf{1}_m\otimes\bar{x}_t\|^2 \leq c_1 \text{ and } \lim_{t\to\infty}\|\boldsymbol{y}_t - \mathbf{1}_m\otimes\bar{y}_t\|^2 \leq c_2, \tag{4}$$

with $c_1 = \frac{8m\rho^2}{(1-\rho)^2}$ and $c_2 = \frac{8(3+\rho)^2 m l_{\max}^2\rho^4}{(1-\rho)^4}$.

Proof: The standard consensus result in [24] implies

$$\|\boldsymbol{x}_{t+1} - \mathbf{1}_m\otimes\bar{x}_{t+1}\| \leq \rho\|\boldsymbol{x}_{\frac{t}{2}} - \mathbf{1}_m\otimes\bar{x}_{t+1}\|, \tag{5}$$

$$\|\boldsymbol{y}_{t+1} - \mathbf{1}_m\otimes\bar{y}_{t+1}\| \leq \rho\|\boldsymbol{y}_{\frac{t}{2}} - \mathbf{1}_m\otimes\bar{y}_{t+1}\|. \tag{6}$$

Based on the 3rd step of Algorithm 1, we have

$$\begin{aligned}&\|\boldsymbol{x}_{\frac{t}{2}} - \mathbf{1}_m\otimes\bar{x}_{t+1}\|\\ &\leq \|\boldsymbol{x}_t - \mathbf{1}_m\otimes\bar{x}_t\| + \|\boldsymbol{\eta}_t\boldsymbol{y}_t\| + \|\mathbf{1}_m\otimes\overline{\eta_t y_t}\|.\end{aligned} \tag{7}$$

The update in the 7th step of Algorithm 1 implies $\|\boldsymbol{\eta}_t\boldsymbol{y}_t\| \leq \sqrt{m}$ and

$$\|\boldsymbol{x}_{\frac{t}{2}} - \mathbf{1}_m\otimes\bar{x}_{t+1}\| \leq \|\boldsymbol{x}_t - \mathbf{1}_m\otimes\bar{x}_t\| + 2\sqrt{m}. \tag{8}$$

Combing (8) and (5), and then iterating the obtained relation from 0 to t yield

$$\|\boldsymbol{x}_t - \mathbf{1}_m\otimes\bar{x}_t\| - b_1 \leq \rho^t\left(\|\boldsymbol{x}_0 - \mathbf{1}_m\otimes\bar{x}_0\| - b_1\right), \tag{9}$$

with $b_1 = \frac{2\sqrt{m}\rho}{1-\rho}$. Taking squares and then the limit on both sides of (9), we arrive at the first inequality in (4).

We proceed to prove the second inequality in (4). Step 5 of Algorithm 1 implies $\bar{y}_t = \frac{1}{m}\sum_{i=1}^m \nabla f_i(x_{i,t})$. Combing this relationship and Assumption 2, we obtain

$$\|\boldsymbol{y}_{\frac{t}{2}} - \mathbf{1}_m\otimes\bar{y}_{t+1}\| \leq \|\boldsymbol{y}_t - \mathbf{1}_m\otimes\bar{y}_t\| + 2l_{\max}\|\boldsymbol{x}_{t+1} - \boldsymbol{x}_t\|. \tag{10}$$

Next, we characterize $\|\boldsymbol{x}_{t+1} - \boldsymbol{x}_t\|$ in (10):

$$\begin{aligned}&\|\boldsymbol{x}_{t+1} - \boldsymbol{x}_t\| \leq \|\boldsymbol{x}_{t+1} - \boldsymbol{x}_{\frac{t}{2}}\| + \|\boldsymbol{x}_{\frac{t}{2}} - \boldsymbol{x}_t\|\\ &\leq \|\boldsymbol{x}_{t+1} - \mathbf{1}_m\otimes\bar{x}_{t+1}\| + \|\mathbf{1}_m\otimes\bar{x}_{t+1} - \boldsymbol{x}_{\frac{t}{2}}\| + \sqrt{m}\\ &\leq (1+\rho)\|\boldsymbol{x}_{\frac{t}{2}} - \mathbf{1}_m\otimes\bar{x}_{t+1}\| + \sqrt{m}\\ &\leq (1+\rho)(\xi + 2\sqrt{m}) + \sqrt{m},\end{aligned} \tag{11}$$

where ξ is given by $\xi = \frac{2\sqrt{m}\rho}{1-\rho} + \rho^t(\|\boldsymbol{x}_0 - \mathbf{1}_m\otimes\bar{x}_0\| - \frac{2\sqrt{m}\rho}{1-\rho})$. Note that in the derivation, we have used the update rule in step 3 of Algorithm 1 and the relation $\|\boldsymbol{\eta}_t\boldsymbol{y}_t\| \leq \sqrt{m}$ in the second inequality. Moreover, we have used (5) in the third inequality, and have used (8) and (9) in the last inequality.

Substituting (11) into (10), and then using (6), we obtain

$$\begin{aligned}\|\boldsymbol{y}_{t+1} - \mathbf{1}_m\otimes\bar{y}_{t+1}\| &\leq \rho\|\boldsymbol{y}_t - \mathbf{1}_m\otimes\bar{y}_t\|\\ &+ 2l_{\max}\rho\left((1+\rho)(\xi + 2\sqrt{m}) + \sqrt{m}\right).\end{aligned} \tag{12}$$

Iterating the inequality (12) from 0 to t yields

$$\|\boldsymbol{y}_t - \mathbf{1}_m\otimes\bar{y}_t\| - b_2 \leq \rho^t\left(\|\boldsymbol{y}_0 - \mathbf{1}_m\otimes\bar{y}_0\| - b_2\right), \tag{13}$$

with $b_2 = \frac{2l_{\max}\rho^2(\xi + 3\sqrt{m} + (2\sqrt{m}+\xi)\rho)}{1-\rho}$. Taking squares and then the limit on both sides of (13), we obtain the second inequality in (4). ■

We proceed to quantify the distance between $x_{i,t}$ and the optimal solution x^*. We first use the following relationship:

$$\begin{aligned}&\|\bar{x}_{t+1} - x^*\|^2 = \|\bar{x}_t - x^*\|^2 - \|\bar{x}_{t+1} - \bar{x}_t\|^2\\ &+ 2\langle\bar{x}_{t+1} - \bar{x}_t, \bar{x}_t - x^*\rangle + 2\langle\bar{x}_{t+1} - \bar{x}_t, \bar{x}_{t+1} - \bar{x}_t\rangle.\end{aligned} \tag{14}$$

To bound the right hand side of (14), we need the following Lemma 3 and Lemma 4.

Lemma 3: Under Assumption 1 and Assumption 2, the following inequality always holds for Algorithm 1:

$$\begin{aligned}&2\langle\bar{x}_{t+1} - \bar{x}_t, \bar{x}_t - x^*\rangle \leq 2\bar{\eta}_t(f(x^*) - f(\bar{x}_t))\\ &\quad + (a_1 + \bar{\eta}_t(a_2 - \bar{\mu}))\|\bar{x}_t - x^*\|^2 + \Delta_{1,t},\ \forall t > 0,\end{aligned} \tag{15}$$

with $a_1, a_2 \in (0,1)$ and $\Delta_{1,t} = \frac{l_{\max}^2\eta_{\max}}{a_2 m}\|\boldsymbol{x}_t - \mathbf{1}_m\otimes\bar{x}_t\|^2 + \frac{\eta_{\max}^2}{a_1 m}\|\boldsymbol{y}_t - \mathbf{1}_m\otimes\bar{y}_t\|^2$.

Proof: Based on the 3rd step of Algorithm 1, we have

$$\begin{aligned}&\langle\bar{x}_{t+1} - \bar{x}_t, \bar{x}_t - x^*\rangle\\ &= -\langle\overline{\eta_t y_t} - \bar{\eta}_t\bar{y}_t, \bar{x}_t - x^*\rangle - \langle\bar{\eta}_t\bar{y}_t, \bar{x}_t - x^*\rangle.\end{aligned} \tag{16}$$

Using the Young's inequality, the first term on the right hand side of (16) satisfies following relation for any $a_1 \in (0,1)$:

$$\begin{aligned}&-\langle\overline{\eta_t y_t} - \bar{\eta}_t\bar{y}_t, \bar{x}_t - x^*\rangle\\ &\leq \frac{1}{2a_1}\frac{1}{m}\sum_{i=1}^m\|\eta_{i,t}(y_{i,t} - \bar{y}_t)\|^2 + \frac{a_1}{2}\|\bar{x}_t - x^*\|^2.\end{aligned} \tag{17}$$

Using Assumption 2, the second term on the right hand side of (16) satisfies the following inequality for any $a_2 \in (0,1)$:

$$\begin{aligned}&-\langle\bar{y}_t, \bar{x}_t - x^*\rangle \leq \frac{l_{\max}^2}{2a_2 m}\|\boldsymbol{x}_t - \mathbf{1}_m\otimes\bar{x}_t\|^2\\ &\quad + \left(\frac{a_2}{2} - \frac{\bar{\mu}}{2}\right)\|\bar{x}_t - x^*\|^2 + f(x^*) - f(\bar{x}_t).\end{aligned} \tag{18}$$

Plugging (17) and (18) into (16) yields (15). ■

Lemma 4: Under Assumption 1 and Assumption 2, the following inequality always holds for Algorithm 1:

$$\begin{aligned}&2\langle\bar{x}_{t+1} - \bar{x}_t, \bar{x}_{t+1} - \bar{x}_t\rangle \leq \Big(\frac{1+a_4}{2} + a_6 + a_7 - \sqrt{2}\bar{\mu}\bar{\eta}_t\Big)\\ &\times\|\bar{x}_t - \bar{x}_{t-1}\|^2 + \Big(\frac{1+a_3}{2} + a_5\Big)\|\bar{x}_{t+1} - \bar{x}_t\|^2\\ &+ 2\sqrt{2}\bar{\eta}_t(f(\bar{x}_{t-1}) - f(\bar{x}_t)) + \Delta_{2,t},\ \forall t > 0,\end{aligned} \tag{19}$$

where a_3 to a_7 are arbitrary numbers within $(0,1)$ and $\Delta_{2,t}$ is given by $\Delta_{2,t} = (1 + \frac{1}{a_4} + \frac{2\eta_{\max}^2 l_{\max}^2}{a_7})\frac{1}{m}\|\boldsymbol{x}_t - \mathbf{1}_m\otimes\bar{x}_t\|^2 + (\frac{\eta_{\max}^2}{a_6} + \frac{\eta_{\max}^2}{a_3})\frac{2}{m}\|\boldsymbol{y}_t - \mathbf{1}_m\otimes\bar{y}_t\|^2 + (1 + \frac{1}{a_4})\frac{1}{m}\|\boldsymbol{x}_{t-1} - \mathbf{1}_m\otimes\bar{x}_{t-1}\|^2 + (\frac{\eta_{\max}^2}{a_5} + \frac{2\eta_{\max}^2}{a_3})\frac{2}{m}\|\boldsymbol{y}_{t-1} - \mathbf{1}_m\otimes\bar{y}_{t-1}\|^2 + \frac{8\eta_{\max}^2 l_{\max}^2}{a_3 m}\left((1+\rho)\xi + 2\sqrt{m}\rho + 3\sqrt{m}\right)^2$ with $\xi = \frac{2\sqrt{m}\rho}{1-\rho} + \rho^t(\|\boldsymbol{x}_0 - \mathbf{1}_m\otimes\bar{x}_0\| - \frac{2\sqrt{m}\rho}{1-\rho})$.

Proof: To prove the inequality in (19), we use the following decomposition:

$$\begin{aligned}2\langle\bar{x}_{t+1} - \bar{x}_t, \bar{x}_{t+1} - \bar{x}_t\rangle &= -\frac{2}{m}\sum_{i=1}^m\langle\eta_{i,t}y_{i,t-1}, \bar{x}_{t+1} - \bar{x}_t\rangle\\ &- \frac{2}{m}\sum_{i=1}^m\langle\eta_{i,t}(y_{i,t} - y_{i,t-1}), \bar{x}_{t+1} - \bar{x}_t\rangle.\end{aligned} \tag{20}$$

Based on step 5 of Algorithm 1, we have

$$\begin{aligned}
&-\frac{2}{m}\sum_{i=1}^{m}\langle \eta_{i,t}(y_{i,t}-y_{i,t-1}), \bar{x}_{t+1}-\bar{x}_t\rangle \\
&\leq \left(1+\frac{1}{a_4}\right)\frac{1}{m}\left(\|\boldsymbol{x}_t-\mathbf{1}_m\otimes\bar{x}_t\|^2+\|\boldsymbol{x}_{t-1}-\mathbf{1}_m\otimes\bar{x}_{t-1}\|^2\right) \\
&\quad+\frac{1+a_3}{2}\|\bar{x}_{t+1}-\bar{x}_t\|^2+\frac{1+a_4}{2}\|\bar{x}_t-\bar{x}_{t-1}\|^2 \\
&\quad+\frac{2\eta_{\max}^2}{a_3 m}(\|\boldsymbol{y}_t-\mathbf{1}_m\otimes\bar{y}_t\|^2+\|\boldsymbol{y}_{\frac{t-1}{2}}-\mathbf{1}_m\otimes\bar{y}_t\|^2). \qquad (21)
\end{aligned}$$

Using the Young's inequality and the definition (2) yields

$$\begin{aligned}
&-\frac{2}{m}\sum_{i=1}^{m}\langle \eta_{i,t}y_{i,t-1}, \bar{x}_{t+1}-\bar{x}_t\rangle\leq\frac{2\eta_{\max}^2}{a_5 m}\|\boldsymbol{y}_{t-1}-\mathbf{1}_m\otimes\bar{y}_{t-1}\|^2 \\
&\quad+a_5\|\bar{x}_{t+1}-\bar{x}_t\|^2+(a_6+a_7-\sqrt{2}\bar{\mu}\bar{\eta}_t)\|\bar{x}_t-\bar{x}_{t-1}\|^2 \\
&\quad+\frac{2\eta_{\max}^2}{a_6 m}\|\boldsymbol{y}_t-\mathbf{1}_m\otimes\bar{y}_t\|^2+\frac{2\eta_{\max}^2 l_{\max}^2}{a_7 m}\|\boldsymbol{x}_t-\mathbf{1}_m\otimes\bar{x}_t\|^2 \\
&\quad+2\sqrt{2}\bar{\eta}_t\left(f(\bar{x}_{t-1})-f(\bar{x}_t)\right). \qquad (22)
\end{aligned}$$

By substituting the results in inequalities (10), (11), (21), and (22) into (20), we can arrive at (19) in Lemma 4. ■

Theorem 1: Under Assumption 1 and Assumption 2, the optimization error of Algorithm 1 satisfies

$$\begin{aligned}
&\lim_{t\to\infty}\|x_{i,t+1}-x^*\|^2 \\
&\leq\frac{16m\rho^2}{(1-\rho)^2}+\frac{2}{1-\gamma}\left[\frac{8\rho^2}{(1-\rho)^2}\left(2+\frac{3l_{\max}^2}{\mu_{\min}^2}+\frac{3+12l_{\max}^3}{\sqrt{2}\mu_{\min}^3}\right)\right. \\
&\left.+\frac{8(3+\rho)^2 l_{\max}^2\rho^4}{(1-\rho)^4}\left(\frac{(36+3\sqrt{2})l_{\max}}{\sqrt{2}\mu_{\min}^3}\right)+\frac{24l_{\max}^3(3+\rho)^2}{\sqrt{2}\mu_{\min}^3(1-\rho)^2}\right], \qquad (23)
\end{aligned}$$

for all $t>0$, where γ is given by $\gamma=\max\{\gamma_1,\gamma_2,\gamma_3\}$ with $\gamma_1=1-\frac{\bar{\mu}}{6l_{\max}}$, $\gamma_2=\frac{6l_{\max}-3\sqrt{2}\bar{\mu}}{6l_{\max}-2\sqrt{2}\bar{\mu}}$, and $\gamma_3\in\left[\frac{2}{1+\sqrt{2}},1\right)$.

Proof: Incorporating (15) and (19) into (14) yields

$$\begin{aligned}
&\|\bar{x}_{t+1}-x^*\|^2+\left(\frac{1-a_3}{2}-a_5\right)\|\bar{x}_{t+1}-\bar{x}_t\|^2 \\
&\quad+\left(2\bar{\eta}_t+2\sqrt{2}\bar{\eta}_t\right)(f(\bar{x}_t)-f(x^*)) \\
&\leq(a_1+\bar{\eta}_t(a_2-\bar{\mu})+1)\|\bar{x}_t-x^*\|^2 \qquad (24) \\
&\quad+\left(\frac{1+a_4}{2}+a_6+a_7-\sqrt{2}\bar{\mu}\bar{\eta}_t\right)\|\bar{x}_t-\bar{x}_{t-1}\|^2 \\
&\quad+2\sqrt{2}\bar{\eta}_t(f(\bar{x}_{t-1})-f(x^*))+\Delta_{1,t}+\Delta_{2,t}.
\end{aligned}$$

We now select parameters a_i $(i=1,\cdots,7)$ to control the coefficients of the items on the right hand side of (24): (i) We set $a_1=\frac{1}{3}\eta_{\min}\bar{\mu}$ and $a_2=\frac{1}{3}\bar{\mu}$, which imply $a_1+\eta_{\min}(a_2-\bar{\mu})+1\leq\gamma_1\times 1$, where $\gamma_1=1-\frac{1}{3}\eta_{\min}\bar{\mu}\in(0,1)$ is always valid since $\eta_{\min}<\frac{3}{\bar{\mu}}$ holds according to Lemma 1. (ii) We select $a_3=a_4=\frac{\sqrt{2}}{3}\bar{\mu}\eta_{\min}$ and $a_5=a_6=a_7=\frac{\sqrt{2}}{6}\bar{\mu}\eta_{\min}$, which imply $\frac{1+a_4}{2}+a_6+a_7-\sqrt{2}\bar{\mu}\eta_{\min}\leq\gamma_2(\frac{1-a_3}{2}-a_5)$ with $\gamma_2=\frac{3-3\sqrt{2}\bar{\mu}\eta_{\min}}{3-2\sqrt{2}\bar{\mu}\eta_{\min}}\in(0,1)$. (iii) Based on (2), we have $\bar{\eta}_t\leq\sqrt{2}\bar{\eta}_{t-1}$, which further implies $2\sqrt{2}\bar{\eta}_t\leq\gamma_3(2\bar{\eta}_{t-1}+2\sqrt{2}\bar{\eta}_{t-1})$, where γ_3 is within the interval $[\frac{2}{1+\sqrt{2}},1)$. Defining $\gamma=\max\{\gamma_1,\gamma_2,\gamma_3\}$ and then iterating (24) from 1 to $t+1$, one obtains

$$\begin{aligned}
&\|\bar{x}_{t+1}-x^*\|^2+\Big(\frac{1}{2}-\frac{\sqrt{2}\bar{\mu}\eta_{\min}}{3}\Big)\|\bar{x}_{t+1}-\bar{x}_t\|^2 \\
&+\left(2\bar{\eta}_t+2\sqrt{2}\bar{\eta}_t\right)(f(\bar{x}_t)-f(x^*)) \\
&\leq\gamma^t\Big[\|\bar{x}_1-x^*\|^2+\Big(\frac{1}{2}-\frac{\sqrt{2}\bar{\mu}\eta_{\min}}{3}\Big)\|\bar{x}_1-\bar{x}_0\|^2 \\
&+2(1+\sqrt{2})\bar{\eta}_0(f(\bar{x}_0)-f(x^*))\Big]+\frac{(\Delta_{1,t}+\Delta_{2,t})(1-\gamma^{t-1})}{1-\gamma}. \qquad (25)
\end{aligned}$$

By taking the limit on both sides of (25) and combining the definitions of $\Delta_{1,t}$ and $\Delta_{2,t}$ in Lemmas 3 and 4 with Lemma 2, we arrive at (23). ■

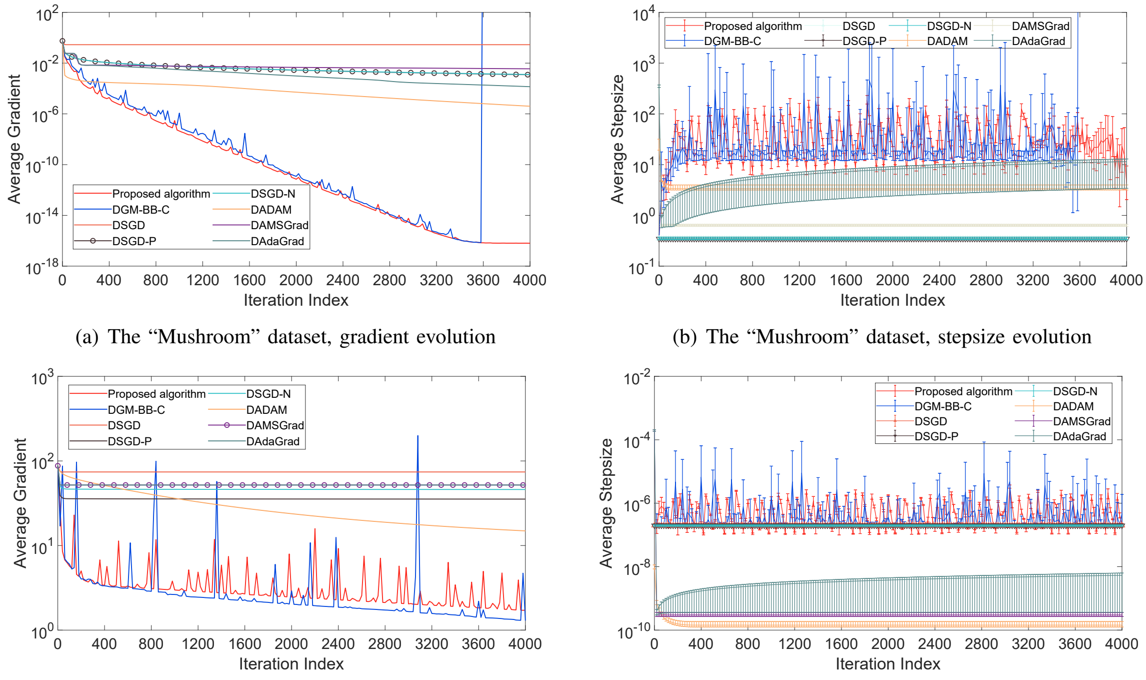
Theorem 1 shows that besides avoiding divergence, Algorithm 1 can ensure convergence to a neighborhood of the optimal solution x^*. The size of this neighborhood is only determined by the size of the network m, the second largest absolute eigenvalue ρ of the weight matrix W, the global Lipschitz parameter $l_{\max}$, and the global strongly convex coefficient $\mu_{\min}$. A larger $\mu_{\min}$ and a smaller $l_{\max}$ will lead to more accurate convergence.

V. Numerical Experiments

In this section, we used three real-world machine learning problems to evaluate the performance of Algorithm 1, including logistic regression using the "Mushroom" dataset and the "Covtype" dataset, respectively, matrix factorization using the "MovieLens 100k" dataset, and image classification using the "MNIST" dataset. For each experiment, we considered heterogeneous data distribution (the distribution of data is non-identical on different agents), which is highly likely in distributed learning applications. In all experiments, we compared Algorithm 1 with distributed optimization algorithms DSGD [25], DSGD with Polyak's momentum (DSGD-P) [25], and DSGD with Nesterov momentum (DSGD-N) [25]. We also compared with the distributed stepsize adaptation methods DGM-BB-C [13], DADAM [15], DAMSGrad [16], and DAdaGrad [16]. The interaction pattern is set as a ring network, where $W=\{w_{ij}\}$ is given by $w_{ii}=0.4$ and $w_{i,i+1}=w_{i,i-1}=0.3$.

A. Logistic Regression

For the first experiment, we ran an l_2-logistic regression classification problem using the "Mushroom" dataset and the "Covtype" dataset [26], respectively. The local objective function for agent i is given by $f_i(x)=\frac{1}{D_i}\sum_{j=1}^{D_i}(1-b_{i,j})a_{i,j}^T x-\log(s(a_{i,j}^T x))+\frac{r_i}{2}\|x\|^2$, where D_i is the number of samples, $r_i>0$ is a regularization parameter, $(a_{i,j},b_{i,j})$ are samples, and $s(t)=1/(1+e^{-t})$ is the sigmoid function. Following [17], we made r_i proportional to $\frac{1}{D_i}$, and selected the initial stepsize as $\frac{1}{L_i}$ with $L_i=\frac{1}{4D_i}\|A_i\|^2+r_i$ for $A_i=\mathrm{col}\{a_{i,j},\cdots,a_{i,D_i}\}$, $j\in D_i$. We spread the data across the agents based on the value of the target, which results in heterogeneous data distributions. For DSGD, DSGD-P, and DSGD-N, the stepsize is fixed to $\frac{1}{L_i}$. For DADAM,



(a) The “Mushroom” dataset, gradient evolution

(b) The “Mushroom” dataset, stepsize evolution

(c) The “Covtype” dataset, gradient evolution

(d) The “Covtype” dataset, stepsize evolution

Fig. 2. Comparison of logistic regression results: panels (a) and (c) depict the average gradient, and panels (b) and (d) depict the average stepsize of all agents.

DAMSGrad, and DAdaGrad, we used the default parameters with $\beta_1 = 0.9$ and $\beta_2 = \beta_3 = 0.99$ given in [16].

Fig. 2 shows that our Algorithm 1 outperforms existing algorithms in terms of both convergence speed and accuracy. It is worth noting that in Fig. 2-(c), DGM-BB-C is slightly faster than our algorithm under the “Covtype” dataset. However, DGM-BB-C is very unstable even for strongly convex and smooth objective functions, because in both the experiment on the “Mushroom” dataset and the experiment for matrix factorization, it leads to divergence (see details in Fig. 2-(a) and Fig. 3). Moreover, DGM-BB-C requires global knowledge of the Lipschitz constant and the strongly convex coefficient, which makes it hard to implement in many practical distributed applications.

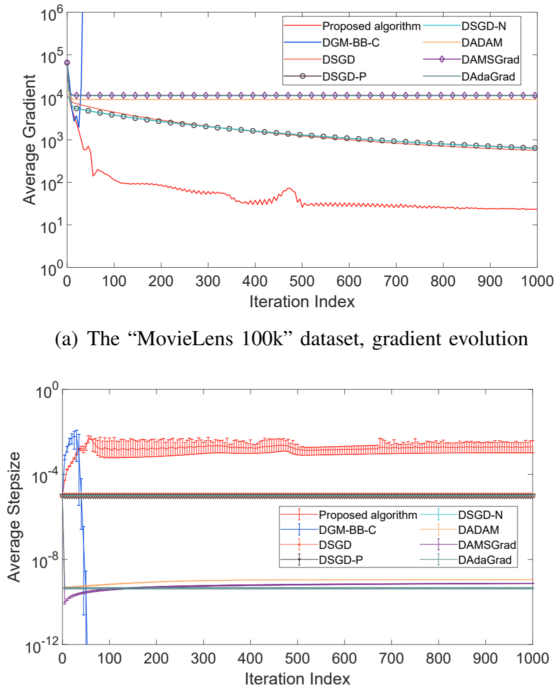
B. Matrix Factorization

For the second experiment, we performed the matrix factorization problem using the “MovieLens 100k” dataset [27], where gradients are not globally Lipschitz. The local objective function for agent i is given by $f_i(U,V) = \frac{1}{2}\|UV^T - A_i\|_F^2$ with $A_i \in \mathbb{R}^{m\times n}$, $U \in \mathbb{R}^{m\times 10}$, and $V \in \mathbb{R}^{n\times 10}$. We split data samples into ten classes and assigned each class to one single agent. In our comparison, we used the best stepsize that we could find for existing distributed algorithms such that doubling the stepsize leads to nonconverging behaviors.

Fig. 3 shows that Algorithm 1 has a faster and more accurate convergence than existing algorithms even when the objective functions are non-smooth and nonconvex.

C. ResNet-18 Training for Image Classification

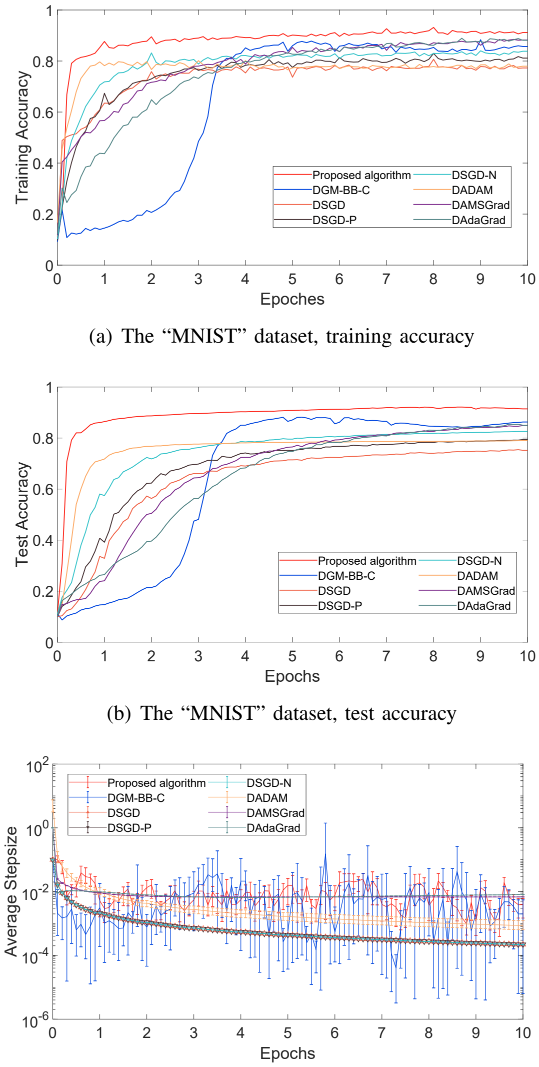
We used a standard ResNet-18 architecture and trained it to classify images from the “MNIST” dataset [28] with cross-entropy loss. We used batch size 60 for all algorithms. In the comparison, following [25], we set $\eta_t = \frac{0.5}{0.07t+1}$ for DSGD, DSGD-P, and DSGD-N. Fig. 4 shows that our algorithm has better training and test accuracies than existing distributed algorithms.



(a) The “MovieLens 100k” dataset, gradient evolution

(b) The “MovieLens 100k” dataset, stepsize evolution

Fig. 3. Comparison of matrix factorization results



(a) The "MNIST" dataset, training accuracy

(b) The "MNIST" dataset, test accuracy

(c) The "MNIST" dataset, stepsize evolution

Fig. 4. Comparison of image classification results

VI. Conclusion

In this paper, we have proposed an algorithm that can automate stepsize selection in distributed optimization and learning with proven convergence guarantees. Note that in the distributed setting, allowing individual agents to adapt their stepsizes results in time-varying stepsize heterogeneity which can easily lead to divergence, so this problem is highly nontrivial. To the best of our knowledge, our approach is the first to successfully automate stepsize in distributed optimization and learning without any manual adjustment. Numerical experimental results on several machine learning problems confirm the effectiveness of the proposed approach.

References

[1] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Trans. Autom. Control*, vol. 54, no. 1, pp. 48–61, 2009.

[2] W. Deng, X. Zeng, and Y. Hong, "Distributed computation for solving the sylvester equation based on optimization," *IEEE Control Syst. Lett.*, vol. 4, no. 2, pp. 414–419, 2019.

[3] M. T. Toghani, S. Lee, and C. A. Uribe, "Pars-push: Personalized, asynchronous and robust decentralized optimization," *IEEE Control Syst. Lett.*, vol. 7, pp. 361–366, 2022.

[4] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011.

[5] R. Rostami, G. Costantini, and D. Görges, "ADMM-based distributed model predictive control: Primal and dual approaches," in *Proc. 56th IEEE Conf. Decis. Control*, 2017, pp. 6598–6603.

[6] R. Carli and M. Dotoli, "Distributed alternating direction method of multipliers for linearly constrained optimization over a network," *IEEE Control Syst. Lett.*, vol. 4, no. 1, pp. 247–252, 2019.

[7] A. Mokhtari, Q. Ling, and A. Ribeiro, "Network Newton distributed optimization methods," *IEEE Trans. Signal Process.*, vol. 65, no. 1, pp. 146–161, 2016.

[8] R. Tutunov, H. Bou-Ammar, and A. Jadbabaie, "Distributed newton method for large-scale consensus optimization," *IEEE Trans. Autom. Control*, vol. 64, no. 10, pp. 3983–3994, 2019.

[9] R. A. Jacobs, "Increased rates of convergence through learning rate adaptation," *Neural Netw.*, vol. 1, no. 4, pp. 295–307, 1988.

[10] T. Schaul, S. Zhang, and Y. LeCun, "No more pesky learning rates," in *Int. Conf. Mach. Learn.*, 2013, pp. 343–351.

[11] S. Xie, M. H. Nazari, G. Yin, *et al.*, "Adaptive step size selection in distributed optimization with observation noise and unknown stochastic target variation," *Automatica*, vol. 135, p. 109940, 2022.

[12] M. Zargham, A. Ribeiro, and A. Jadbabaie, "A distributed line search for network optimization," in *Am. Control Conf.*, 2012, pp. 472–477.

[13] J. Gao, X.-W. Liu, Y.-H. Dai, Y. Huang, and P. Yang, "Achieving geometric convergence for distributed optimization with Barzilai-Borwein step sizes," *Sci. China Inf. Sci.*, vol. 65, no. 4, p. 149204, 2022.

[14] X. Li, B. Karimi, and P. Li, "On distributed adaptive optimization with gradient compression," in *Int. Conf. Learn. Representations*, 2021.

[15] P. Nazari, D. A. Tarzanagh, and G. Michailidis, "DADAM: A consensus-based distributed adaptive gradient method for online optimization," *IEEE Trans. Signal Process.*, vol. 70, pp. 6065–6079, 2022.

[16] X. Chen, B. Karimi, W. Zhao, and P. Li, "On the convergence of decentralized adaptive gradient methods," in *Asian Conf. Mach. Learn.*, 2023, pp. 217–232.

[17] Y. Malitsky and K. Mishchenko, "Adaptive gradient descent without descent," in *Int. Conf. Mach. Learn.* PMLR, 2020, pp. 6702–6712.

[18] J. Xu, S. Zhu, Y. C. Soh, and L. Xie, "Augmented distributed gradient methods for multi-agent optimization under uncoordinated constant stepsizes," in *Proc. 54th IEEE Conf. Decis. Control*. IEEE, 2015, pp. 2055–2060.

[19] G. Qu and N. Li, "Harnessing smoothness to accelerate distributed optimization," *IEEE Trans. Control Netw. Syst.*, vol. 5, no. 3, pp. 1245–1260, 2017.

[20] R. Xin and U. A. Khan, "A linear algorithm for optimization over directed graphs with geometric convergence," *IEEE Control Syst. Lett.*, vol. 2, no. 3, pp. 315–320, 2018.

[21] Q. Lü, H. Li, and D. Xia, "Geometrical convergence rate for distributed optimization with time-varying directed graphs and uncoordinated step-sizes," *Inf. Sci.*, vol. 422, pp. 516–530, 2018.

[22] Y. Wang and A. Nedić, "Decentralized gradient methods with time-varying uncoordinated stepsizes: Convergence analysis and privacy design," *IEEE Trans. Autom. Control (Early Access)*, 2023.

[23] X. Shi, G. Wen, and X. Yu, "Finite-time convergent algorithms for time-varying distributed optimization," *IEEE Control Syst. Lett.*, vol. 7, pp. 3223–3228, 2023.

[24] J. Ma, H. Ji, D. Sun, and G. Feng, "An approach to quantized consensus of continuous-time linear multi-agent systems," *Automatica*, vol. 91, pp. 98–104, 2018.

[25] H. Yu, R. Jin, and S. Yang, "On the linear speedup analysis of communication efficient momentum SGD for distributed non-convex optimization," in *Int. Conf. Mach. Learn.*, 2019, pp. 7184–7193.

[26] D. Dua, C. Graff, *et al.*, "UCI Machine Learning Repository," https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/, 2017.

[27] F. M. Harper and J. A. Konstan, "The MovieLens datasets: History and context," *ACM Trans. Interact. Intell. Syst.*, vol. 5, no. 4, pp. 1–19, 2015.

[28] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.