Compressed Gradient Tracking for Decentralized Optimization Over General Directed Networks

Zhuoqing Song, Lei Shi, Shi Pu, and Ming Yan

***Abstract*—In this paper, we propose two communication-efficient decentralized optimization algorithms over a general directed multi-agent network. The first algorithm, termed Compressed Push-Pull (CPP), combines the gradient tracking Push-Pull method with communication compression. We show that CPP is applicable to a general class of unbiased compression operators and achieves linear convergence rate for strongly convex and smooth objective functions. The second algorithm is a broadcast-like version of CPP (B-CPP), and it also achieves linear convergence rate under the same conditions on the objective functions. B-CPP can be applied in an asynchronous broadcast setting and further reduce communication costs compared to CPP. Numerical experiments complement the theoretical analysis and confirm the effectiveness of the proposed methods.**

***Index Terms*—Decentralized optimization, compression, directed graphs, first-order methods, linear convergence.**

I. Introduction

In THIS paper, we focus on solving the decentralized optimization problem, where a system of n agents, each having access to a private function $f_i(\boldsymbol{x})$, work collaboratively to obtain a consensual solution to the following problem:

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

where $\boldsymbol{x}$ is the global decision variable. The n agents are connected through a general directed network and only communicate directly with their immediate neighbors. The problem (1) has received much attention in recent years due to its wide applications in distributed machine learning [2]–[4], multi-agent target seeking [5], [6], and wireless networks [7]–[9], among many others. For example, the rapid development of distributed machine learning involves data whose size is getting increasingly large, and they are usually stored across multiple computing agents that are spatially distributed. Centering large amounts of data is often undesirable due to limited communication resources and/or privacy concerns, and decentralized optimization serves as an important tool to solve such large-scale distributed learning problems due to its scalability, sparse communication, and better protection for data privacy [10].

Many methods have been proposed to solve the problem (1) under various settings on the optimization objectives, network topologies, and communication protocols. The paper [11] developed a decentralized subgradient descent method (DGD) with diminishing stepsizes to reach the optimum for convex objective functions over an undirected network topology. Subsequently, decentralized optimization methods for undirected networks, or more generally, with doubly stochastic mixing matrices, have been extensively studied in the literature; see, e.g., [12]–[17]. Among these works, EXTRA [15] was the first method that achieves linear convergence for strongly convex and smooth objective functions under symmetric stochastic matrices. For directed networks, however, constructing a doubly stochastic mixing matrix usually requires a weight-balancing step, which could be costly when carried out in a distributed manner. Therefore, the push-sum technique [18] was utilized to overcome this issue. Specifically, the push-sum based subgradient method in [19] can be implemented over time-varying directed graphs, and linear convergence rates were achieved in [20], [21] for minimizing strongly convex and smooth objective functions by applying the push-sum technique to EXTRA.

Gradient tracking is an important technique that has been successfully applied in many decentralized optimization algorithms. Specifically, the methods proposed in [13], [22]–[24] employ gradient tracking to achieve linear convergence for strongly convex and smooth objective functions, where the work in [22]–[24] particularly considered combining gradient tracking with the push-sum technique to accommodate directed graphs. The methods can also be applied to time-varying graphs [22] and asynchronous settings [23]. The Push-Pull/$\mathcal{AB}$ method introduced in [25], [26] modified the gradient tracking methods to deal with directed network topologies without the push-sum technique. The algorithm uses a row stochastic matrix to mix the local decision variables and a column stochastic matrix

Manuscript received August 13, 2021; revised January 1, 2022 and February 14, 2022; accepted March 3, 2022. Date of publication March 17, 2022; date of current version April 20, 2022. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. David I Shuman. The work of Zhuoqing Song was supported by the Shanghai Science and Technology Program under Grant 21JC1400600. The work of Lei Shi was supported in part by the Shanghai Science and Technology Program under Grants 20JC1412700 and 19JC1420101, and in part by the National Natural Science Foundation of China (NSFC) under Grant 12171093. The work of Shi Pu was supported in part by the Shenzhen Research Institute of Big Data under Grant J00120190011, and in part by the National Natural Science Foundation of China (NSFC) under Grant 62003287. The work of Ming Yan was supported by NSF under Grant DMS-2012439. *(Corresponding authors: Shi Pu; Ming Yan.)*

Zhuoqing Song is with the Shanghai Center for Mathematical Sciences, Fudan University, Shanghai 200437, China (e-mail: zqsong19@fudan.edu.cn).

Lei Shi is with the School of Mathematical Sciences, Shanghai Key Laboratory for Contemporary Applied Mathematics, Fudan University, Shanghai 200437, China (e-mail: leishi@fudan.edu.cn).

Shi Pu is with the School of Data Science, Shenzhen Research Institute of Big Data, The Chinese University of Hong Kong, Shenzhen 518172, China (e-mail: shipu.idda@gmail.com).

Ming Yan is with the Department of Computational Mathematics, Science and Engineering and Department of Mathematics, Michigan State University, East Lansing, MI 48864 USA (e-mail: myan@msu.edu).

Digital Object Identifier 10.1109/TSP.2022.3160238

to mix the auxiliary variables that track the average gradients over the network. It can unify different network architectures, including peer-to-peer, master-slave, and leader-follower architectures [25]. For minimizing strongly convex and smooth objectives, the Push-Pull/$\mathcal{AB}$ method not only enjoys linear convergence over fixed graphs [25], [26], but also works well under time-varying graphs and asynchronous settings [25], [27], [28].

In decentralized optimization, efficient communication is critical for enhancing algorithm performance and system scalability. One major approach to reduce communication costs is considering communication compression, which is essential especially under limited communication bandwidth. Recently, several compression methods have been proposed for distributed and federated learning, including [29]–[41]. Recent works have tried to combine the communication compression methods with decentralized optimization. The existence of compression errors may result in inferior convergence performance compared to uncompressed or centralized algorithms. For example, the methods considered by [42]–[47] can only guarantee to reach a neighborhood of the desired solutions when the compression errors exist. QDGD [48] achieves a vanishing mean solution error with a slower rate than the centralized method. To reduce the error from compression, some works [49]–[51] increase compression accuracy as the iteration grows to guarantee the convergence. However, they still need high communication costs to get highly accurate solutions. Techniques to remedy this increased communication costs include gradient difference compression [35], [52], [53] and error compensation [38], [54], [55], which enjoy better performance than direct compression. In [56], the difference compression (DCD-PSGD) and extrapolation compression (ECD-PSGD) algorithms were proposed to reach the same convergence rate of the centralized schemes with additional requirements on the compression ratio. In [55], a quantized decentralized SGD (CHOCO-SGD) method was proposed and shown to converge to the optimal solution or a stationary point at a comparable rate as the centralized SGD method. The works [54] and [57] also achieve comparable convergence rates with the centralized scheme for solving nonconvex problems.

For strongly convex and smooth objective functions, [58] first considered a linearly convergent gradient tracking method based on a specific quantizer. More recently, the paper [53] introduced LEAD that works with a general class of compression operators and still enjoys linear convergence. Some recent developments can be found in [59]–[61], where [60] particularly combined Push-Pull/$\mathcal{AB}$ with a special quantizer to achieve linear convergence over directed graphs.

In this paper, we consider decentralized optimization over general directed networks and propose a novel Compressed Push-Pull method (CPP) that combines Push-Pull/$\mathcal{AB}$ with a general class of unbiased compression operators. CPP enjoys large flexibility in both the compression method and the network topology. We show CPP achieves linear convergence rate under strongly convex and smooth objective functions.

Broadcast or gossip-based communication protocols are popular choices for distributed computation due to their low communication costs [62]–[64]. In the second part of this paper, we propose a broadcast-like CPP algorithm (B-CPP) that allows for asynchronous updates of the agents: at every iteration of the algorithm, only a subset of the agents wake up to perform prescribed updates. Thus, B-CPP is more flexible, and due to its broadcast nature, it can further save communication over CPP in certain scenarios [64]. We show that B-CPP also achieves linear convergence for minimizing strongly convex and smooth objectives.

The main contributions of this paper are summarized as follows.

- We propose CPP – a novel decentralized optimization method with communication compression. The method works under a general class of compression operators and is shown to achieve linear convergence for strongly convex and smooth objective functions over general directed graphs. To the best of our knowledge, CPP is the first method that enjoys linear convergence under such a general setting.
- We consider an asynchronous broadcast version of CPP (B-CPP). B-CPP further reduces the communicated data per iteration and is also provably linearly convergent over directed graphs for minimizing strongly convex and smooth objective functions. Numerical experiments demonstrate the advantages of B-CPP in saving communication costs.

The rest of this paper is organized as follows. We provide necessary notation and assumptions in Section II. CPP is introduced and analyzed in Section III. In Section IV, we consider the algorithm B-CPP. Numerical examples are presented in Section V, and we conclude the paper in Section VI.

II. Notation and Assumptions

Denote the set of agents as $\mathcal{N} = \{1, 2, \ldots, n\}$. At iteration k, each agent i has a local estimation $\boldsymbol{x}_i^k \in \mathbb{R}^p$ of the global decision variable and an auxiliary variable $\boldsymbol{y}_i^k$. We use lowercase bold letters to denote the local variables, and their counterpart uppercase bold letters denote the concatenation of these local variables. For instance, $\boldsymbol{X}^k$, $\nabla \boldsymbol{F}(\boldsymbol{X}^k)$ are the concatenation of $\{\boldsymbol{x}_i^k\}_{i\in\mathcal{N}}$, $\{\nabla f_i(\boldsymbol{x}_i^k)\}_{i\in\mathcal{N}}$, respectively, and their connections are

$$\boldsymbol{X}^k = \left[\boldsymbol{x}_1^k, \ldots, \boldsymbol{x}_n^k\right]^\top \in \mathbb{R}^{n\times p},$$

$$\nabla \boldsymbol{F}(\boldsymbol{X}^k) = \left[\nabla f_1(\boldsymbol{x}_1^k), \ldots, \nabla f_n(\boldsymbol{x}_n^k)\right]^\top \in \mathbb{R}^{n\times p}.$$

Assumption 1: Each f_i is μ-strongly convex ($\mu > 0$) and L-smooth, i.e., for any $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^p$,

$$\langle \boldsymbol{x}_1 - \boldsymbol{x}_2, \nabla f_i(\boldsymbol{x}_1) - \nabla f_i(\boldsymbol{x}_2)\rangle \geq \mu \left\|\boldsymbol{x}_1 - \boldsymbol{x}_2\right\|_2^2,$$

$$\left\|\nabla f_i(\boldsymbol{x}_1) - \nabla f_i(\boldsymbol{x}_2)\right\|_2 \leq L \left\|\boldsymbol{x}_1 - \boldsymbol{x}_2\right\|_2.$$

Since all $f_i(\boldsymbol{x})$ are strongly convex, $f(\boldsymbol{x})$ admits a unique minimizer $\boldsymbol{x}^*$. Denote $\boldsymbol{x}^* = \mathbf{1}\boldsymbol{X}^{*\top}$, where $\mathbf{1}$ is the all-ones column vector.

Given any nonnegative matrix $\boldsymbol{M}$, we denote by $\mathcal{G}_{\boldsymbol{M}}$ the induced graph by $\boldsymbol{M}$. The sets $\mathcal{N}_{\boldsymbol{M},i}^- \overset{\text{def}}{=} \{j \in \mathcal{N} : \boldsymbol{M}_{ij} > 0\}$ and $\mathcal{N}_{\boldsymbol{M},i}^+ \overset{\text{def}}{=} \{j \in \mathcal{N} : \boldsymbol{M}_{ji} > 0\}$ are called the in-neighbors and out-neighbors of agent i.

The communication between all the agents is modeled by directed graphs. Given a strongly connected graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ with $\mathcal{E} \subset \mathcal{N} \times \mathcal{N}$ being the edge set, agent i can receive information from agent j if and only if $(i, j) \in \mathcal{E}$. There are

two n-by-n nonnegative matrices $\boldsymbol{R}$ and $\boldsymbol{C}$. A spanning tree $\mathcal{T}$ rooted at some $i \in \mathcal{N}$ in $\mathcal{G}_{\boldsymbol{R}}$ is a subgraph of $\mathcal{G}_{\boldsymbol{R}}$ containing $n-1$ edges, and each node except i can be connected to i by a path in $\mathcal{T}$. Let $\mathcal{R}_{\boldsymbol{R}}, \mathcal{R}_{\boldsymbol{C}^\top}$ denote the roots of the spanning trees in $\mathcal{G}_{\boldsymbol{R}}$ and $\mathcal{G}_{\boldsymbol{C}^\top}$. We have the following assumption on $\boldsymbol{R}$ and $\boldsymbol{C}$.

Assumption 2: The matrix $\boldsymbol{R}$ is supported by $\mathcal{G}$, i.e., $\mathcal{E}_{\boldsymbol{R}} = \{(i,j) \in \mathcal{N} \times \mathcal{N} \big| \boldsymbol{R}_{ij} > 0\} \subset \mathcal{E}$, and $\boldsymbol{R}$ is a row stochastic matrix, i.e., $\boldsymbol{R}\mathbf{1} = \mathbf{1}$. The matrix $\boldsymbol{C}$ is also supported by $\mathcal{G}$, and $\boldsymbol{C}$ is a column stochastic matrix, i.e., $\mathbf{1}^\top \boldsymbol{C} = \mathbf{1}^\top$. In addition, $\mathcal{R}_{\boldsymbol{R}} \cap \mathcal{R}_{\boldsymbol{C}^\top} \neq \emptyset$.

By [25, Lemma 1], Assumption 2 implies the following facts: $\boldsymbol{R}$ has a unique left eigenvector $\boldsymbol{s}_{\mathrm{R}}$ with respect to 1 such that $\boldsymbol{s}_{\mathrm{R}}^\top \mathbf{1} = n$. $\boldsymbol{C}$ has a unique right eigenvector $\boldsymbol{s}_{\mathrm{C}}$ with respect to 1 such that $\boldsymbol{s}_{\mathrm{C}}^\top \mathbf{1} = n$. In addition, the entries of $\boldsymbol{s}_{\mathrm{R}}$ and $\boldsymbol{s}_{\mathrm{C}}$ are all nonnegative. All nonzero entries of $\boldsymbol{s}_{\mathrm{R}}$, $\boldsymbol{s}_{\mathrm{C}}$ correspond to $\mathcal{R}_{\boldsymbol{R}}$, $\mathcal{R}_{\boldsymbol{C}^\top}$, respectively. Because $\mathcal{R}_{\boldsymbol{R}} \cap \mathcal{R}_{\boldsymbol{C}^\top} \neq \emptyset$, we have $\boldsymbol{s}_{\mathrm{R}}^\top \boldsymbol{s}_{\mathrm{C}} > 0$.

We denote the spectral radius of matrix $\boldsymbol{A}$ as $\rho(\boldsymbol{A})$. The inner product of two matrices is defined as $\langle \boldsymbol{X}, \boldsymbol{Y} \rangle = \mathbf{tr}(\boldsymbol{X}^\top \boldsymbol{Y})$, and the Frobenius norm is $\|\boldsymbol{X}\|_{\mathrm{F}} = \sqrt{\langle \boldsymbol{X}, \boldsymbol{X} \rangle}$. Given a vector $\boldsymbol{d}$, $\boldsymbol{d}_{a:b}$ is the subvector of $\boldsymbol{d}$ containing the entries indexed from a to b. Given a matrix $\boldsymbol{A}$, the notion $\boldsymbol{A}_{a:b,c:d}$ denotes the submatrix containing the entries with row index in $[a,b]$ and column index in $[c,d]$. We abbreviate "$1:n$" by ":" and "$i:i$" is abbreviated as "i". Especially in our notations, $\boldsymbol{A}_{i,:}$ and $\boldsymbol{A}_{:,j}$ denote the i-th row and j-th column of $\boldsymbol{A}$, respectively. For vector $\boldsymbol{v} \in \mathbb{R}^m$, $\mathbf{Diag}(\boldsymbol{v})$ is an m-by-m diagonal matrix with $\mathbf{Diag}(\boldsymbol{v})_{ii} = \boldsymbol{v}_i$.

Definition 1: Given a vector norm $\|\cdot\|_*$, we define the corresponding matrix norm on an $n \times p$ matrix $\boldsymbol{A}$ as

$$|||\boldsymbol{A}|||_* = \left\| \left[\|\boldsymbol{A}_{:,1}\|_*, \|\boldsymbol{A}_{:,2}\|_*, \ldots, \|\boldsymbol{A}_{:,p}\|_* \right] \right\|_2 .$$

When $\|\cdot\|_* = \|\cdot\|_2$, we have $|||\boldsymbol{A}|||_2 = \|\boldsymbol{A}\|_{\mathrm{F}}$, the Frobenius norm.

Definition 2: Given a vector norm $\|\cdot\|_*$, we define the corresponding induced norm on an $n \times n$ matrix $\boldsymbol{A}$ as $\|\boldsymbol{A}\|_* = \sup_{\boldsymbol{x} \neq 0} \frac{\|\boldsymbol{A}\boldsymbol{x}\|_*}{\|\boldsymbol{x}\|_*}$.

Lemma 1 (Lemma 5 in [25]): For any matrices $\boldsymbol{A} \in \mathbb{R}^{n \times p}$, $\boldsymbol{W} \in \mathbb{R}^{n \times n}$, and a vector norm $\|\cdot\|_*$, we have $|||\boldsymbol{W}\boldsymbol{A}|||_* \leq \|\boldsymbol{W}\|_* |||\boldsymbol{A}|||_*$. For any vectors $\boldsymbol{a} \in \mathbb{R}^n$, $\boldsymbol{b} \in \mathbb{R}^p$, $|||\boldsymbol{a}\boldsymbol{b}^\top|||_* \leq \|\boldsymbol{a}\|_* \|\boldsymbol{b}\|_2$.

The compression in this paper is denoted by $\textsc{Compress}(\cdot)$. For any matrix $\boldsymbol{X} \in \mathbb{R}^{n \times p}$, we denote $\textsc{Compress}(\boldsymbol{X})$ as an n-by-p matrix with the i-th row being $\textsc{Compress}(\boldsymbol{X}_{i,:})$.

Assumption 3: The compression is unbiased, i.e., given $\boldsymbol{x} \in \mathbb{R}^p$ and $\widehat{\boldsymbol{x}} = \textsc{Compress}(\boldsymbol{x})$, there exists $C_2 > 0$ such that $\mathbb{E}[\widehat{\boldsymbol{x}} \big| \boldsymbol{x}] = \boldsymbol{x}$, and $\mathbb{E}[\|\widehat{\boldsymbol{x}} - \boldsymbol{x}\|_2^2 \big| \boldsymbol{x}] \leq C_2 \|\boldsymbol{x}\|_2^2$. And the random variables generated inside the procedure $\textsc{Compress}(\boldsymbol{x})$ depends on $\boldsymbol{x}$ only.

III. A Push-Pull Method with Compression

In this section, we propose a Push-Pull method with Compression (CPP) in Algorithm 1.

Algorithm 1: Compressed Push-Pull (in agent i 's perspective).

Input: initial position $\boldsymbol{x}_i^0$, stepsize α_i, averaging parameters $\beta, \gamma, \eta \in (0,1]$, network information $\boldsymbol{R}_{ij} (j \in \mathcal{N}_{\mathrm{R},i}^-)$, $\boldsymbol{C}_{ij} (j \in \mathcal{N}_{\mathrm{C},i}^-)$ and the sets $\mathcal{N}_{\mathrm{R},i}^+$, $\mathcal{N}_{\mathrm{C},i}^+$, total iteration number K

Output: $\boldsymbol{x}_i^K$

Initiate $\boldsymbol{y}_i^0 = \nabla f_i(\boldsymbol{x}_i^0)$, $\boldsymbol{u}_i^0 = \boldsymbol{u}_{\mathrm{R},i}^0 = \mathbf{0}$.

for $k = 0$ *to* $K-1$ **do**

Call procedure PULL: $\left(\widehat{\boldsymbol{x}}_{\mathrm{R},i}^k, \boldsymbol{u}_i^{k+1}, \boldsymbol{u}_{\mathrm{R},i}^{k+1}\right) =$ $\mathrm{PULL}\left(\boldsymbol{x}_i^k, \boldsymbol{u}_i^k, \boldsymbol{u}_{\mathrm{R},i}^k\right)$.

Update $\boldsymbol{x}_i^{k+1} = (1-\beta)\boldsymbol{x}_i^k + \beta \widehat{\boldsymbol{x}}_{\mathrm{R},i}^k - \alpha_i \boldsymbol{y}_i^k$.

Call procedure PUSH: $\left(\widehat{\boldsymbol{y}}_i^k, \widehat{\boldsymbol{y}}_{\mathrm{C},i}^k\right) = \mathrm{PUSH}\left(\boldsymbol{y}_i^k\right)$.

Compute local gradient $\nabla f_i\left(\boldsymbol{x}_i^{k+1}\right)$.

Update $\boldsymbol{y}_i^{k+1} =$ $\boldsymbol{y}_i^k + \gamma \widehat{\boldsymbol{y}}_{\mathrm{C},i}^k - \gamma \widehat{\boldsymbol{y}}_i^k + \nabla f_i\left(\boldsymbol{x}_i^{k+1}\right) - \nabla f_i\left(\boldsymbol{x}_i^k\right)$.

end

procedure $\mathrm{PULL}\left(\boldsymbol{x}_i, \boldsymbol{u}_i, \boldsymbol{u}_{\mathrm{R},i}\right)$

$\boldsymbol{p}_i = \textsc{Compress}\left(\boldsymbol{x}_i - \boldsymbol{u}_i\right)$.

$\boldsymbol{p}_{\mathrm{R},i} = \sum_{j \in \mathcal{N}_{\mathrm{R},i}^-} \boldsymbol{R}_{ij} \boldsymbol{p}_j$.

$\widehat{\boldsymbol{x}}_i = \boldsymbol{u}_i + \boldsymbol{p}_i$.

$\widehat{\boldsymbol{x}}_{\mathrm{R},i} = \boldsymbol{u}_{\mathrm{R},i} + \boldsymbol{p}_{\mathrm{R},i}$.

$\boldsymbol{u}_i \leftarrow (1-\eta)\, \boldsymbol{u}_i + \eta \widehat{\boldsymbol{x}}_i$.

$\boldsymbol{u}_{\mathrm{R},i} \leftarrow (1-\eta)\, \boldsymbol{u}_{\mathrm{R},i} + \eta \widehat{\boldsymbol{x}}_{\mathrm{R},i}$.

Return: $\widehat{\boldsymbol{x}}_{\mathrm{R},i}, \boldsymbol{u}_i, \boldsymbol{u}_{\mathrm{R},i}$.

end procedure

procedure $\mathrm{PUSH}\left(\boldsymbol{y}_i\right)$

$\widehat{\boldsymbol{y}}_i = \textsc{Compress}\left(\boldsymbol{y}_i\right)$.

$\widehat{\boldsymbol{y}}_{\mathrm{C},i} = \sum_{j \in \mathcal{N}_{\mathrm{C},i}^-} \boldsymbol{C}_{ij} \widehat{\boldsymbol{y}}_j$.

Return: $\widehat{\boldsymbol{y}}_i, \widehat{\boldsymbol{y}}_{\mathrm{C},i}$.

end procedure

We start from discussing the following scheme of Push-Pull/$\mathcal{AB}$ from the viewpoint of agent i [25], [26]:

$$\begin{cases} \boldsymbol{x}_i^{k+1} = \sum_{j \in \mathcal{N}_{\mathrm{R,i}}^-} \boldsymbol{R}_{ij} \boldsymbol{x}_j^k - \alpha_i \boldsymbol{y}_i^k \\ \boldsymbol{y}_i^{k+1} = \sum_{j \in \mathcal{N}_{\mathrm{C,i}}^-} \boldsymbol{C}_{ij} \boldsymbol{y}_j^k + \nabla f_i\left(\boldsymbol{x}_i^{k+1}\right) - \nabla f_i\left(\boldsymbol{x}_i^k\right) \end{cases}$$

and $\boldsymbol{y}_i^0 = \nabla f_i(\boldsymbol{x}_i^0)$. At each iteration, agent i computes $\sum_{j \in \mathcal{N}_{\mathrm{R},i}^-} \boldsymbol{R}_{ij} \boldsymbol{x}_j^k$ to average the local decision variables received from its neighbors. As each agent takes such a step, the local decision variables tend to get closer with each other and eventually reach consensus when $\boldsymbol{y}_i^k$ goes to zero. In the next "gradient tracking" step, by adding the gradient difference term $\nabla f_i(\boldsymbol{x}_i^{k+1}) - \nabla f_i(\boldsymbol{x}_i^k)$ into $\boldsymbol{y}_i^{k+1}$ and considering that $\boldsymbol{y}_i^0 = \nabla f_i(\boldsymbol{x}_i^0)$ and $\mathbf{1}^\top \boldsymbol{C} = \mathbf{1}^\top$, we have $\sum_{i \in \mathcal{N}} \boldsymbol{y}_i^k = \sum_{i \in \mathcal{N}} \nabla f_i(\boldsymbol{x}_i^k)$ by induction, which indicates that $\{\boldsymbol{y}_i^k\}_{i \in \mathcal{N}}$ can help track the average gradients over the network.

The Compressed Push-Pull (CPP) method differs from Push-Pull/$\mathcal{AB}$ mainly in the averaging step which requires communication. To alleviate the impact of compression errors, we use a "damped" averaging step and replace the exact local variables by their communication-efficient version, i.e., $(1-\beta)\boldsymbol{x}_i^k + \beta\sum_{j\in\mathcal{N}_{\mathrm{R,i}}^-}\boldsymbol{R}_{ij}\widehat{\boldsymbol{x}}_j^k$. For $\boldsymbol{y}_i^k$, we take a slightly different averaging step so that the relation $\sum_{i\in\mathcal{N}}\boldsymbol{y}_i^k = \sum_{i\in\mathcal{N}}\nabla f_i(\boldsymbol{x}_i^k)$ is preserved and the compression errors can also be bounded "safely".

However, sending $\widehat{\boldsymbol{x}}_i^k$ itself can still be expensive. Therefore, we call the procedure PULL to save the communication costs. In this procedure, $(\boldsymbol{u}_i, \boldsymbol{u}_{\mathrm{R},i})$ is used to track $(\boldsymbol{x}_i, \sum_{j\in\mathcal{N}_{\mathrm{R,i}}^-}\boldsymbol{R}_{ij}\boldsymbol{x}_j)$. The compression and the communication are applied on the difference $(\boldsymbol{x}_i - \boldsymbol{u}_i)$ and its compressed version, respectively. It can be derived from the relation $\boldsymbol{p}_{\mathrm{R},i} = \sum_{j\in\mathcal{N}_{\boldsymbol{R},i}^-}\boldsymbol{R}_{ij}\boldsymbol{p}_j$ and induction that in each call to PULL, $\boldsymbol{u}_{\mathrm{R,i}} = \sum_{j\in\mathcal{N}_{\mathrm{R,i}}^-}\boldsymbol{R}_{ij}\boldsymbol{u}_j$. And the compression error could be very small if the difference $\boldsymbol{x}_i - \boldsymbol{u}_i$ is small. With this observation, we have $\widehat{\boldsymbol{x}}_{\mathrm{R,i}} = \sum_{j\in\mathcal{N}_{\mathrm{R,i}}^-}\boldsymbol{R}_{ij}\widehat{\boldsymbol{x}}_j$ with $\widehat{\boldsymbol{x}}_j$ being an unbiased approximation of $\boldsymbol{x}_j$, whose variance converges to 0.

In the PUSH procedure, since the variable $\boldsymbol{y}_i^k$ converges to zero as we will show later, we can simply compress it and estimate $\sum_{j\in\mathcal{N}_{\mathrm{C,i}}^-}\boldsymbol{C}_{ij}\boldsymbol{y}_j^k$ using the compressed values. Similarly, we have that in each call to PUSH, $\widehat{\boldsymbol{y}}_{\mathrm{C},i} = \sum_{j\in\mathcal{N}_{\mathrm{C,i}}^-}\boldsymbol{C}_{ij}\widehat{\boldsymbol{y}}_j$, with $\widehat{\boldsymbol{y}}_j$ being the unbiased compression of $\boldsymbol{y}_j$, whose variance converges to 0 as the input $\boldsymbol{y}_i^k$ converges to $\boldsymbol{0}$.

Let $\widehat{\alpha} = \max_{1\le i\le n}\alpha_i$, and define $\boldsymbol{\alpha}$ as a diagonal matrix with $\boldsymbol{\alpha}_{ii} = \alpha_i$. Then, we can rewrite Algorithm 1 into the following matrix form

$$\begin{cases} \widehat{\boldsymbol{X}}^k = \boldsymbol{U}^k + \textsc{Compress}\left(\boldsymbol{X}^k - \boldsymbol{U}^k\right), \\ \widehat{\boldsymbol{Y}}^k = \textsc{Compress}\left(\boldsymbol{Y}^k\right), \\ \boldsymbol{x}^{k+1} = (1-\beta)\,\boldsymbol{x}^k + \beta\boldsymbol{R}\widehat{\boldsymbol{X}}^k - \boldsymbol{\alpha}\boldsymbol{Y}^k, \\ \boldsymbol{Y}^{k+1} = \boldsymbol{Y}^k + \gamma\,(\boldsymbol{C}-\boldsymbol{I})\,\widehat{\boldsymbol{Y}}^k \\ \qquad\quad +\nabla\boldsymbol{F}(\boldsymbol{X}^{k+1}) - \nabla\boldsymbol{F}(\boldsymbol{X}^k), \\ \boldsymbol{U}^{k+1} = (1-\eta)\,\boldsymbol{U}^k + \eta\widehat{\boldsymbol{X}}^k. \end{cases} \tag{2}$$

Comparing to the standard Push-Pull method in [25], the procedures PUSH and PULL save communication costs at the cost of error in the mixing. Note $\widehat{\boldsymbol{X}}^k$ and $\widehat{\boldsymbol{Y}}^k$ are approximations for $\boldsymbol{X}^k$ and $\boldsymbol{Y}^k$, respectively. The difference between $\boldsymbol{X}^k$ (or $\boldsymbol{Y}^k$) and $\widehat{\boldsymbol{X}}^k$ (or $\widehat{\boldsymbol{Y}}^k$) are induced by the compression errors. Let us denote the approximation error as $\boldsymbol{w}_i^k = \boldsymbol{x}_i^k - \widehat{\boldsymbol{x}}_i^k$ and $\boldsymbol{e}_i^k = \boldsymbol{y}_i^k - \widehat{\boldsymbol{y}}_i^k$. We further write them into the following matrix form

$$\boldsymbol{W}^k = \boldsymbol{X}^k - \widehat{\boldsymbol{X}}^k, \boldsymbol{E}^k = \boldsymbol{Y}^k - \widehat{\boldsymbol{Y}}^k. \tag{3}$$

rom the PULL procedure, we can see that $\boldsymbol{W}^k$ is also the compression error for $\boldsymbol{X}^k - \boldsymbol{U}^k$. The update of $\boldsymbol{U}^{k+1} = (1-\eta)\boldsymbol{U}^k + \eta\widehat{\boldsymbol{X}}^k$ indicates that $\boldsymbol{U}^k$ is tracking the motions of $\boldsymbol{X}^k$. As $\boldsymbol{U}^k$ approaches $\boldsymbol{X}^k$, by Assumption 3, the variance of the approximation error $\boldsymbol{W}^k$ will also tends to $\boldsymbol{0}$.

Since $\boldsymbol{Y}^0 = \nabla\boldsymbol{F}(\boldsymbol{X}^0)$ and $\boldsymbol{1}^\top\boldsymbol{C} = \boldsymbol{1}^\top$, it follows from (5d) and induction that

$$\boldsymbol{1}^\top\boldsymbol{Y}^k = \boldsymbol{1}^\top\nabla\boldsymbol{F}\left(\boldsymbol{X}^k\right). \tag{4}$$

Then, as the local variables become closer to each other, $\boldsymbol{y}_i^k$ are more and more parallel to $\boldsymbol{1}^\top\nabla\boldsymbol{F}(\boldsymbol{X}^k)$. As $\boldsymbol{X}^k$ approaches the optimal point, $\boldsymbol{1}^\top\nabla\boldsymbol{F}(\boldsymbol{X}^k)$ tends to $\boldsymbol{1}^\top\nabla\boldsymbol{F}\left(\boldsymbol{X}^*\right) = \boldsymbol{0}^\top$. This indicates that $\boldsymbol{y}^k$ also tends to $\boldsymbol{0}$. Then, by Assumption 3, the variance of compression error $\boldsymbol{E}^k$ will also tend to $\boldsymbol{0}$. As the compression errors $\boldsymbol{W}^k$, $\boldsymbol{E}^k$ tend to $\boldsymbol{0}$, we will have $\widehat{\boldsymbol{X}}^k \approx \boldsymbol{X}^k$ and $\widehat{\boldsymbol{Y}}^k \approx \boldsymbol{Y}^k$. We remark that if $\widehat{\boldsymbol{X}}^k = \boldsymbol{X}^k$, $\widehat{\boldsymbol{Y}}^k = \boldsymbol{Y}^k$, Algorithm 1 will reduce to the PUSH-PULL method in [25].

To show the convergence of the proposed compressed algorithm, we define $\boldsymbol{R}_\beta = (1-\beta)\boldsymbol{I} + \beta\boldsymbol{R}$, $\boldsymbol{C}_\gamma = (1-\gamma)\boldsymbol{I} + \gamma\boldsymbol{C}$, and rewrite (2) as

$$\begin{cases} \boldsymbol{W}^k = \boldsymbol{X}^k - \boldsymbol{U}^k - \textsc{Compress}\left(\boldsymbol{X}^k - \boldsymbol{U}^k\right), & \text{(5a)} \\ \boldsymbol{E}^k = \boldsymbol{Y}^k - \textsc{Compress}\left(\boldsymbol{Y}^k\right), & \text{(5b)} \\ \boldsymbol{X}^{k+1} = \boldsymbol{R}_\beta\boldsymbol{X}^k - \boldsymbol{\alpha}\boldsymbol{Y}^k - \beta\boldsymbol{R}\boldsymbol{W}^k, & \text{(5c)} \\ \boldsymbol{Y}^{k+1} = \boldsymbol{C}_\gamma\boldsymbol{Y}^k + \nabla\boldsymbol{F}(\boldsymbol{X}^{k+1}) - \nabla\boldsymbol{F}(\boldsymbol{X}^k) \\ \qquad\qquad +\gamma\left(\boldsymbol{I}-\boldsymbol{C}\right)\boldsymbol{E}^k, & \text{(5d)} \\ \boldsymbol{U}^{k+1} = (1-\eta)\boldsymbol{U}^k + \eta\boldsymbol{X}^k - \eta\boldsymbol{W}^k, & \text{(5e)} \end{cases}$$

We define the value $\bar{\alpha} = \boldsymbol{s}_{\mathrm{R}}^\top\boldsymbol{\alpha}\boldsymbol{s}_{\mathrm{C}}/n$. Under Assumption 2, $\bar{\alpha} > 0$ can be satisfied by setting $\alpha_i > 0$ for at least one $i \in \mathcal{R}_{\boldsymbol{R}} \cap \mathcal{R}_{\boldsymbol{C}^\top}$. When $\bar{\alpha} > 0$, let us define a constant $w \ge \widehat{\alpha}/\bar{\alpha} \ge \frac{n}{\boldsymbol{s}_{\mathrm{R}}^\top\boldsymbol{s}_{\mathrm{C}}}$. Note that the analysis in the rest of this section holds true for arbitrary $w \ge \widehat{\alpha}/\bar{\alpha}$. As we will choose $w \ge \widehat{\alpha}/\bar{\alpha}$ as an auxiliary parameter to help choose proper stepsizes $\{\alpha_i\}_{i\in\mathcal{N}}$ in Theorem 7 which is the main theorem of this section, we use the same w in the analysis below.

Let $\mathcal{F}_k$ denote the σ-field generated by $\{\boldsymbol{E}^j, \boldsymbol{W}^j\}_{j=0}^{k-1}$, and $\mathcal{F}_k^+$ is the σ-field generated by $\{\boldsymbol{E}^j, \boldsymbol{W}^j\}_{j=0}^{k-1} \cup \{\boldsymbol{W}^k\}$. Thus, we have $\mathcal{F}_1 \subset \mathcal{F}_1^+ \subset \mathcal{F}_2 \subset \mathcal{F}_2^+ \subset \cdots \subset \mathcal{F}_k \subset \mathcal{F}_k^+ \subset \cdots$. By Lemma 2 below, $\mathcal{F}_k = \sigma(\boldsymbol{X}^i, \boldsymbol{Y}^i, \boldsymbol{U}^i : 0 \le i < k)$ and $\mathcal{F}_k^+ = \sigma(\boldsymbol{X}^i, \boldsymbol{U}^i, \boldsymbol{Y}^j : 0 \le i \le k, 0 \le j < k)$.

The following lemma comes from (5c)–(5e) and Assumption 3.

Lemma 2: The variables $\boldsymbol{X}^k, \boldsymbol{Y}^k, \boldsymbol{U}^k, \boldsymbol{V}^k$ are measurable with respect to $\mathcal{F}_k$, and $\boldsymbol{X}^k$ is measurable with respect to $\mathcal{F}_{k-1}^+$. Moreover, we have

$$\mathbb{E}\left[\boldsymbol{W}^k|\mathcal{F}_k\right] = \mathbb{E}\left[\boldsymbol{E}^k|\mathcal{F}_k^+\right] = \boldsymbol{0}. \tag{6}$$

Proof: By expanding (5c)–(5e) recursively, $\boldsymbol{X}^k$, $\boldsymbol{Y}^k$, $\boldsymbol{U}^k$, and $\boldsymbol{V}^k$ can be represented by linear combinations of $\boldsymbol{X}^0, \boldsymbol{Y}^0, \boldsymbol{U}^0, \boldsymbol{V}^0$ and random variables $\{\boldsymbol{E}^j, \boldsymbol{W}^j\}_{j=0}^{k-1}$, i.e., $\boldsymbol{X}^k, \boldsymbol{Y}^k, \boldsymbol{U}^k, \boldsymbol{V}^k$ are measurable with respect to $\mathcal{F}_k$. By (5c), $\boldsymbol{X}^k$ is a linear combination of $\boldsymbol{W}^{k-1}$ and $\mathcal{F}_{k-1}$-measurable variables $\boldsymbol{X}^{k-1}, \boldsymbol{Y}^{k-1}$, thus, $\boldsymbol{X}^k$ is measurable with respect to $\mathcal{F}_{k-1}^+$. And we obtain (6) directly from Assumption 3. ■

A. Convergence Analysis for CPP

In this section, we analyze the convergence rates of Algorithm 1. To begin with, we define the averages of $\boldsymbol{X}^k$, $\boldsymbol{Y}^k$ as follows,

$$\overline{\boldsymbol{x}}^k = \frac{1}{n}\boldsymbol{s}_{\mathrm{R}}^\top\boldsymbol{X}^k,\ \overline{\boldsymbol{Y}}^k = \frac{1}{n}\boldsymbol{1}^\top\boldsymbol{Y}^k. \tag{7}$$

We define two matrices $\boldsymbol{\Pi}_{\mathrm{R}}=\boldsymbol{I}-\frac{\mathbf{1}\boldsymbol{s}_{\mathrm{R}}^{\top}}{n}$, $\boldsymbol{\Pi}_{\mathrm{C}}=\boldsymbol{I}-\frac{\boldsymbol{s}_{\mathrm{C}}\mathbf{1}^{\top}}{n}$. It follows from direct calculation that $\boldsymbol{\Pi}_{\mathrm{R}}\boldsymbol{R}=\boldsymbol{R}\boldsymbol{\Pi}_{\mathrm{R}}=\boldsymbol{R}-\frac{\mathbf{1}\boldsymbol{s}_{\mathrm{R}}^{\top}}{n}$,$\boldsymbol{\Pi}_{\mathrm{C}}\boldsymbol{C}=\boldsymbol{C}\boldsymbol{\Pi}_{\mathrm{C}}=\boldsymbol{C}-\frac{\boldsymbol{s}_{\mathrm{C}}\mathbf{1}^{\top}}{n}$,$\boldsymbol{\Pi}_{\mathrm{R}}\boldsymbol{X}^k=\boldsymbol{X}^k-\mathbf{1}\overline{\boldsymbol{x}}_k$, $\boldsymbol{\Pi}_{\mathrm{C}}\boldsymbol{Y}^k=\boldsymbol{Y}^k-\boldsymbol{s}_{\mathrm{C}}\overline{\boldsymbol{Y}}_k$,$(\boldsymbol{R}-\boldsymbol{I})\boldsymbol{\Pi}_{\mathrm{R}}=\boldsymbol{R}-\boldsymbol{I}$, and $\boldsymbol{\Pi}_{\mathrm{C}}(\boldsymbol{I}-\boldsymbol{C})=\boldsymbol{I}-\boldsymbol{C}$.

In the analysis below, $\mathbb{E}\left[\|\overline{\boldsymbol{x}}^k-\boldsymbol{x}^*\|_2^2\right]$ is employed to measure the closeness to the optimal point; $\mathbb{E}\left[|||\boldsymbol{\Pi}_{\mathrm{R}}\boldsymbol{X}^k|||_{\mathrm{R}}^2\right]$ and $\mathbb{E}\left[|||\boldsymbol{\Pi}_{\mathrm{C}}\boldsymbol{Y}^k|||_{\mathrm{C}}^2\right]$ measure the consensus error and the gradient tracking error, respectively. To bound the compression errors, we use $\mathbb{E}\left[|||\boldsymbol{U}^k-\boldsymbol{X}^k|||_2^2\right]$ to measure the convergence of the momentums. The matrix norms $|||\cdot|||_{\mathrm{R}}$ and $|||\cdot|||_{\mathrm{C}}$ are defined in Lemma 3.

Compared with the convergence analysis for Push-Pull, the analysis for CPP additionally requires dealing with the compression errors and establishing the relationship between the error terms of different types. Moreover, another term $\mathbb{E}\left[|||\boldsymbol{Y}^k|||_2^2\right]$ is considered to simplify the proof, and its role will be made clear in the follow-up analysis.

Now, we have the following expansion by multiplying $\frac{\boldsymbol{s}_{\mathrm{R}}^{\top}}{n}$ on both sides of (5c)

$$\begin{aligned}\overline{\boldsymbol{x}}^{k+1}&=\frac{1}{n}\boldsymbol{s}_{\mathrm{R}}^{\top}\left(\boldsymbol{R}_{\beta}\boldsymbol{X}^k-\boldsymbol{\alpha}\boldsymbol{Y}^k-\beta\boldsymbol{R}\boldsymbol{W}^k\right)\\&=\overline{\boldsymbol{x}}^k-\frac{1}{n}\boldsymbol{s}_{\mathrm{R}}^{\top}\boldsymbol{\alpha}\left(\boldsymbol{Y}^k-\boldsymbol{s}_{\mathrm{C}}\overline{\boldsymbol{Y}}^k+\boldsymbol{s}_{\mathrm{C}}\overline{\boldsymbol{Y}}^k\right)-\frac{\beta}{n}\boldsymbol{s}_{\mathrm{R}}^{\top}\boldsymbol{W}^k\\&=\overline{\boldsymbol{x}}^k-\bar{\alpha}\overline{\boldsymbol{Y}}^k-\frac{1}{n}\boldsymbol{s}_{\mathrm{R}}^{\top}\boldsymbol{\alpha}\boldsymbol{\Pi}_{\mathrm{C}}\boldsymbol{Y}^k-\frac{\beta}{n}\boldsymbol{s}_{\mathrm{R}}^{\top}\boldsymbol{W}^k\\&=\overline{\boldsymbol{x}}^k-\bar{\alpha}\boldsymbol{g}^k+\bar{\alpha}\left(\boldsymbol{g}^k-\overline{\boldsymbol{Y}}^k\right)-\frac{1}{n}\boldsymbol{s}_{\mathrm{R}}^{\top}\boldsymbol{\alpha}\boldsymbol{\Pi}_{\mathrm{C}}\boldsymbol{Y}^k\\&\quad-\frac{\beta}{n}\boldsymbol{s}_{\mathrm{R}}^{\top}\boldsymbol{W}^k,\end{aligned}\tag{8}$$

where $\boldsymbol{g}^k=\nabla f(\overline{\boldsymbol{x}}^k)$.

Multiplying $\boldsymbol{\Pi}_{\mathrm{R}}$ on both sides of (5c), we have

$$\begin{aligned}&\boldsymbol{\Pi}_{\mathrm{R}}\boldsymbol{X}^{k+1}\\&=\boldsymbol{\Pi}_{\mathrm{R}}\boldsymbol{R}_{\beta}\boldsymbol{\Pi}_{\mathrm{R}}\boldsymbol{X}^k-\boldsymbol{\Pi}_{\mathrm{R}}\boldsymbol{\alpha}\boldsymbol{Y}^k-\beta\boldsymbol{\Pi}_{\mathrm{R}}\boldsymbol{R}\boldsymbol{W}^k,\end{aligned}\tag{9}$$

Multiplying $\boldsymbol{\Pi}_{\mathrm{C}}$ on both sides of (5d), we obtain

$$\begin{aligned}\boldsymbol{\Pi}_{\mathrm{C}}\boldsymbol{Y}^{k+1}&=\boldsymbol{\Pi}_{\mathrm{C}}\boldsymbol{C}_{\gamma}\boldsymbol{\Pi}_{\mathrm{C}}\boldsymbol{Y}^k+\gamma\left(\boldsymbol{I}-\boldsymbol{C}\right)\boldsymbol{E}^k\\&\quad+\boldsymbol{\Pi}_{\mathrm{C}}\left(\nabla\boldsymbol{F}\left(\boldsymbol{X}^{k+1}\right)-\nabla\boldsymbol{F}\left(\boldsymbol{X}^k\right)\right).\end{aligned}\tag{10}$$

Lemma 3: There are invertible matrices $\widetilde{\boldsymbol{R}},\widetilde{\boldsymbol{C}}\in\mathbb{R}^{n\times n}$ inducing vector norms $\|\boldsymbol{x}\|_{\mathrm{R}}\overset{\text{def}}{=}\|\widetilde{\boldsymbol{R}}\boldsymbol{x}\|_2,\|\boldsymbol{x}\|_{\mathrm{C}}\overset{\text{def}}{=}\|\widetilde{\boldsymbol{C}}\boldsymbol{x}\|_2$ for $\boldsymbol{x}\in\mathbb{R}^n$. The corresponding matrix norms $\|\cdot\|_{\mathrm{R}},\|\cdot\|_{\mathrm{C}}$ defined by Definition 2 satisfy: for any $\beta,\gamma\in[0,1]$,

$$\|\boldsymbol{\Pi}_{\mathrm{R}}\boldsymbol{R}_{\beta}\|_{\mathrm{R}}\leq 1-\theta_{\mathrm{R}}\beta,\ \|\boldsymbol{\Pi}_{\mathrm{C}}\boldsymbol{C}_{\gamma}\|_{\mathrm{C}}\leq 1-\theta_{\mathrm{C}}\gamma,\tag{11}$$

where $\theta_{\mathrm{R}},\theta_{\mathrm{C}}$ are constants in $(0,1]$. Especially, $\|\boldsymbol{\Pi}_{\mathrm{R}}\|_{\mathrm{R}}=\|\boldsymbol{\Pi}_{\mathrm{C}}\|_{\mathrm{C}}=1$ and $\|\boldsymbol{R}\|_{\mathrm{R}}\leq 2,\|\boldsymbol{R}-\boldsymbol{I}\|_{\mathrm{R}}\leq 3,\|\boldsymbol{C}\|_{\mathrm{C}}\leq 2,\|\boldsymbol{C}-\boldsymbol{I}\|_{\mathrm{C}}\leq 3$. Additionally, $\|\boldsymbol{x}\|_2\leq\|\boldsymbol{x}\|_{\mathrm{R}}$, $\|\boldsymbol{x}\|_2\leq\|\boldsymbol{x}\|_{\mathrm{C}}$, $\forall\boldsymbol{x}\in\mathbb{R}^n$; $|||\boldsymbol{A}|||_2\leq|||\boldsymbol{A}|||_{\mathrm{R}},|||\boldsymbol{A}|||_2\leq|||\boldsymbol{A}|||_{\mathrm{C}},\forall\boldsymbol{A}\in\mathbb{R}^{n\times p}$. There exist constants $\delta_{\mathrm{R},2},\delta_{\mathrm{C},2}$ such that $\|\boldsymbol{x}\|_{\mathrm{R}}\leq\delta_{\mathrm{R},2}\|\boldsymbol{x}\|_2,\|\boldsymbol{x}\|_{\mathrm{C}}\leq\delta_{\mathrm{C},2}\|\boldsymbol{x}\|_2,\forall\boldsymbol{x}\in\mathbb{R}^n$. And for any diagonal matrix $\boldsymbol{V}$, $\|\boldsymbol{V}\|_{\mathrm{R}}=\|\boldsymbol{V}\|_2,\|\boldsymbol{V}\|_{\mathrm{C}}=\|\boldsymbol{V}\|_2$.

Proof: See Lemma 3 of [1]. ■

In the following, $\|\cdot\|_{\mathrm{R}}$, $\|\cdot\|_{\mathrm{C}}$, $|||\cdot|||_{\mathrm{R}}$, and $|||\cdot|||_{\mathrm{C}}$ refer to the norms defined in the above lemma.

Lemma 4: For any vector norm $\|\cdot\|_*$ induced by the inner product $\langle\cdot,\cdot\rangle_*$, and for any $\theta>1$, we have

$$|||\boldsymbol{A}+\boldsymbol{B}|||_*^2\leq\theta|||\boldsymbol{A}|||_*^2+\frac{\theta}{\theta-1}|||\boldsymbol{B}|||_*^2,\ \forall\boldsymbol{A},\boldsymbol{B}\in\mathbb{R}^{n\times p}.$$

Especially, $\|\cdot\|_*$ can be taken as $\|\cdot\|_2$, $\|\cdot\|_{\mathrm{R}}$, $\|\cdot\|_{\mathrm{C}}$.

Proof: By Definition 1, it suffices to prove $\|\boldsymbol{a}+\boldsymbol{b}\|_*^2\leq\theta\|\boldsymbol{a}\|_*^2+\frac{\theta}{\theta-1}\|\boldsymbol{b}\|_*^2$ for any vectors $\boldsymbol{a},\boldsymbol{b}\in\mathbb{R}^n$. And it can be verified by the elementary inequality $2\langle\boldsymbol{a},\boldsymbol{b}\rangle_*\leq(\theta-1)\|\boldsymbol{a}\|_*^2+\frac{1}{\theta-1}\|\boldsymbol{b}\|_*^2$. The vector norms $\|\cdot\|_{\mathrm{R}},\|\cdot\|_{\mathrm{C}}$ are induced by inner products $\langle\widetilde{\boldsymbol{R}}\boldsymbol{x},\widetilde{\boldsymbol{R}}\boldsymbol{Y}\rangle_*,\langle\widetilde{\boldsymbol{C}}\boldsymbol{x},\widetilde{\boldsymbol{C}}\boldsymbol{Y}\rangle_*$, respectively. ■

The next lemma uses inequalities to derive recursive bounds for the quantities $\mathbb{E}\left[\|\overline{\boldsymbol{x}}^k-\boldsymbol{x}^*\|_2^2\right],\mathbb{E}\left[|||\boldsymbol{\Pi}_{\mathrm{R}}\boldsymbol{X}^k|||_{\mathrm{R}}^2\right],\mathbb{E}\left[|||\boldsymbol{\Pi}_{\mathrm{C}}\boldsymbol{Y}^k|||_{\mathrm{C}}^2\right]$ and $\mathbb{E}\left[|||\boldsymbol{U}^k-\boldsymbol{X}^k|||_2^2\right]$. The quantity $\mathbb{E}\left[|||\boldsymbol{Y}^k|||_2^2\right]$ is introduced to help simplify the proof.

Lemma 5: For $k\geq 0$, take $\boldsymbol{d}^k\in\mathbb{R}^5$ as

$$\begin{aligned}\boldsymbol{d}^k=\mathbb{E}\Big[&\left(\|\overline{\boldsymbol{x}}^k-\boldsymbol{x}^*\|_2^2,|||\boldsymbol{\Pi}_{\mathrm{R}}\boldsymbol{X}^k|||_{\mathrm{R}}^2,\right.\\&\left.|||\boldsymbol{\Pi}_{\mathrm{C}}\boldsymbol{Y}^k|||_{\mathrm{C}}^2,|||\boldsymbol{U}^k-\boldsymbol{X}^k|||_2^2,|||\boldsymbol{Y}^k|||_2^2\right)\Big]^{\top}.\end{aligned}$$

Define a parameterized matrix $\boldsymbol{A}(\widehat{\alpha},\beta,\gamma,\eta)$ as follows

$$\boldsymbol{A}_{:,1:3}=\begin{pmatrix}1-c_1\widehat{\alpha}&c_2\widehat{\alpha}&c_3\widehat{\alpha}\\0&1-\theta_{\mathrm{R}}\beta&0\\0&\frac{8nc_7\beta^2}{\gamma}&1-\theta_{\mathrm{C}}\gamma\\0&\frac{8n\beta^2}{\eta}&0\\c_{10}&c_9&3\end{pmatrix},$$

$$\boldsymbol{A}_{:,4:5}=\begin{pmatrix}c_4\beta^2&0\\c_6\beta^2&c_5\frac{\widehat{\alpha}^2}{\beta}\\\frac{C_2nc_7\beta^2}{\gamma}&c_8\gamma^2+\frac{2c_7\widehat{\alpha}^2}{\gamma}\\\boldsymbol{A}_{44}&\frac{2\widehat{\alpha}^2}{\eta}\\0&0\end{pmatrix},\tag{12}$$

where $\boldsymbol{A}_{44}=1-\eta+2C_2\eta^2+2nC_2\beta^2$ and c_1-c_{10} are constants given by (34), (36), (39) and (44) in the Supplementary Material of [1]. Then, under Assumptions 1-3 and the condition $\bar{\alpha}\leq\frac{2}{\mu+L}$, we have the following inequalities:

$$\begin{cases}\mathbb{E}\left[|||\boldsymbol{Y}^k|||_2^2\right]\leq\boldsymbol{A}\left(\widehat{\alpha},\beta,\gamma,\eta\right)_{5,1:4}\boldsymbol{d}_{1:4}^k,&\text{(13a)}\\\boldsymbol{d}_{1:4}^{k+1}\leq\boldsymbol{A}\left(\widehat{\alpha},\beta,\gamma,\eta\right)_{1:4,:}\boldsymbol{d}^k.&\text{(13b)}\end{cases}$$

Proof: See Lemma 5 of [1]. ■

Now, we proceed to show the spectral radius of $\boldsymbol{A}(\widehat{\alpha},\beta,\gamma,\eta)$ defined in (12) is less than 1 with properly chosen parameters.

Lemma 6: Given any $\alpha'>0,\beta'>0,w\geq\frac{n}{\boldsymbol{s}_{\mathrm{R}}^{\top}\boldsymbol{s}_{\mathrm{C}}}$ and $0<\eta<\frac{1}{2C_2}$ with $C_2>0$ given in Assumption 3, there exists $\gamma'>0$ such that for any $0<\gamma<\gamma'$, if we take $\widehat{\alpha}=\alpha'\gamma^3$ and $\beta=\beta'\gamma^2$, then

$$\rho\left(\boldsymbol{A}\left(\widehat{\alpha},\beta,\gamma,\eta\right)\right)<1.$$

More specifically, we can choose $\gamma' = \sup_{\{\widehat{\boldsymbol{v}}_i\}_{i=1}^5} \in \mathcal{P} \min\{\xi_1, \xi_2, \xi_3, \xi_4\}$, where $\mathcal{P} = \left\{ \{\widehat{\boldsymbol{v}}_i\}_{i=1}^5 | \{\widehat{\boldsymbol{v}}_i\}_{i=1}^5 satisfies\ (14a), (14b) \right\}$ and $\xi_1, \xi_2, \xi_3, \xi_4$ are the minimum positive roots of the polynomials (15a)–(15d).

Proof: To begin with, we choose positive numbers $\widehat{\boldsymbol{v}}_1, \widehat{\boldsymbol{v}}_2, \widehat{\boldsymbol{v}}_3, \widehat{\boldsymbol{v}}_4, \widehat{\boldsymbol{v}}_5$ satisfying

$$\begin{cases} c_2\widehat{\boldsymbol{v}}_2 + c_3\widehat{\boldsymbol{v}}_3 < c_1\widehat{\boldsymbol{v}}_1, & (14a) \\ c_{10}\widehat{\boldsymbol{v}}_1 + c_9\widehat{\boldsymbol{v}}_2 + 3\widehat{\boldsymbol{v}}_3 < \widehat{\boldsymbol{v}}_5 & (14b). \end{cases}$$

Define $\widehat{\boldsymbol{v}} = (\widehat{\boldsymbol{v}}_1, \widehat{\boldsymbol{v}}_2, \widehat{\boldsymbol{v}}_3, \widehat{\boldsymbol{v}}_4, \widehat{\boldsymbol{v}}_5)^\top$, then the five entries $\{g_i\}_{i=1}^5$ of the vector $[\boldsymbol{A}(\alpha'\gamma^3, \beta'\gamma^2, \gamma, \eta)\widehat{\boldsymbol{v}} - \widehat{\boldsymbol{v}}]$ (as functions of γ) are given by

$$\begin{aligned} g_1(\gamma) = & (-c_1\widehat{\boldsymbol{v}}_1 + c_2\widehat{\boldsymbol{v}}_2 + c_3\widehat{\boldsymbol{v}}_3)\alpha'\gamma^3 \\ & + c_4\widehat{\boldsymbol{v}}_4\beta'^2\gamma^4 \qquad (15a) \\ g_2(\gamma) = & -\theta_{\mathrm{R}}\beta'\widehat{\boldsymbol{v}}_2\gamma^2 \\ & + \left(c_6\widehat{\boldsymbol{v}}_4\beta'^2 + c_5\widehat{\boldsymbol{v}}_5\frac{\alpha'^2}{\beta'}\right)\gamma^4 \qquad (15b) \\ g_3(\gamma) = & -\theta_{\mathrm{C}}\widehat{\boldsymbol{v}}_3\gamma + c_8\widehat{\boldsymbol{v}}_5\gamma^2 \\ & + \left(8nc_7\widehat{\boldsymbol{v}}_2\beta'^2 + C_2nc_7\widehat{\boldsymbol{v}}_4\beta'^2\right)\gamma^3 \\ & + 2c_7\widehat{\boldsymbol{v}}_5\alpha'^2\gamma^5 \qquad (15c) \\ g_4(\gamma) = & -\left(\eta - 2C_2\eta^2\right)\widehat{\boldsymbol{v}}_4 \\ & + \left(\frac{8n\beta'^2\widehat{\boldsymbol{v}}_2}{\eta} + 2nC_2\beta'^2\widehat{\boldsymbol{v}}_4\right)\gamma^4 + \frac{2\alpha'^2\widehat{\boldsymbol{v}}_5}{\eta}\gamma^6 \qquad (15d) \\ g_5(\gamma) = & c_{10}\widehat{\boldsymbol{v}}_1 + c_9\widehat{\boldsymbol{v}}_2 + 3\widehat{\boldsymbol{v}}_3 - \widehat{\boldsymbol{v}}_5. \qquad (15e) \end{aligned}$$

By (14a), for any $0 < \gamma < \xi_1 = \frac{\alpha'(c_1\widehat{\boldsymbol{v}}_1 - c_2\widehat{\boldsymbol{v}}_2 - c_3\widehat{\boldsymbol{v}}_3)}{c_4\widehat{\boldsymbol{v}}_4\beta'^2}$, we have $g_1(\gamma) < 0$, where ξ_1 is the minimum positive root of $g_1(\gamma)$.

By direct calculation, $g_2(\gamma) < 0$, for any $0 < \gamma < \xi_2 = \sqrt{\frac{\theta_{\mathrm{R}}\beta'\widehat{\boldsymbol{v}}_2}{c_6\widehat{\boldsymbol{v}}_4\beta'^2 + c_5\widehat{\boldsymbol{v}}_5\frac{\alpha'^2}{\beta'}}}$, where ξ_2 is the minimum positive root of $g_2(\gamma)$.

Since $g_3(0) = 0$, $g_3'(0) = -\theta_{\mathrm{C}}\widehat{\boldsymbol{v}}_3 < 0$ and $\lim_{\gamma\to+\infty} g_3(\gamma) = +\infty$, $g_3(\gamma)$ has at least one positive root. By defining ξ_3 as this minimum positive root of $g_3(\gamma)$, we have $g_3(\gamma) < 0$, for any $0 < \gamma < \xi_3$.

As we have assumed $0 < \eta < \frac{1}{2C_2}$ and $\widehat{\boldsymbol{v}}_4 > 0$, $g_4(0) = -(\eta - 2C_2\eta^2)\widehat{\boldsymbol{v}}_4 < 0$. Since $\lim_{\gamma\to+\infty} g_4(\gamma) = +\infty$, $g_4(\gamma)$ has a minimum positive root ξ_4. Then, we have $g_4(\gamma) < 0$, for any $0 < \gamma < \xi_4$. From (14b), $g_5(\gamma)$ is a negative constant.

Thus, by the above discussion, for $0 < \gamma < \min\{\xi_1, \xi_2, \xi_3, \xi_4\}$, we have $g_i(\gamma) < 0 (1 \le i \le 5)$, i.e., $\boldsymbol{A}(\widehat{\alpha}'\gamma^3, \beta'\gamma^2, \gamma, \eta)\widehat{\boldsymbol{v}} = \boldsymbol{A}(\widehat{\alpha}, \beta, \gamma, \eta)\widehat{\boldsymbol{v}} < \widehat{\boldsymbol{v}}$. Since $\widehat{\boldsymbol{v}}$ is a positive vector, $\boldsymbol{A}(\widehat{\alpha}, \beta, \gamma, \eta)$ is a nonnegative matrix, by [65, Corollary 8.1.29], $\rho(\boldsymbol{A}(\widehat{\alpha}, \beta, \gamma, \eta)) < 1$. The above arguments hold true for any positive numbers $\{\widehat{\boldsymbol{v}}_i\}_{i=1}^5$ satisfying (14a) and (14b). Then, by defining $\mathcal{P}$ as the set containing these positive number sets $\{\widehat{\boldsymbol{v}}_i\}_{i=1}^5$, we can choose γ' as $\sup_{\{\widehat{\boldsymbol{v}}_i\}_{i=1}^5 \in \mathcal{P}} \min\{\xi_1, \xi_2, \xi_3, \xi_4\}$. ■

The next theorem shows that Algorithm 1 converges linearly with proper parameters and stepsizes.

Theorem 7: Under Assumptions 1-3, given positive numbers α', β' and $0 < \eta < \frac{1}{2C_2}$, $\eta \le 1$, $w \ge \frac{n}{\boldsymbol{s}_{\mathrm{R}}^\top \boldsymbol{s}_{\mathrm{C}}}$, there exists $\gamma' > 0$ (the value of γ' is given in Lemma 6) such that for any γ satisfying $0 < \gamma < \gamma'$ and

$$\gamma \le \min\left\{1, \beta'^{-1/2}, \left(\frac{2n}{\alpha'(\mu + L)\left(\boldsymbol{s}_{\mathrm{R}}^\top \boldsymbol{s}_{\mathrm{C}}\right)}\right)^{1/3}\right\}, \quad (16)$$

if we set $\beta = \beta'\gamma^2$ and the stepsizes $\{\alpha_i\}_{i\in\mathcal{N}}$ satisfy $\widehat{\alpha} = \max_{i\in\mathcal{N}} \alpha_i = \alpha'\gamma^3$, with $\widehat{\alpha}/\bar{\alpha} \le w$, then $\mathbb{E}[\|\overline{\boldsymbol{x}}^k - \boldsymbol{x}^*\|_2^2]$, $\mathbb{E}[|||\boldsymbol{\Pi}_{\mathrm{R}}\boldsymbol{X}^k|||_{\mathrm{R}}^2]$ converge linearly to 0. More specifically,

$$\mathbb{E}\left[\left\|\overline{\boldsymbol{x}}^k - \boldsymbol{x}^*\right\|_2^2\right] \le \sigma\overline{\boldsymbol{v}}_1\rho\left(\boldsymbol{A}\left(\widehat{\alpha}, \beta, \gamma, \eta\right)\right)^k,$$

$$\mathbb{E}\left[\left|\left|\left|\boldsymbol{\Pi}_{\mathrm{R}}\boldsymbol{X}^k\right|\right|\right|_{\mathrm{R}}^2\right] \le \sigma\overline{\boldsymbol{v}}_2\rho\left(\boldsymbol{A}\left(\widehat{\alpha}, \beta, \gamma, \eta\right)\right)^k,$$

where $\sigma, \overline{\boldsymbol{v}}_1, \overline{\boldsymbol{v}}_2$ are constants given in the proof.

Proof: Denote $\boldsymbol{A} = \boldsymbol{A}(\widehat{\alpha}, \beta, \gamma, \eta)$ for simplicity. Since $\boldsymbol{A}$ is a regular nonnegative matrix,[1] by the Perron-Frobenius theorem [66], $\rho(\boldsymbol{A})$ is an eigenvalue of $\boldsymbol{A}$, and $\boldsymbol{A}$ has a unique right positive eigenvector $\overline{\boldsymbol{v}}$ with respect to the eigenvalue $\rho(\boldsymbol{A})$. Define the constant $\sigma = \max_{1\le i\le 4} \frac{\boldsymbol{d}_i^0}{\overline{\boldsymbol{v}}_i}$. Thus, by the definition of σ, $\boldsymbol{d}_{1:4}^0 \le \sigma\overline{\boldsymbol{v}}_{1:4}$.

Next, we prove the linear convergence by induction. If we have proved $\boldsymbol{d}_{1:4}^k \le \sigma\rho(\boldsymbol{A})^k\overline{\boldsymbol{v}}_{1:4}$, we will show that it also holds true for $k+1$. The requirement (16) guarantees that $\gamma \le 1$, $\beta = \beta'\gamma^2 \le 1$, $\bar{\alpha} \le \frac{(\boldsymbol{s}_{\mathrm{R}}^\top \boldsymbol{s}_{\mathrm{C}})\widehat{\alpha}}{n} = \frac{(\boldsymbol{s}_{\mathrm{R}}^\top \boldsymbol{s}_{\mathrm{C}})\alpha'\gamma^3}{n} \le \frac{2}{\mu+L}$. According to Lemma 6, $\rho(\boldsymbol{A}) < 1$.

By (13a) and the inductive hypothesis, there holds

$$\mathbb{E}\left[\left|\left|\left|\boldsymbol{Y}^k\right|\right|\right|_2^2\right] \le \boldsymbol{A}_{5,1:4}\boldsymbol{d}_{1:4}^k \le \sigma\rho\left(\boldsymbol{A}\right)^k\boldsymbol{A}_{5,1:4}\overline{\boldsymbol{v}}_{1:4}$$

$$= \sigma\rho\left(\boldsymbol{A}\right)^k\boldsymbol{A}_{5,:}\overline{\boldsymbol{v}} = \sigma\rho\left(\boldsymbol{A}\right)^{k+1}\overline{\boldsymbol{v}}_5 \le \sigma\rho\left(\boldsymbol{A}\right)^k\overline{\boldsymbol{v}}_5.$$

Combining with the inductive hypothesis, we obtain

$$\boldsymbol{d}^k \le \sigma\rho\left(\boldsymbol{A}\right)^k\overline{\boldsymbol{v}}. \quad (17)$$

Now, by (13b) and (17),

$$\boldsymbol{d}_{1:4}^{k+1} \le \boldsymbol{A}_{1:4,:}\boldsymbol{d}^k \le \sigma\rho\left(\boldsymbol{A}\right)^k\boldsymbol{A}_{1:4,:}\overline{\boldsymbol{v}} = \sigma\rho\left(\boldsymbol{A}\right)^{k+1}\overline{\boldsymbol{v}}_{1:4},$$

i.e., the statement holds for $k+1$. Therefore, by induction, $\boldsymbol{d}_{1:4}^k \le \sigma\rho(\boldsymbol{A})^k\overline{\boldsymbol{v}}_{1:4}^k$, for any $k \ge 0$, which completes the proof. ■

Remark 8: In practice, we can take for instance $\alpha' = \frac{1}{L}, \beta' = 1, \gamma = 1, \eta = \min\{\frac{1}{2C_2}, 1\}$. Then, we hand-tune γ under the relations $\alpha_i = \widehat{\alpha} = \alpha'\gamma^3 (\forall i \in \mathcal{N})$ and $\beta = \beta'\gamma^2$ to achieve the optimal performance.

Remark 9: We can also add momentums for the communication of $\boldsymbol{Y}^k$ like what we did for $\boldsymbol{X}^k$. In this case, linear convergence can be proved similarly. However, we see little improvement in the numerical experiments when $\boldsymbol{Y}^k$ is equipped with momentums. In addition, more momentums will take more storage space. Therefore, we omit the details of this case here.

[1] A matrix $\boldsymbol{A}$ is said to be regular if all the entries of $\boldsymbol{A}^k$ are positive for some integer $k \ge 0$.

IV. A Broadcast-Like Gradient Tracking Method With Compression (B-CPP)

In this section, we consider a broadcast-like gradient tracking method with compression (B-CPP). Broadcast or gossip-based communication protocols are popular for distributed computation due to their low communication costs [62]–[64]. In B-CPP, at every iteration k, one agent $i_k \in \mathcal{N}$ wakes up with uniform probability. This can be easily realized, for example, if each agent wakes up according to an independent Poisson clock with the same parameter. Hence the probability $\mathbf{Pr}[i_k = j] = \frac{1}{n}$ for any $j \in \mathcal{N}$. In addition, $\{i_k\}_{k\geq 0}$ are independent with each other.

Briefly speaking, each iteration k of the B-CPP algorithm consists of the following steps:

1) One random agent i_k wakes up.
2) Agent i_k sends information to all its out-neighbors in $\boldsymbol{R}$ ($\mathcal{N}^+_{\mathrm{R},i_k}$) and $\boldsymbol{C}$ ($\mathcal{N}^+_{\mathrm{C},i_k}$).
3) The agents who received information from i_k are awaken and update their local variables.

We remark that the number k is only used for analysis purpose and does not need to be known by the agents. B-CPP can be generalized to the case when each agent wakes up with different but known probabilities. The awakened agents will know whether they are i_k and whether they are in the set $\mathcal{N}^+_{\mathrm{R},i_k}$ or in the set $\mathcal{N}^+_{\mathrm{C},i_k}$, and will take different actions accordingly. The B-CPP method is illustrated in Algorithm 2. To implement communication compression in a broadcast setting, a naive way is to let each agent i hold different momentums $\boldsymbol{u}_{i,j}$ for each neighbors j. In this way, at each time agent i receives information $\boldsymbol{p}$ from neighboring agent j, it can restore the information sent from agent j directly by summing $\boldsymbol{p} + \boldsymbol{u}_{i,j}$. However, this procedure would require momentums as many as the total number of edges. Hence it could be impractical when the storage space is limited and the graph is dense. In B-CPP, we overcome this issue so that each agent only uses 2 momentums.

To help analyze the convergence rate of Algorithm 2, for $0 \leq k \leq K$, let us define the final state of variable $\boldsymbol{x}_j$ before iteration k as $\boldsymbol{x}_j^k$. These $\boldsymbol{x}_j^k$ are written compactly into an n-by-p matrix $\boldsymbol{X}^k$. And $\boldsymbol{Y}^k, \boldsymbol{U}^k, \boldsymbol{U}^k_{\mathrm{R}}, \nabla \boldsymbol{F}(\boldsymbol{X}^k)$ are defined analogously. Let $\mathcal{D}_k$ be the σ-field generated by $\{i_j, \boldsymbol{p}^j, \boldsymbol{q}^j\}_{0\leq j\leq k-1}$. Let $\mathcal{D}_k^+$ denote the σ-field generated by $\mathcal{D}_k$ and i_k. Let $\mathcal{D}_k^{++}$ denote the σ-field generated by $\mathcal{D}_k$ and $\{i_k, \boldsymbol{p}^k\}$. For any event A, let χ_A denote the indicator function of A.

The differences between B-CPP and CPP mainly lie in the averaging step. Taking the averaging step for $\boldsymbol{X}^k$ as an example. Firstly, we also have by induction that $\boldsymbol{U}^k_{\mathrm{R}} = \boldsymbol{R}\boldsymbol{U}^k,\ \forall 0 \leq k \leq K$. For $i \in \mathcal{N}$, denote the in-degree of i in $\boldsymbol{R}$ by $r_i = |\mathcal{N}^-_{\mathrm{R},i}|$. Since $\boldsymbol{R}$ is row stochastic, $r_i > 0$ for any $i \in \mathcal{N}$. Notice that the update step (18a) will be implemented by agent j when agent i_k is an in-neighbor of agent j (or equivalently, j is an out-neighbor of i_k).

$$\begin{aligned}
&\mathbb{E}\left[\chi_{\left\{j\in\mathcal{N}^+_{\mathrm{R},i_k}\right\}} n\boldsymbol{R}_{j,i_k}\boldsymbol{p}^k|\mathcal{D}_k\right]\\
&= \mathbb{E}\left[\chi_{\left\{j\in\mathcal{N}^+_{\mathrm{R},i_k}\right\}} n\boldsymbol{R}_{j,i_k}\mathbb{E}\left[\boldsymbol{p}^k|i_k\right]|\mathcal{D}_k\right]\\
&= \mathbb{E}\left[n\boldsymbol{R}_{j,i_k}\left(\boldsymbol{x}^k_{i_k} - \boldsymbol{u}^k_{i_k}\right)|\mathcal{D}_k\right]\\
&= \frac{1}{n}\sum_{i_k=1}^{n} n\boldsymbol{R}_{j,i_k}\left(\boldsymbol{x}^k_{i_k} - \boldsymbol{u}^k_{i_k}\right)\\
&= \boldsymbol{R}_{j,:}\left(\boldsymbol{X}^k - \boldsymbol{U}^k\right).
\end{aligned}$$

Analogously,

$$\begin{aligned}
&\mathbb{E}\left[\chi_{\left\{j\in\mathcal{N}^+_{\mathrm{R},i_k}\right\}}\left(1-\frac{\beta n}{r_j}\right)\boldsymbol{x}_j^k|\mathcal{D}_k\right]\\
&= \frac{1}{n}\sum_{i_k\in\mathcal{N},\ \boldsymbol{R}_{j,i_k}>0}\left(1-\frac{\beta n}{r_j}\right)\boldsymbol{x}_j^k\\
&= \frac{r_j}{n}\boldsymbol{x}_j^k - \beta\boldsymbol{x}_j^k,
\end{aligned}$$

Algorithm 2: B-CPP.

Input: each agent i is instilled its initial position $\boldsymbol{x}_i$, parameters $\beta, \gamma, \eta \in (0, 1]$, stepsize α_i and network information $\boldsymbol{R}_{ij} (j \in \mathcal{N}^-_{\mathrm{R},i})$, $\boldsymbol{C}_{ij} (j \in \mathcal{N}^-_{\mathrm{C},i})$ and the sets $\mathcal{N}^+_{\mathrm{R},i}$, $\mathcal{N}^+_{\mathrm{C},i}$

Output: each agent i outputs its final local decision variable $\boldsymbol{x}_i$

Each agent i initializes $\boldsymbol{y}_i^0 = \nabla f_i\left(\boldsymbol{x}_i^0\right), \boldsymbol{u}_i^0 = \boldsymbol{0}, \boldsymbol{u}^0_{\mathrm{R},i} = \boldsymbol{0}$.

for $k = 0$ *to* $K-1$ **do**

Agent i_k awakes.

Agent i_k compresses $\boldsymbol{p}^k = \textsc{Compress}\left(\boldsymbol{x}_{i_k} - \boldsymbol{u}_{i_k}\right)$, sends $\boldsymbol{p}^k$ to all agents $j \in \mathcal{N}^+_{\mathrm{R},i_k}$ and wakes up these agents.

Agent i_k compresses $\boldsymbol{q}^k = \textsc{Compress}\left(\boldsymbol{y}_{i_k}\right)$, sends $\boldsymbol{q}^k$ to all agents in $\mathcal{N}^+_{\mathrm{C},i_k}$ and wakes up these agents.

For all agents $j \in \{i_k\} \cup \mathcal{N}^+_{\mathrm{R},i_k} \cup \mathcal{N}^+_{\mathrm{C},i_k}$, agent j records $\boldsymbol{d}_j \leftarrow \nabla f_j\left(\boldsymbol{x}_j\right)$.

For $j \in \mathcal{N}^+_{\mathrm{R},i_k}$, agent j updates

$$\boldsymbol{u}_{\mathrm{R},j} \leftarrow \boldsymbol{u}_{\mathrm{R},j} + \eta n\boldsymbol{R}_{j,i_k}\boldsymbol{p}^k, \tag{18a}$$

$$\boldsymbol{x}_j \leftarrow \left(1-\frac{\beta n}{r_j}\right)\boldsymbol{x}_j + \frac{\beta n}{r_j}\boldsymbol{u}_{\mathrm{R},j} + \beta n\boldsymbol{R}_{j,i_k}\boldsymbol{p}^k. \tag{18b}$$

Agent i_k updates $\boldsymbol{u}_{i_k} \leftarrow \boldsymbol{u}_{i_k} + \eta n\boldsymbol{p}^k$.

For all $j \in \{i_k\} \cup \mathcal{N}^+_{\mathrm{R},i_k} \cup \mathcal{N}^+_{\mathrm{C},i_k}$, agent j updates $\boldsymbol{x}_j \leftarrow \boldsymbol{x}_j - \alpha_j\boldsymbol{y}_j$.

For all $j \in \{i_k\} \cup \mathcal{N}^+_{\mathrm{R},i_k} \cup \mathcal{N}^+_{\mathrm{C},i_k}$, agent j updates $\boldsymbol{y}_j \leftarrow \boldsymbol{y}_j + \nabla f_j\left(\boldsymbol{x}_j\right) - \boldsymbol{d}_j$.

Agent i_k update $\boldsymbol{y}_{i_k} \leftarrow \boldsymbol{y}_{i_k} - \gamma n\boldsymbol{q}^k$.

For $j \in \mathcal{N}^+_{\mathrm{C},i_k}$, agent j updates $\boldsymbol{y}_j \leftarrow \boldsymbol{y}_j + \gamma n\boldsymbol{C}_{ji_k}\boldsymbol{q}^k$.

end

Return the final local decision variable $\{\boldsymbol{x}_i\}_{i\in\mathcal{N}}$.

and

$$\begin{aligned}&\mathbb{E}\left[\chi_{\left\{j\in\mathcal{N}^+_{\mathrm{R},i_k}\right\}}\left(\frac{n}{r_j}\boldsymbol{u}^k_{\mathrm{R},j}\right)|\mathcal{D}_k\right]\\&=\frac{n}{r_j}\boldsymbol{u}^k_{\mathrm{R},j}\mathbf{Pr}\left[j\in\mathcal{N}^+_{\mathrm{R},i_k}|\mathcal{D}_k\right]\\&=\frac{1}{n}\sum_{i_k\in\mathcal{N},\,\boldsymbol{R}_{j,i_k}>0}\frac{n}{r_j}\boldsymbol{u}^k_{\mathrm{R},j}\\&=\boldsymbol{u}^k_{\mathrm{R},j}=\boldsymbol{R}_{j,:}\boldsymbol{U}^k.\end{aligned}$$

Thus, taking expectation on the RHS of (18b) yields

$$\begin{aligned}&\mathbb{E}\left[\chi_{\left\{j\notin\mathcal{N}^+_{\mathrm{R},i_k}\right\}}\boldsymbol{x}^k_j+\chi_{\left\{j\in\mathcal{N}^+_{\mathrm{R},i_k}\right\}}\left(\left(1-\frac{\beta n}{r_j}\right)\boldsymbol{x}^k_j\right)\right.\\&\left.+\chi_{\left\{j\in\mathcal{N}^+_{\mathrm{R},i_k}\right\}}\left(\frac{\beta n}{r_j}\boldsymbol{u}^k_{\mathrm{R},j}+\beta n\boldsymbol{R}_{j,i_k}\boldsymbol{p}^k\right)\Big|\mathcal{D}_k\right]\\&=\left(\frac{n-r_j}{n}+\frac{r_j}{n}-\beta\right)\boldsymbol{x}^k_j+\beta\boldsymbol{R}_{j,:}\left(\boldsymbol{U}^k+\boldsymbol{X}^k-\boldsymbol{U}^k\right)\\&=\left[\boldsymbol{R}_\beta\right]_{j,:}\boldsymbol{X}^k.\end{aligned}$$

Briefly speaking, after taking conditional expectation, the averaging step of B-CPP reduces to that of CPP (the first term of the RHS of (5c)). By choosing a proper value for β, the variance of the RHS of (18) can be made small enough. In this way, the averaging step in B-CPP could have a similar effect to that of CPP.

To show the linear convergence of B-CPP, for simplicity, we assume $\alpha=\alpha_1=\alpha_2=\cdots=\alpha_n$. We remark that the analysis below can be easily generalized to the case when the stepsizes differ among the agents.

We denote $\mathbf{1}_i\in\mathbb{R}^{n\times1}$ as the vector with 1 on the i-th component and 0 on the others. Define

$$\begin{aligned}\boldsymbol{P}^k&=\boldsymbol{X}^k-\boldsymbol{U}^k+\mathbf{1}_{i_k}\left(\boldsymbol{p}^k-\boldsymbol{x}^k_{i_k}+\boldsymbol{u}^k_{i_k}\right),\\\boldsymbol{Q}^k&=\boldsymbol{Y}^k+\mathbf{1}_{i_k}\left(\boldsymbol{q}^k-\boldsymbol{Y}^k_{i_k}\right).\end{aligned}$$

As $\boldsymbol{U}^k+\boldsymbol{P}^k$ and $\boldsymbol{Q}^k$ are used to estimate $\boldsymbol{X}^k$ and $\boldsymbol{y}^k$, respectively, we define the error matrices induced by the compression as follows

$$\begin{aligned}\boldsymbol{W}^k&=\boldsymbol{X}^k-\boldsymbol{U}^k-\boldsymbol{P}^k=\mathbf{1}_{i_k}\left(\boldsymbol{x}^k_{i_k}-\boldsymbol{u}^k_{i_k}-\boldsymbol{p}^k\right),\\\boldsymbol{E}^k&=\boldsymbol{Y}^k-\boldsymbol{Q}^k=\mathbf{1}_{i_k}\left(\boldsymbol{Y}^k_{i_k}-\boldsymbol{q}^k\right).\end{aligned}$$

The following lemma is a direct corollary of Assumption 3.

Lemma 10: The random variable i_k is independent with $\mathcal{D}_k$. And $\mathbb{E}[\boldsymbol{W}^k|\mathcal{D}^+_k]=\mathbf{0}$, $\mathbb{E}[\boldsymbol{E}^k|\mathcal{D}^{++}_k]=\mathbf{0}$.

Moreover, by Assumption 3,

$$\begin{aligned}&\mathbb{E}\left[|||\boldsymbol{W}^k|||_2^2|\mathcal{D}^+_k\right]=\mathbb{E}\left[\left\|\boldsymbol{x}^k_{i_k}-\boldsymbol{u}^k_{i_k}-\boldsymbol{p}^k\right\|_2^2|\mathcal{D}^+_k\right]\\&\le C_2\mathbb{E}\left[\left\|\boldsymbol{x}^k_{i_k}-\boldsymbol{u}^k_{i_k}\right\|_2^2|\mathcal{D}_k\right]=\frac{C_2}{n}|||\boldsymbol{X}^k-\boldsymbol{U}^k|||_2^2,\quad(19)\\&\mathbb{E}\left[|||\boldsymbol{E}^k|||_2^2|\mathcal{D}^{++}_k\right]=\mathbb{E}\left[\left\|\boldsymbol{Y}_{i_k}-\boldsymbol{q}^k\right\|_2^2|\mathcal{D}_k\right]\\&\le C_2\mathbb{E}\left[\left\|\boldsymbol{Y}^k_{i_k}\right\|_2^2|\mathcal{D}^{++}_k\right]=\frac{C_2}{n}|||\boldsymbol{Y}^k|||_2^2.\quad(20)\end{aligned}$$

We will also use the following random matrices, which are all measurable with respect to i_k,

$$\begin{aligned}&\Lambda^k=n\mathbf{Diag}\left(\mathbf{1}_{i_k}\right),\widetilde{\Lambda}^k=\sum_{j\in\{i_k\}\cup\mathcal{N}^+_{\mathrm{R},i_k}\cup\mathcal{N}^+_{\mathrm{C},i_k}}\mathbf{Diag}\left(\mathbf{1}_j\right),\\&\widetilde{\Lambda}^k_{\mathrm{R}}=\sum_{j\in\mathcal{N}^+_{\mathrm{R},i_k}}\frac{n}{r_j}\mathbf{Diag}\left(\mathbf{1}_j\right),\overline{\boldsymbol{R}}^k=\sum_{j\in\mathcal{N}^+_{\mathrm{R},i_k}}\frac{n}{r_j}\mathbf{1}_j\boldsymbol{R}_{j,:},\\&\widehat{\boldsymbol{R}}^k=n\boldsymbol{R}_{:,i_k}\mathbf{1}^\top_{i_k},\widehat{\boldsymbol{C}}^k=n\boldsymbol{C}_{:,i_k}\mathbf{1}^\top_{i_k}.\end{aligned}$$

Now, Algorithm 2 can be rewritten into a more compact form:

$$\begin{cases}\boldsymbol{X}^{k+1}=\left(\boldsymbol{I}-\beta\widetilde{\Lambda}^k_{\mathrm{R}}\right)\boldsymbol{X}^k+\beta\widetilde{\Lambda}^k_{\mathrm{R}}\boldsymbol{R}\boldsymbol{U}^k\\\qquad\qquad+\beta\widehat{\boldsymbol{R}}^k\boldsymbol{P}^k-\alpha\widetilde{\Lambda}^k\boldsymbol{Y}^k,&(21a)\\\boldsymbol{Y}^{k+1}=\boldsymbol{Y}^k+\gamma\left(\widehat{\boldsymbol{C}}^k-\Lambda^k\right)\boldsymbol{Q}^k\\\qquad\qquad+\nabla\boldsymbol{F}(\boldsymbol{X}^{k+1})-\nabla\boldsymbol{F}(\boldsymbol{X}^k),&(21b)\\\boldsymbol{U}^{k+1}=\boldsymbol{U}^k+\eta\Lambda^k\boldsymbol{P}^k.&(21c)\end{cases}$$

Compared to CPP, the stochastic matrices $\widehat{\boldsymbol{R}}^k,\overline{\boldsymbol{R}}^k,\Lambda^k,\widetilde{\Lambda}^k,\widetilde{\Lambda}^k_{\mathrm{R}}$ will induce additional errors that need to be dealt with in the convergence analysis of B-CPP.

To analyze the convergence rate of B-CPP, similar to what we did for CPP, we also use $\mathbb{E}[\|\overline{\boldsymbol{x}}^k-\boldsymbol{x}^*\|_2^2]$, $\mathbb{E}[|||\boldsymbol{\Pi}_{\mathrm{R}}\boldsymbol{X}^k|||^2_{\mathrm{R}}],\mathbb{E}[|||\boldsymbol{\Pi}_{\mathrm{C}}\boldsymbol{Y}^k|||^2_{\mathrm{C}}]$ and $\mathbb{E}[|||\boldsymbol{U}^k-\boldsymbol{X}^k|||_2^2]$ to measure the closeness to the optimal point, consensus error, gradient tracking error and the convergence of the momentums, respectively. And we use $\mathbb{E}[|||\boldsymbol{Y}^k|||_2^2],\mathbb{E}[|||\boldsymbol{X}^{k+1}-\boldsymbol{X}^k|||_2^2]$ and $\mathbb{E}[|||\mathcal{V}^k|||_2^2]$ to help simplify the recursive relations between the above four quantities. The definition of $\mathcal{V}^k$ is given by (47) in the Supplementary Material of [1].

Lemma 11: For $k\ge0$, define $\widetilde{\boldsymbol{d}}^k\in\mathbb{R}^7$ as an extension of $\boldsymbol{d}^k$:

$$\widetilde{\boldsymbol{d}}^k=\left(\boldsymbol{d}^k,\mathbb{E}\left[|||\mathcal{V}^k|||_2^2\right],\mathbb{E}\left[|||\boldsymbol{X}^{k+1}-\boldsymbol{X}^k|||_2^2\right]\right)^\top.$$

Define a parameterized matrix $\widetilde{\boldsymbol{A}}(\alpha,\beta,\gamma,\eta)$ as follows

$$\widetilde{\boldsymbol{A}}_{:,1:3}=\begin{pmatrix}1-d_1\alpha&d_2\alpha&d_3\alpha\\0&1-\theta_{\mathrm{R}}\beta&0\\0&0&1-\theta_{\mathrm{C}}\gamma\\0&0&0\\c_{10}&c_9&3\\0&3V_1\beta^2&0\\0&18\beta^2&0\end{pmatrix},$$

$$\widetilde{\boldsymbol{A}}_{:,4:7}=\begin{pmatrix}0&0&d_4&0\\0&\frac{\alpha^2}{\beta\theta_{\mathrm{R}}}&\delta^2_{\mathrm{R},2}&0\\0&\frac{d_5\gamma^2}{1-\theta_{\mathrm{C}}\gamma}+d_7\gamma^2&0&\frac{d_6}{\gamma}\\\widetilde{\boldsymbol{A}}_{44}&0&0&\frac{2}{\eta}\\0&0&0&0\\d_9\beta^2&3V_4\alpha^2&0&0\\0&2\alpha^2&1&0\end{pmatrix}\quad(22)$$

where $\widetilde{\boldsymbol{A}}_{44}=1-\eta+\frac{\eta^2(n-1)^2}{1-\eta}+d_8\frac{\beta^2}{\eta}+2C_2n\eta^2+d_8\beta^2$ and the constants d_1-d_9 are given by (51), (55), (57) and (59) in the Supplementary Material of [1]. Then, under Assumptions 1-3

and the condition $\widetilde{\alpha} \leq \frac{2}{\mu+L}$, we have the following inequalities:

$$\begin{cases} \mathbb{E}\left[\left|\left|\left|\boldsymbol{Y}^k\right|\right|\right|_2^2\right] \leq \widetilde{\boldsymbol{A}}\left(\alpha,\beta,\gamma,\eta\right)_{5,1:4}\widetilde{\boldsymbol{d}}_{1:4}^k, & \text{(23a)} \\ \mathbb{E}\left[\left|\left|\left|\mathcal{V}^k\right|\right|\right|_2^2\right] \leq \widetilde{\boldsymbol{A}}\left(\alpha,\beta,\gamma,\eta\right)_{6,1:5}\widetilde{\boldsymbol{d}}_{1:5}^k, & \text{(23b)} \\ \mathbb{E}\left[\left|\left|\left|\boldsymbol{X}^{k+1}-\boldsymbol{X}^k\right|\right|\right|_2^2\right] \leq \widetilde{\boldsymbol{A}}\left(\alpha,\beta,\gamma,\eta\right)_{7,1:6}\widetilde{\boldsymbol{d}}_{1:6}^k, & \text{(23c)} \\ \widetilde{\boldsymbol{d}}_{1:4}^{k+1} \leq \widetilde{\boldsymbol{A}}\left(\alpha,\beta,\gamma,\eta\right)_{1:4,:}\widetilde{\boldsymbol{d}}^k. & \text{(23d)} \end{cases}$$

Proof: See Lemma 11 of [1]. ■

Next, we show that with properly chosen parameters and stepsize, the spectral radius of the parameterized matrix $\widetilde{\boldsymbol{A}}(\alpha,\beta,\gamma,\eta)$ in (22) will be less than 1.

Lemma 12: Given any positive numbers $\alpha,'\beta'$ and η satisfying

$$0<\eta\leq\frac{1}{2},\quad \eta<\frac{1}{2(n-1)^2+2C_2n}, \tag{24}$$

there exists $\gamma'>0$ such that for any $0<\gamma<\gamma'$, if we take $\alpha=\alpha'\gamma^3$ and $\beta=\beta'\gamma^2$, then

$$\rho\left(\widetilde{\boldsymbol{A}}\left(\alpha,\beta,\gamma,\eta\right)\right)<1.$$

More specifically, we can take $\gamma'=\sup_{\{\widetilde{\boldsymbol{v}}_i\}_{i=1}^7\in\mathcal{P}'}\min\{\psi_2,\psi_3,\psi_4,\psi_6,\psi_7,\frac{1}{\theta_{\mathrm{C}}}\}$, where θ_{C} is given in Lemma 3 and $\mathcal{P},'\psi_i(i=2,3,4,6,7)$ are given in the proof.

Proof: Given positive numbers $\{\widetilde{\boldsymbol{v}}_i\}_{i=1}^7$ satisfying

$$\begin{cases} d_2\widetilde{\boldsymbol{v}}_2\alpha'+d_3\widetilde{\boldsymbol{v}}_3\alpha'+d_4\widetilde{\boldsymbol{v}}_6<d_1\widetilde{\boldsymbol{v}}_1\alpha,' & \text{(25a)} \\ c_{10}\widetilde{\boldsymbol{v}}_1+c_9\widetilde{\boldsymbol{v}}_2+3\widetilde{\boldsymbol{v}}_3<\widetilde{\boldsymbol{v}}_5, & \text{(25b)} \\ \widetilde{\boldsymbol{v}}_6<\widetilde{\boldsymbol{v}}_7\ . & \text{(25c)} \end{cases}$$

Define $\widehat{\boldsymbol{v}}=(\widetilde{\boldsymbol{v}}_1,\widetilde{\boldsymbol{v}}_2,\widetilde{\boldsymbol{v}}_3,\widetilde{\boldsymbol{v}}_4,\widetilde{\boldsymbol{v}}_5,\gamma^3\widetilde{\boldsymbol{v}}_6,\gamma^3\widetilde{\boldsymbol{v}}_7)^\top$. The seven entries $\{g_i\}_{i=1}^7$ of the vector $[\widehat{\boldsymbol{v}}-\widetilde{\boldsymbol{A}}(\alpha'\gamma^3,\beta'\gamma^2,\gamma,\eta)\widehat{\boldsymbol{v}}]$ (as functions of γ) are given by

$$\begin{aligned} g_1(\gamma) &= \gamma^3\left(-d_1\widetilde{\boldsymbol{v}}_1\alpha'+d_2\widetilde{\boldsymbol{v}}_2\alpha'+d_3\widetilde{\boldsymbol{v}}_3\alpha'+d_4\widetilde{\boldsymbol{v}}_6\right) \\ g_2(\gamma) &= \gamma^2\left(-\theta_{\mathrm{R}}\beta'\widetilde{\boldsymbol{v}}_2+\frac{\alpha'^2\widetilde{\boldsymbol{v}}_5}{\beta'\theta_{\mathrm{R}}}\gamma^2+\delta_{\mathrm{R},2}^2\widetilde{\boldsymbol{v}}_6\gamma\right), \\ g_3(\gamma) &= \gamma\left(-\theta_{\mathrm{C}}\widetilde{\boldsymbol{v}}_3+\frac{d_5\widetilde{\boldsymbol{v}}_5\gamma}{1-\theta_{\mathrm{C}}\gamma}+d_7\widetilde{\boldsymbol{v}}_5\gamma+d_6\widetilde{\boldsymbol{v}}_7\gamma\right), \\ g_4(\gamma) &= \left(-\eta+\frac{\eta^2(n-1)^2}{1-\eta}+2C_2n\eta^2\right)\widetilde{\boldsymbol{v}}_4+d_8\widetilde{\boldsymbol{v}}_4\frac{\beta'^2}{\eta}\gamma^4 \\ &\quad +d_8\widetilde{\boldsymbol{v}}_4\beta'^2\gamma^4+\frac{2\widetilde{\boldsymbol{v}}_7}{\eta}\gamma^3, \\ g_5(\gamma) &= c_{10}\widetilde{\boldsymbol{v}}_1+c_9\widetilde{\boldsymbol{v}}_2+3\widetilde{\boldsymbol{v}}_3-\widetilde{\boldsymbol{v}}_5, \\ g_6(\gamma) &= \gamma^3\left(3V_1\widetilde{\boldsymbol{v}}_2\beta'^2\gamma+d_9\widetilde{\boldsymbol{v}}_4\beta'^2\gamma+3V_4\widetilde{\boldsymbol{v}}_5\alpha'^2\gamma^3-\widetilde{\boldsymbol{v}}_6\right), \\ g_7(\gamma) &= \gamma^3\left((\widetilde{\boldsymbol{v}}_6-\widetilde{\boldsymbol{v}}_7)+18\widetilde{\boldsymbol{v}}_2\beta'^2\gamma+2\widetilde{\boldsymbol{v}}_5\alpha'^2\gamma^3\right). \end{aligned}$$

Denote $\widetilde{g}_i=g_i/\gamma^3(i=1,6,7)$, $\widetilde{g}_2=g_2/\gamma^2$, $\widetilde{g}_3=g_3/\gamma$. By (25a), $\widetilde{g}_1=-d_1\widetilde{\boldsymbol{v}}_1\alpha'+d_2\widetilde{\boldsymbol{v}}_2\alpha'+d_3\widetilde{\boldsymbol{v}}_3\alpha'+d_4\widetilde{\boldsymbol{v}}_6<0$. By (25b), $g_5=c_{10}\widetilde{\boldsymbol{v}}_1+c_9\widetilde{\boldsymbol{v}}_2+3\widetilde{\boldsymbol{v}}_3-\widetilde{\boldsymbol{v}}_5<0$. By (25c), $\widetilde{g}_7(0)=\widetilde{\boldsymbol{v}}_6-\widetilde{\boldsymbol{v}}_7<0$. By (24), $\eta\leq1/2$, then $g_4(0)\leq-\eta+2\eta^2(n-1)^2+2C_2n\eta^2<0$. It follows directly that $\widetilde{g}_2(0)=-\theta_{\mathrm{R}}\beta'\widetilde{\boldsymbol{v}}_2<0$, $\widetilde{g}_3(0)=-\theta_{\mathrm{C}}\widetilde{\boldsymbol{v}}_3<0$, $\widetilde{g}_6(0)=-\widetilde{\boldsymbol{v}}_6<0$.

If $0<\gamma<\frac{1}{\theta_{\mathrm{C}}}$, we have $\lim_{\gamma\to+\infty}g_3(\gamma)=+\infty$. Combining with the facts that $\lim_{\gamma\to+\infty}g_i(\gamma)=+\infty$, for $i=2,4,6,7$, we can define ψ_i to be the minimum positive solution of $g_i(\gamma)$ $(i=2,3,4,6,7)$ when $0<\gamma<\frac{1}{\theta_{\mathrm{C}}}$.

Thus, for any $0<\gamma<\min\{\psi_2,\psi_3,\psi_4,\psi_6,\psi_7,\frac{1}{\theta_{\mathrm{C}}}\}$, there holds $g_i(\gamma)<0$, for any $1\leq i\leq7$, i.e., $\widetilde{\boldsymbol{A}}(\alpha,\beta,\gamma,\eta)\widehat{\boldsymbol{v}}=\widetilde{\boldsymbol{A}}(\alpha'\gamma^3,\beta'\gamma^2,\gamma,\eta)\widehat{\boldsymbol{v}}<\widehat{\boldsymbol{v}}$. Since $\widetilde{\boldsymbol{A}}(\alpha,\beta,\gamma,\eta)$ is a nonnegative matrix, $\widehat{\boldsymbol{v}}$ is a positive vector, by [65, Corollary 8.1.29], we have $\rho(\widetilde{\boldsymbol{A}}(\alpha,\beta,\gamma,\eta))<1$.

The above arguments hold true for arbitrary $\{\widetilde{\boldsymbol{v}}_i\}_{i=1}^7$ satisfying (25a), (25b) and (25c). Then, let the set $\mathcal{P}'$ consist of all these $\{\widetilde{\boldsymbol{v}}_i\}_{i=1}^7$, we have $\gamma'\geq\sup_{\{\widetilde{\boldsymbol{v}}_i\}_{i=1}^7\in\mathcal{P}'}\min\{\psi_2,\psi_3,\psi_4,\psi_6,\psi_7,\frac{1}{\theta_{\mathrm{C}}}\}$.

■

The following theorem shows the linear convergence of Algorithm 2 given proper parameters and stepsize.

Theorem 13: Under Assumptions 1-3, if we choose arbitrary positive numbers $\alpha,'\beta,'\eta$ satisfying $0<\eta\leq\frac{1}{2}$, $\eta<\frac{1}{2(n-1)^2+2C_2n}$, and choose $\gamma'>0$ as in Lemma 12. Then, for any γ satisfying $0<\gamma<\gamma'$ and

$$\gamma\leq\min\left\{1,\beta'^{-\frac{1}{2}},\left(\frac{2n}{\alpha'\left(\mu+L\right)\left(\boldsymbol{s}_{\mathrm{R}}^\top\widetilde{\Lambda}\boldsymbol{s}_{\mathrm{C}}\right)}\right)^{\frac{1}{3}}\right\}, \tag{26}$$

$\beta=\beta'\gamma^2$, and $\alpha=\alpha'\gamma^3$, there holds, for any $k\geq0$,

$$\begin{aligned} \mathbb{E}\left[\left\|\overline{\boldsymbol{x}}^k-\boldsymbol{x}^*\right\|_2^2\right] &\leq \sigma'\overline{\boldsymbol{v}}_1'\rho\left(\widetilde{\boldsymbol{A}}\left(\alpha,\beta,\gamma,\eta\right)\right)^k, \\ \mathbb{E}\left[\left|\left|\left|\boldsymbol{\Pi}_{\mathrm{R}}\boldsymbol{X}^k\right|\right|\right|_{\mathrm{R}}^2\right] &\leq \sigma'\overline{\boldsymbol{v}}_2'\rho\left(\widetilde{\boldsymbol{A}}\left(\alpha,\beta,\gamma,\eta\right)\right)^k, \end{aligned}$$

where $\sigma',\overline{\boldsymbol{v}}_1',\overline{\boldsymbol{v}}_2'$ are constants given in the proof.

Proof: To begin with, we denote $\widetilde{\boldsymbol{A}}=\widetilde{\boldsymbol{A}}(\alpha,\beta,\gamma,\eta)$ for simplicity. Since $\widetilde{\boldsymbol{A}}$ is a regular nonnegative matrix, by the Perron-Frobenius theorem [66], $\rho(\widetilde{\boldsymbol{A}})$ is an eigenvalue of $\widetilde{\boldsymbol{A}}$, and $\widetilde{\boldsymbol{A}}$ has a unique positive right eigenvector $\overline{\boldsymbol{v}}'$ with respect to the eigenvalue $\rho(\widetilde{\boldsymbol{A}})$. Define $\sigma'=\max_{1\leq i\leq4}\frac{\widetilde{\boldsymbol{d}}_i^0}{\overline{\boldsymbol{v}}_i'}$. Then, $\widetilde{\boldsymbol{d}}_{1:4}^0\leq\sigma'\overline{\boldsymbol{v}}_{1:4}'$.

Next, we prove the linear convergence by induction. If we have proved $\widetilde{\boldsymbol{d}}_{1:4}^k\leq\sigma'\rho(\widetilde{\boldsymbol{A}})^k\overline{\boldsymbol{v}}_{1:4}'$, we will show that it also holds for $k+1$. The requirement (26) guarantees that $\gamma\leq1$, $\beta\leq1$, $\widetilde{\alpha}\leq\frac{2}{\mu+L}$. From Lemma 12, $\rho(\widetilde{\boldsymbol{A}})<1$.

By (23a) and the inductive hypothesis, there holds

$$\begin{aligned} &\mathbb{E}\left[\left|\left|\left|\boldsymbol{Y}^k\right|\right|\right|_2^2\right] \leq \widetilde{\boldsymbol{A}}_{5,1:4}\widetilde{\boldsymbol{d}}_{1:4}^k \leq \sigma'\rho\left(\widetilde{\boldsymbol{A}}\right)^k\widetilde{\boldsymbol{A}}_{5,1:4}\overline{\boldsymbol{v}}_{1:4}' \\ &=\sigma'\rho\left(\widetilde{\boldsymbol{A}}\right)^k\left[\widetilde{\boldsymbol{A}}\overline{\boldsymbol{v}}'\right]_5=\sigma'\rho\left(\widetilde{\boldsymbol{A}}\right)^{k+1}\overline{\boldsymbol{v}}_5'\leq\sigma'\rho\left(\widetilde{\boldsymbol{A}}\right)^k\overline{\boldsymbol{v}}_5'. \end{aligned}$$

Then combining with the inductive hypothesis, we have

$$\widetilde{\boldsymbol{d}}_{1:5}^k\leq\sigma'\rho\left(\widetilde{\boldsymbol{A}}\right)^k\overline{\boldsymbol{v}}_{1:5}'. \tag{27}$$

By (23b) and (27),

$$\begin{aligned} &\mathbb{E}\left[\left|\left|\left|\mathcal{V}^k\right|\right|\right|_2^2\right] \leq \widetilde{\boldsymbol{A}}_{6,1:5}\widetilde{\boldsymbol{d}}_{1:5}^k \leq \sigma'\rho\left(\widetilde{\boldsymbol{A}}\right)^k\widetilde{\boldsymbol{A}}_{6,1:5}\overline{\boldsymbol{v}}_{1:5}' \\ &=\sigma'\rho\left(\widetilde{\boldsymbol{A}}\right)^k\left[\widetilde{\boldsymbol{A}}\overline{\boldsymbol{v}}'\right]_6=\sigma'\rho\left(\widetilde{\boldsymbol{A}}\right)^{k+1}\overline{\boldsymbol{v}}_6'\leq\sigma'\rho\left(\widetilde{\boldsymbol{A}}\right)^k\overline{\boldsymbol{v}}_6'. \end{aligned}$$

Then combining with (27), we have

$$\widetilde{\boldsymbol{d}}_{1:6}^k\leq\sigma'\rho\left(\widetilde{\boldsymbol{A}}\right)^k\overline{\boldsymbol{v}}_{1:6}'. \tag{28}$$

By (23c) and (28),

$$\mathbb{E}\left[\left|\left|\left|\boldsymbol{X}^{k+1}-\boldsymbol{X}^{k}\right|\right|\right|_2^2\right] \leq \widetilde{\boldsymbol{A}}_{7,1:6}\widetilde{\boldsymbol{d}}^{k}_{1:6} \leq \sigma'\rho\left(\widetilde{\boldsymbol{A}}\right)^{k}\widetilde{\boldsymbol{A}}_{7,1:6}\overline{\boldsymbol{v}}'_{1:6}$$

$$= \sigma'\rho\left(\widetilde{\boldsymbol{A}}\right)^{k}\left[\widetilde{\boldsymbol{A}}\overline{\boldsymbol{v}}'\right]_7 = \sigma'\rho\left(\widetilde{\boldsymbol{A}}\right)^{k+1}\overline{\boldsymbol{v}}'_7 \leq \sigma'\rho\left(\widetilde{\boldsymbol{A}}\right)^{k}\overline{\boldsymbol{v}}'_7.$$

Then combining with (28), we have obtained

$$\widetilde{\boldsymbol{d}}^{k} \leq \sigma'\rho\left(\widetilde{\boldsymbol{A}}\right)^{k}\overline{\boldsymbol{v}}'. \tag{29}$$

Using (23d) and (29), we have

$$\widetilde{\boldsymbol{d}}^{k+1}_{1:4} \leq \widetilde{\boldsymbol{A}}_{1:4,:}\widetilde{\boldsymbol{d}}^{k} \leq \sigma'\rho\left(\widetilde{\boldsymbol{A}}\right)^{k}\widetilde{\boldsymbol{A}}_{1:4,:}\overline{\boldsymbol{v}}'$$

$$= \sigma'\rho\left(\widetilde{\boldsymbol{A}}\right)^{k}\left[\widetilde{\boldsymbol{A}}\overline{\boldsymbol{v}}'\right]_{1:4} = \sigma'\rho\left(\widetilde{\boldsymbol{A}}\right)^{k+1}\overline{\boldsymbol{v}}'_{1:4},$$

i.e., the statement holds for $k+1$. Therefore, by induction, $\widetilde{\boldsymbol{d}}^{k}_{1:4} \leq \sigma'\rho(\widetilde{\boldsymbol{A}})^{k}\overline{\boldsymbol{v}}'_{1:4}$, for any $k \geq 0$, which completes the proof. ■

Remark 14: In practice, the parameters β, γ, η and stepsize α can be chosen in the same way as in CPP.

V. Numerical Experiments

In this section, we compare the numerical performance of CPP and B-CPP with the Push-Pull/$\mathcal{AB}$ method [25], [26]. In the experiments, we equip CPP and B-CPP with different compression operators and consider different graph topologies.

We consider the following decentralized ℓ_2-regularized logistic regression problem:

$$\min_{\boldsymbol{x}\in\mathbb{R}^p} \frac{1}{n}\sum_{i=1}^{n}\log\left(1+e^{-\lambda_i \boldsymbol{z}_i^{\top}\boldsymbol{x}}\right) + \frac{\mu}{2}\|\boldsymbol{x}\|_2^2,$$

where $(\boldsymbol{z}_i, \lambda_i) \in \mathbb{R}^p \times \{-1, +1\}$ is the training example possessed by agent i. The data is from QSAR biodegradation Data Set[2] [67], where each feature vector is of $p = 41$ dimension. In our experiments, we set $n = 20$ and $\mu = 0.001$. We construct the directed graphs $\mathcal{G}_{\boldsymbol{R}}$ and $\mathcal{G}_{\boldsymbol{C}}$ by adding d directed links to the undirected cycle, respectively. In our experiments below, we will consider the cases $d = 5, 20, 50$, representing sparse, normal and dense graphs respectively.

We consider two kinds of compression operators. The first one is the b bit 2-quantization given by

$$\text{COMPRESS}\left(\boldsymbol{x}\right) = \left(\|\boldsymbol{x}\|_2 \operatorname{sign}\left(\boldsymbol{x}\right)2^{1-b}\right)\left\lfloor \frac{2^{b-1}\left|\boldsymbol{x}\right|}{\|\boldsymbol{x}\|_2} + \boldsymbol{v}\right\rfloor,$$

where $\boldsymbol{v} \in \mathbb{R}^p$ is a random vector picked uniformly from $(0,1)^p$ and we choose $b = 2, 4, 6$ in the experiments. The second one is Rand-k which is defined by

$$\text{COMPRESS}\left(\boldsymbol{x}\right) = \frac{p}{k}\widehat{\boldsymbol{x}},$$

where $\widehat{\boldsymbol{x}}$ is obtained by randomly choosing k entries of $\boldsymbol{x}$ and setting the rest entries to be 0. We will set $k = 5, 10, 20$ respectively. Then, we hand-tune to find the optimal parameters for all the numerical cases below. In the figures, the y-axis represents the loss $f(\overline{\boldsymbol{x}}^i) - f(\boldsymbol{x}^*)$ and x-axis is the iteration number or the number of transmitted bits.

[2]https://archive.ics.uci.edu/ml/datasets/QSAR+biodegradation

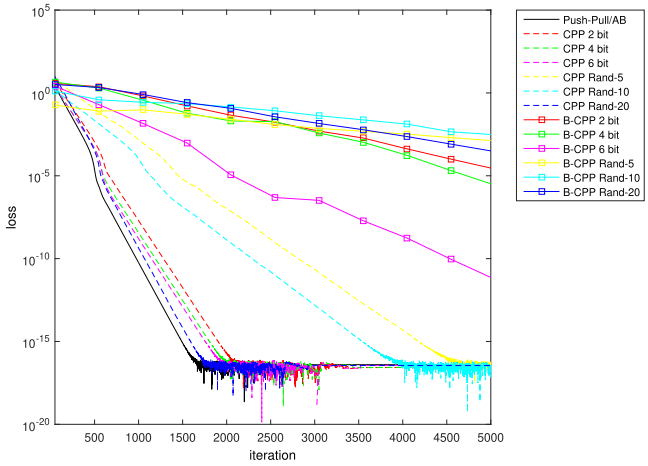


Fig. 1. Linear convergence of Push-Pull/$\mathcal{AB}$, CPP, and B-CPP with b bit quantization ($b = 2, 4, 6$) and Rand-k ($k = 5, 10, 20$) compressors.

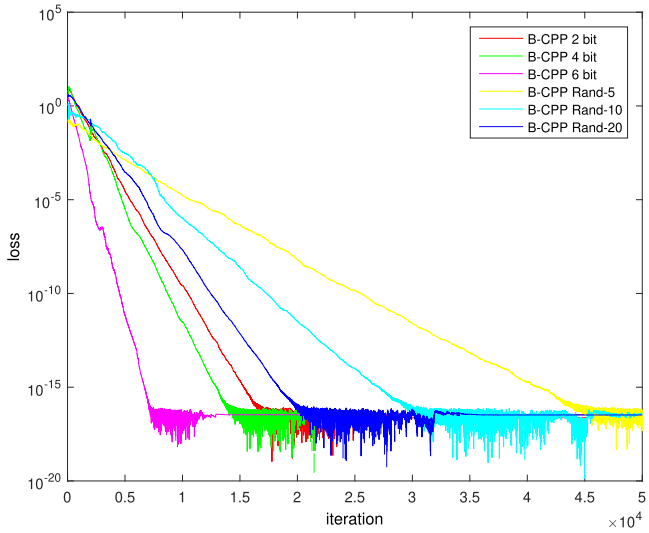


Fig. 2. Linear convergence of B-CPP with b bit quantization ($b = 2, 4, 6$) and Rand-k ($k = 5, 10, 20$) compressors.

A. Linear Convergence

The performance of Push-Pull/$\mathcal{AB}$, CPP and B-CPP is illustrated in Fig. 1. To see the performance of B-CPP more clearly, we plot the trajectories of B-CPP additionally at a larger scale in Fig. 2. The experiments illustrated in Fig. 1 and Fig. 2 are conducted on graphs with $d = 20$. The quantization with $b = 2, 4, 6$ and Rand-k with $k = 5, 10, 20$ are considered. It can be seen from Fig. 1 and Fig. 2 that all the trajectories exhibit linear convergence, and the exact Push-Pull/$\mathcal{AB}$ method is faster than CPP and B-CPP with any compression methods. This is reasonable as the compression operator induces additional errors compared to the exact method, and these additional errors could slow down the convergence. Meanwhile, as the values of b or k increases, both CPP and B-CPP speed up since the compression errors decrease. When $b = 6$ or $k = 20$, the trajectories of CPP are very close to that of exact Push-Pull/$\mathcal{AB}$, which indicates that when the compression errors are small, they are no longer the bottleneck of convergence.

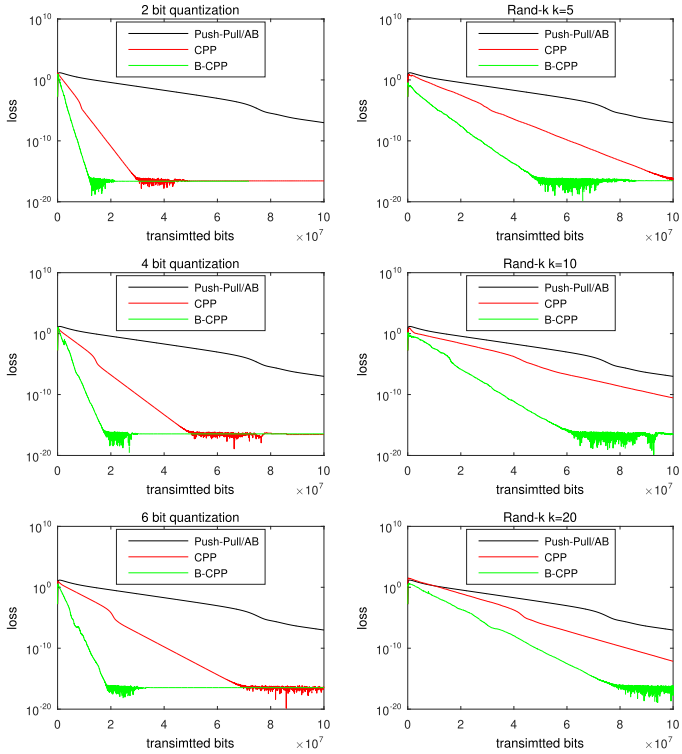


Fig. 3. Performance of Push-Pull/$\mathcal{AB}$, CPP, B-CPP against the number of transmitted bits: the left column shows the results with quantization ($b = 2, 4, 6$) and the right column shows the results with Rand-k ($k = 5, 10, 20$).

Within the same number of iterations, CPP outperforms B-CPP, and the trajectories of CPP are smoother than B-CPP. These results can be expected since CPP updates all the local variables in a single iteration, while in B-CPP, only one node updates with its neighbors. To guarantee the linear convergence, B-CPP requires smaller parameters and stepsizes. In addition, the mixing matrix at each iteration of B-CPP can be regarded as a stochastic matrix, which will induce additional variances.

B. Communication Efficiency

When we compare the number of transmitted bits to reach certain levels of accuracy, CPP and B-CPP show their superiority. We consider the same settings as in Section V-A. We can see from all of the sub-figures of Fig. 3 that, to reach a high accuracy within about 10^{-15}, the number of transmitted bits required by these methods have the ranking: B-CPP $<$ CPP $<$ Push-Pull/$\mathcal{AB}$.

To see why CPP outperforms Push-Pull/$\mathcal{AB}$, note that the vectors sent in CPP have been compressed, and hence the transmitted bits at each iteration are greatly reduced compared to Push-Pull/$\mathcal{AB}$. Although the additional compression errors slow down the convergence, our design for CPP guarantees that the impact on the convergence rate is relatively small. Therefore, CPP is much more communication-efficient than the exact Push-Pull/$\mathcal{AB}$ method. Moreover, in all cases, B-CPP is much more communication-efficient than CPP. This is because when CPP converges, each agent will receive similar, or "overlapping" vectors from different neighbors, which impacts the communication-efficiency. By contrast, for B-CPP, only one agent wakes up in each iteration, and its information can be utilized by its neighbors more efficiently.

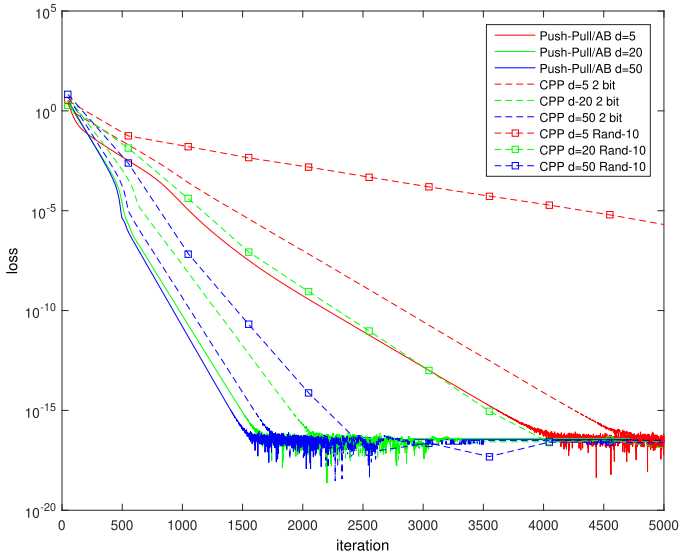


Fig. 4. Performance of CPP and Push-Pull/$\mathcal{AB}$ with different communication networks under both quantization and Rand-k compressors.

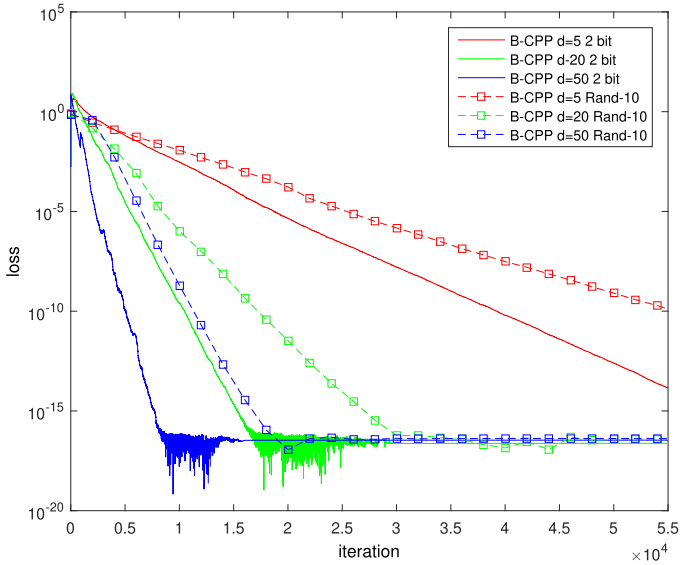


Fig. 5. Performance of B-CPP with different communication networks under both quantization and Rand-k compressors.

It is worth noting that for both CPP and B-CPP, the choices $b = 2$ for quantization or $k = 5$ for Rand-k are more communication-efficient than $b = 4, 6$ or $k = 10, 20$. This indicates that as the compression accuracy becomes smaller, its impact exhibits "marginal effects". In other words, when the compression errors are not the bottleneck for the convergence, sacrificing the communication costs for faster convergence will reduce the communication efficiency.

C. Different Topologies

We also examine the performance of CPP and B-CPP on communication networks with different levels of connectivity.

We consider $d = 5, 10, 20$ respectively: as the value of d increases, the communication network has a better connectivity. As before, both the quantization and the Rand-k compressors are considered, and we show the performance of CPP and B-CPP separately for better clarity.

In Fig. 4 and Fig. 5, we can see that when d increases, the convergence of both CPP/B-CPP and Push-Pull/$\mathcal{AB}$ speed up. This is expected since better connectivity implies less consensus errors in each iteration, and the algorithms perform closer to the centralized gradient descent algorithm which is faster.

VI. Conclusion

In this paper, we proposed two communication-efficient algorithms for decentralized optimization over a multi-agent network with general directed topology. First, we consider a novel communication-efficient gradient tracking based method, termed CPP, that combines the Push-Pull method with communication compression. CPP can be applied to a general class of unbiased compression operators and achieves linear convergence for strongly convex and smooth objective functions. Second, we consider a broadcast-like version of CPP (B-CPP) which also achieves linear convergence rate for strongly convex and smooth objective functions. B-CPP can be applied in an asynchronous broadcast setting and further reduce communication costs compared to CPP.

Acknowledgment

Most work was done while the first author was visiting the School of Data Science, The Chinese University of Hong Kong, Shenzhen.

References

[1] Z. Song, L. Shi, S. Pu, and M. Yan, "Compressed gradient tracking for decentralized optimization over general directed networks," 2021, *arXiv:2106.07243*.

[2] K. Cohen, A. Nedić, and R. Srikant, "On projected stochastic gradient descent algorithm with weighted averaging for least squares regression," *IEEE Trans. Autom. Control*, vol. 62, no. 11, pp. 5974–5981, Nov. 2017.

[3] A. I. Forrester, A. Sóbester, and A. J. Keane, "Multi-fidelity optimization via surrogate modelling," in *Proc. Royal Soc. London A: Math. Physical Eng. Sci.*, 2007, pp. 3251–3269.

[4] A. Nedić, A. Olshevsky, and C. A. Uribe, "Fast convergence rates for distributed non-Bayesian learning," *IEEE Trans. Autom. Control*, vol. 62, no. 11, pp. 5538–5553, Nov. 2017.

[5] J. Chen and A. H. Sayed, "Diffusion adaptation strategies for distributed optimization and learning over networks," *IEEE Trans. Signal Process.*, vol. 60, no. 8, pp. 4289–4305, Aug. 2012.

[6] S. Pu, A. Garcia, and Z. Lin, "Noise reduction by swarming in social foraging," *IEEE Trans. Autom. Control*, vol. 61, no. 12, pp. 4007–4013, Dec. 2016.

[7] B. Baingana, G. Mateos, and G. B. Giannakis, "Proximal-gradient algorithms for tracking cascades over social networks," *IEEE J. Sel. Topics Signal Process.*, vol. 8, no. 4, pp. 563–575, Aug. 2014.

[8] K. Cohen, A. Nedić, and R. Srikant, "Distributed learning algorithms for spectrum sharing in spatial random access wireless networks," *IEEE Trans. Autom. Control*, vol. 62, no. 6, pp. 2854–2869, Jun. 2017.

[9] G. Mateos and G. B. Giannakis, "Distributed recursive least-squares: Stability and performance analysis," *IEEE Trans. Signal Process.*, vol. 60, no. 7, pp. 3740–3754, Jul. 2012.

[10] A. Nedić and J. Liu, "Distributed optimization for control," *Annu. Rev. Control, Robot., Auton. Syst.*, vol. 1, pp. 77–103, 2018.

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

[12] Z. Li, W. Shi, and M. Yan, "A decentralized proximal-gradient method with network independent step-sizes and separated convergence rates," *IEEE Trans. Signal Process.*, vol. 67, no. 17, pp. 4494–4506, Sep. 2019.

[13] G. Qu and N. Li, "Harnessing smoothness to accelerate distributed optimization," *IEEE Trans. Control Netw. Syst.*, vol. 5, no. 3, pp. 1245–1260, Sep. 2018.

[14] K. Scaman, F. Bach, S. Bubeck, L. Massoulié, and Y. T. Lee, "Optimal algorithms for non-smooth distributed optimization in networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 2745–2754.

[15] W. Shi, Q. Ling, G. Wu, and W. Yin, "EXTRA: An exact first-order algorithm for decentralized consensus optimization," *SIAM J. Optim.*, vol. 25, no. 2, pp. 944–966, 2015.

[16] C. A. Uribe, S. Lee, A. Gasnikov, A. Gasnikov, and A. Nedić, "A dual approach for optimal algorithms in distributed optimization over networks," *Optim. Methods Softw.*, vol. 36, no. 1, pp. 171–210, 2021, doi: 10.1080/10556788.2020.1750013.

[17] 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*, 2015, pp. 2055–2060.

[18] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information," in *Proc. 44th Annu. IEEE Symp. Foundations Comput. Sci.*, 2003, pp. 482–491.

[19] A. Nedić and A. Olshevsky, "Distributed optimization over time-varying directed graphs," *IEEE Trans. Autom. Control*, vol. 60, no. 3, pp. 601–615, Mar. 2015.

[20] C. Xi and U. A. Khan, "DEXTRA: A fast algorithm for optimization over directed graphs," *IEEE Trans. Autom. Control*, vol. 62, no. 10, pp. 4980–4993, Oct. 2017.

[21] J. Zeng and W. Yin, "ExtraPush for convex smooth decentralized optimization over directed networks," *J. Comput. Math.*, vol. 35, no. 4, pp. 383–396, 2017.

[22] A. Nedic, A. Olshevsky, and W. Shi, "Achieving geometric convergence for distributed optimization over time-varying graphs," *SIAM J. Optim.*, vol. 27, no. 4, pp. 2597–2633, 2017.

[23] Y. Tian, Y. Sun, and G. Scutari, "ASY-SONATA: Achieving linear convergence in distributed asynchronous multiagent optimization," in *Proc. 56th Annu. Allerton Conf. Commun., Control, Comput.*, 2018, pp. 543–551.

[24] C. Xi, R. Xin, and U. A. Khan, "ADD-OPT: Accelerated distributed directed optimization," *IEEE Trans. Autom. Control*, vol. 63, no. 5, pp. 1329–1339, May 2018.

[25] S. Pu, W. Shi, J. Xu, and A. Nedic, "Push-pull gradient methods for distributed optimization in networks," *IEEE Trans. Autom. Control*, vol. 66, no. 1, pp. 1–16, Jan. 2021.

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

[27] F. Saadatniaki, R. Xin, and U. A. Khan, "Decentralized optimization over time-varying directed graphs with row and column-stochastic matrices," *IEEE Trans. Autom. Control*, vol. 65, no. 11, pp. 4769–4780, Nov. 2020.

[28] J. Zhang and K. You, "Fully asynchronous distributed optimization with linear convergence in directed networks," 2021, *arXiv:1901.08215*.

[29] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "QSGD: Communication-efficient SGD via gradient quantization and encoding," *Adv. Neural Inf. Process. Syst.*, pp. 1709–1720, 2017.

[30] J. Bernstein, Y.-X. Wang, K. Azizzadenesheli, and A. Anandkumar, "signSGD: Compressed optimisation for non-convex problems," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 560–569.

[31] A. Beznosikov, S. Horváth, P. Richtárik, and M. Safaryan, "On biased compression for distributed learning," 2020, *arXiv:2002.12410*.

[32] S. P. Karimireddy, Q. Rebjock, S. Stich, and M. Jaggi, "Error feedback fixes signsgd and other gradient compression schemes," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 3252–3261.

[33] J. Liu *et al.*, "Distributed learning systems with first-order methods," *Foundations Trends Databases*, vol. 9, no. 1, pp. 1–100, 2020.

[34] X. Liu, Y. Li, J. Tang, and M. Yan, "A double residual compression algorithm for efficient distributed learning," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2020, pp. 133–143.

[35] K. Mishchenko, E. Gorbunov, M. Takáč, and P. Richtárik, "Distributed learning with compressed gradient differences," 2019, *arXiv:1901.09269*.

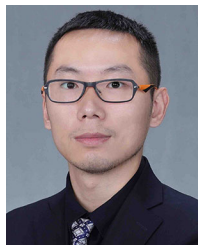
[36] F. Seide, H. Fu, J. Droppo, G. Li, and D. Yu, "1-bit stochastic gradient descent and its application to data-parallel distributed training of speech DNNs," in *Proc. 15th Annu. Conf. Int. Speech Commun. Assoc.*, 2014, pp. 1058–1062.

[37] S. U. Stich, "On communication compression for distributed optimization on heterogeneous data," 2020, *arXiv:2009.02388*.

[38] S. U. Stich, J.-B. Cordonnier, and M. Jaggi, "Sparsified SGD with memory," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 4447–4458.
[39] H. Tang, C. Yu, X. Lian, T. Zhang, and J. Liu, "Doublesqueeze: Parallel stochastic gradient descent with double-pass error-compensated compression," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 6155–6165.
[40] H. Xu *et al.*, "Compressed communication for distributed deep learning: Survey and quantitative evaluation," King Abdullah University of Science and Technology, Tech. Rep., 2020.
[41] Y. Lin, S. Han, H. Mao, Y. Wang, and B. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," in *Proc. Int. Conf. Learn. Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=SkhQHMW0W
[42] L. Xiao, S. Boyd, and S. Lall, "A scheme for robust distributed sensor fusion based on average consensus," in *Proc. 4th Int. Symp. Inf. Process. Sensor Netw.*, 2005, pp. 63–70.
[43] R. Carli, F. Fagnani, P. Frasca, T. Taylor, and S. Zampieri, "Average consensus on networks with transmission noise or quantization," in *Proc. Eur. Control Conf.*, 2007, pp. 1852–1857.
[44] A. Nedic, A. Olshevsky, A. Ozdaglar, and J. N. Tsitsiklis, "Distributed subgradient methods and quantization effects," in *Proc. 47th IEEE Conf. Decis. Control*, 2008, pp. 4177–4184.
[45] T. C. Aysal, M. J. Coates, and M. G. Rabbat, "Distributed average consensus with dithered quantization," *IEEE Trans. Signal Process.*, vol. 56, no. 10, pp. 4905–4918, Oct. 2008.
[46] R. Carli, F. Fagnani, P. Frasca, and S. Zampieri, "Gossip consensus algorithms via quantized communication," *Automatica*, vol. 46, no. 1, pp. 70–80, 2010.
[47] D. Yuan, S. Xu, H. Zhao, and L. Rong, "Distributed dual averaging method for multi-agent optimization with quantized communication," *Syst. Control Lett.*, vol. 61, no. 11, pp. 1053–1061, 2012.
[48] A. Reisizadeh, A. Mokhtari, H. Hassani, and R. Pedarsani, "An exact quantized decentralized gradient descent algorithm," *IEEE Trans. Signal Process.*, vol. 67, no. 19, pp. 4934–4947, Oct. 2019.
[49] R. Carli, F. Bullo, and S. Zampieri, "Quantized average consensus via dynamic coding/decoding schemes," *Int. J. Robust Nonlinear Control: IFAC-Affiliated J.*, vol. 20, no. 2, pp. 156–175, 2010.
[50] T. T. Doan, S. T. Maguluri, and J. Romberg, "Fast convergence rates of distributed subgradient methods with adaptive quantization," *IEEE Trans. Autom. Control*, vol. 66, no. 5, pp. 2191–2205, 2020.
[51] A. S. Berahas, C. Iakovidou, and E. Wei, "Nested distributed gradient methods with adaptive quantized communication," in *Proc. IEEE 58th Conf. Decis. Control*, 2019, pp. 1519–1525.
[52] Z. Li, D. Kovalev, X. Qian, and P. Richtárik, "Acceleration for compressed gradient descent in distributed and federated optimization," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 5895–5904.
[53] X. Liu, Y. Li, R. Wang, J. Tang, and M. Yan, "Linear convergent decentralized optimization with compression," in *Proc. Int. Conf. Learn. Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=84gjULz1t5
[54] A. Koloskova, S. Stich, and M. Jaggi, "Decentralized stochastic optimization and gossip algorithms with compressed communication," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 3478–3487.
[55] A. Koloskova, T. Lin, S. U. Stich, and M. Jaggi, "Decentralized deep learning with arbitrary communication compression," in *Proc. Int. Conf. Learn. Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=SkgGCkrKvH
[56] H. Tang, S. Gan, C. Zhang, T. Zhang, and J. Liu, "Communication compression for decentralized training," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 7663–7673.
[57] H. Tang *et al.*, "Deepsqueeze: Decentralization meets error-compensated compression," 2019, *arXiv:1907.07346*.
[58] Y. Kajiyama, N. Hayashi, and S. Takai, "Linear convergence of consensus-based quantized optimization for smooth and strongly convex cost functions," *IEEE Trans. Autom. Control*, vol. 66, no. 3, pp. 1254–1261, Mar. 2020.
[59] Z. Li, Y. Liao, K. Huang, and S. Pu, "Compressed gradient tracking for decentralized optimization with linear convergence," 2021, *arXiv:2103.13748*.
[60] Y. Xiong, L. Wu, K. You, and L. Xie, "Quantized distributed gradient tracking algorithm with linear convergence in directed networks," 2021, *arXiv:2104.03649*.
[61] J. Zhang, K. You, and L. Xie, "Innovation compression for communication-efficient distributed optimization with linear convergence," 2021, *arXiv:2105.06697*.
[62] T. C. Aysal, M. E. Yildiz, A. D. Sarwate, and A. Scaglione, "Broadcast gossip algorithms for consensus," *IEEE Trans. Signal Process.*, vol. 57, no. 7, pp. 2748–2761, Jul. 2009.
[63] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2508–2530, Jun. 2006.
[64] S. Pu and A. Nedić, "Distributed stochastic gradient tracking methods," *Math. Program.*, vol. 187, no. 1, pp. 409–457, 2021.
[65] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge, U.K.: Cambridge Univ. Press, 2012.
[66] A. Berman and R. J. Plemmons, *Nonnegative Matrices in the Mathematical Sciences*. Philadelphia, PA, USA: SIAM, 1994.
[67] K. Mansouri, T. Ringsted, D. Ballabio, R. Todeschini, and V. Consonni, "Quantitative structure-activity relationship models for ready biodegradability of chemicals," *J. Chem. Inf. Model.*, vol. 53, no. 4, pp. 867–878, 2013.

Zhuoqing Song received the B.S. degree in mathematics and applied mathematics in 2019 from Fudan University, Shanghai, China, where he is currently working toward the Ph.D. degree in applied mathematics. His research interests include decentralized optimization, graph algorithms, and machine learning.

Lei Shi received the Ph.D. degree in mathematics from the City University of Hong Kong, Hong Kong, in 2010. He is currently a Professor with the School of Mathematical Sciences, Fudan University, Shanghai, China. His research interests include learning theory, approximation theory and optimization. He was jointly awarded by University of Science and Technology of China in 2010.



Shi Pu received the B.S. degree from Peking University, Beijing, China, in 2012, and the Ph.D. degree in systems engineering from the University of Virginia, Charlottesville, VA, USA, in 2016. He is currently an Assistant Professor with the School of Data Science, The Chinese University of Hong Kong, Shenzhen, China. He is also affiliated with Shenzhen Research Institute of Big Data. He was a Postdoctoral Associate with the University of Florida, Gainesville, FL, USA, Arizona State University, Tempe, AZ, USA, and Boston University, Boston, MA, USA, from 2016 to 2019. His research interests include distributed optimization, network science, machine learning, and game theory.

Ming Yan received the B.S. and M.S. degrees from the University of Science and Technology of China, Hefei, China, and the Ph.D. degree from the University of California, Los Angeles, Los Angeles, CA, USA, in 2012. He is currently an Associate Professor with the Department of Computational Mathematics, Science and Engineering and the Department of Mathematics, Michigan State University, East Lansing, MI, USA. His research interests include optimization methods and their applications in sparse recovery and regularized inverse problems, variational methods for image processing, and parallel and distributed algorithms for solving big data problems.