A general framework for decentralized optimization with first-order methods

Journal:	*Proceedings of the IEEE*
Manuscript ID	0007-SIP-2020-PIEEE.R1
Manuscript Categories:	Special Issue Paper
Date Submitted by the Author:	n/a
Complete List of Authors:	Xin, Ran; Carnegie Mellon University, Electrical and Computer Engineering Pu, Shi; The Chinese University of Hong Kong - Shenzhen, Research Institute of Big Data and Institute for Data and Decision Analytics Nedich, Angelia; Arizona State University, Electrical, Computer, and Energy Engineering Khan, Usman; Tufts University, Electrical and Computer Engineering
Keyword:	Optimization methods, Distributed algorithms, Stochastic optimization, Machine learning

A general framework for decentralized optimization with first-order methods

Ran Xin, Shi Pu, Angelia Nedić, and Usman A. Khan

***Abstract*—Decentralized optimization to minimize a finite sum of functions has been a significant focus within control and signal processing research due to its relevance to optimal control and signal estimation problems. More recently, the emergence of large-scale data science and machine learning has led to a resurgence of activity in this area. In this paper, we discuss decentralized first-order methods, which have found tremendous success in emerging machine learning problems where such methods, due to their simplicity, serve as the first method of choice for many complex inference and training tasks. In particular, we provide a general framework of decentralized first-order methods that is applicable to undirected and directed communication networks alike, and show that much of the existing work on optimization and consensus can be related explicitly to this framework. We further extend the discussion to decentralized stochastic first-order methods that rely on stochastic gradients at each node and describe how variance reduction, previously shown to have promise in the centralized settings, are able to improve the performance of decentralized methods when combined with gradient tracking. We motivate and demonstrate the effectiveness of the first-order methods in discussion on machine learning and signal processing problems that arise in decentralized environments.**

I. Introduction

Minimizing a cost function to select an optimal action or decision has been an important problem in mathematics, science, and engineering. The cost function, say $F:\mathbb{R}^p\rightarrow\mathbb{R}$, typically quantifies the risk in fitting data or measurements under a model parameterized by $\mathbf{x}\in\mathbb{R}^p$. An optimal model or decision $\mathbf{x}^*$ is chosen as the one that minimizes the corresponding risk F. Optimization theory and algorithms provide the fundamental tools to address such problems. Examples include the classical signal estimation and optimal control problems where the goal in the former is to minimize the estimation error and in the latter is to minimize the cost of control actions. More recently, with the advent of modern computational machinery, complex nonlinear problems, such as image classification, natural language processing, and deep learning, have enabled a resurgence of interest in the domain of optimization theory and methods.

R. Xin is with the Electrical and Computer Engineering Department at Carnegie Mellon University, Pittsbirgh, PA, USA. S. Pu is with Institute for Data and Decision Analytics, The Chinese University of Hong Kong, Shenzhen, China and Shenzhen Research Institute of Big Data. A. Nedić is with the School of Electrical, Computer and Energy Engineering at Arizona State University, Tempe, AZ, USA. U. A. Khan is with the ECE Department at Tufts University, Medford, MA, USA. The work of R. Xin and U. A. Khan has been partially supported by NSF under grants CMMI-1903972 and CBET-1935555. The work of S. Pu has been partially supported by the Shenzhen Research Institute of Big Data (SRIBD) Startup Fund No. J00120190011. The work by A. Nedić has been supported by NSF grant CCF-1717391. Email addresses: `ranx@andrew.cmu.edu`, `pushi@cuhk.edu.cn`, `angelia.nedich@asu.edu`, `khan@ece.tufts.edu`.

Classical optimization methods for minimizing a (smooth) function F are built on a simple observation that moving along its negative gradient $-\nabla F$ decreases the function value. Thus, given $\mathbf{x}$ and $\nabla F(\mathbf{x})$, both in $\mathbb{R}^p$, a step to decrease $F(\mathbf{x})$ is $\mathbf{x}_+=\mathbf{x}-\alpha\nabla F(\mathbf{x})$, if α, which controls the size of the step taken in the descent direction, is small enough, i.e., we have that $F(\mathbf{x}_+)\leq F(\mathbf{x})$. The algorithm that recursively applies this step is well-known as *gradient descent* and it finds a minimizer of the corresponding cost function under certain conditions on the function F and the step-size α. Moreover, gradient descent is a *first-order* method as it only uses the gradient information of the cost function, in contrast to for example second-order methods that typically compute at each iteration the inverse of the Hessian $\nabla^2 F$ of the cost.

In this article, our focus is on decentralized optimization, or decentralized optimization, in networks where data samples are available across multiple nodes, such as machines, sensors, robots or mobile devices, and the nodes communicate with each other according to a peer-to-peer network, without a central coordinator, to solve the underlying optimization problem. Such problems are prevalent in modern-day machine learning where for example a large collection of images is stored on multiple machines in a data center for the purpose of image classification. Moreover, classical applications like sensor networks and robotic swarms also fit in this paradigm where the sensors and robots collect measurements in order to learn an underlying phenomenon, navigate an environment, or decide on an optimal control action. In such settings, the data samples available at the ith node lead to a local cost f_i, and the global goal of the networked nodes is to *agree* on a minimizer of the average cost $F=\frac{1}{n}\sum_{i=1}^n f_i$ based on the entire data across n nodes. In related applications of interest, raw data sharing among the nodes is often not permitted due to limited communication resources and/or the private nature of the local data, such as text messages or medical images.

Decentralized first-order methods thus rely on information exchange among the nodes to build the solution of the global optimization problem. Each node i iterates on a local variable $\mathbf{x}_k^i$ that estimates the minimizer $\mathbf{x}^*$, at iteration k, and updates this estimate as a function of the *neighboring estimates* and *local gradients*. In other words, the nodes do not share their data (local gradients) and only communicate their estimates of the minimizer (and other auxiliary variables). Early work along these lines can be found in [1] that is built on average consensus and in [2] where a diffusion principle is used for agreement. More recent developments include *gradient tracking* where the local descent ∇f_i at each node i is replaced with a local iterative tracker of the global descent ∇F. Thus, as these trackers approach the global gradient, each local iterate $\mathbf{x}_k^i$ descends in the global direction and converges exponentially (linearly on the log scale) to $\mathbf{x}^*$ for smooth and strongly convex cost functions (in a similar way as the centralized gradient descent). Our primary focus in this article is on the class of smooth and strongly convex cost functions for the ease of illustrating the key technical ideas. We emphasize however that most of the algorithms described herein apply to non-convex problems directly.

The first half of this article is devoted to decentralized first-order methods based on gradient tracking where the focus is on a recently introduced algorithm, **`AB`** [3] or **`Push-Pull`** [4], that utilizes a novel application of both row and column stochastic weights to achieve linear (on the log scale) convergence for smooth and strongly convex cost functions. Since doubly stochastic weights are not required in **`AB`**/**`Push-Pull`**, it is applicable to arbitrary undirected and directed networks alike. We further describe how **`AB`**/**`Push-Pull`** unifies much of the existing work (over both undirected and directed networks) on decentralized first-order methods that use gradient tracking and subsumes some non-trivial average consensus algorithms as special cases. Similarly, we emphasize the push and pull aspects of the information exchange in **`AB`**/**`Push-Pull`**, enabled by the column and row stochastic weights, respectively, and show how **`AB`**/**`Push-Pull`** unifies various communication architectures.

The second half of this article is devoted to decentralized stochastic first-order methods detailing the current state-of-the-art and open problems where some progress has been made only recently. In decentralized stochastic methods, each node has access only to an imperfect local gradient, which results from either an incomplete knowledge of the true gradient or sampling a small subset from a large number of local data samples. In this context, we describe how gradient tracking, previously successful in non-stochastic cases, does not lead to the same performance improvement and show that exact linear convergence to the global minimum (for smooth and strongly convex problems) can be obtained when gradient tracking is further combined with what is known as *variance reduction* in centralized optimization. We emphasize that much of the existing work on decentralized stochastic problems has focused on undirected networks and the results on directed networks are rather restrictive.

A. *Literature Survey*

Decentralized optimization has been a topic of significant research, see e.g., [1]–[12]. Optimization in undirected networks based on average consensus [13]–[15] includes [1], [16]–[19] that are built on doubly stochastic network weight matrices. Relevant work that builds on diffusion can be found in [2], [20], [21]. For general directed networks, the construction of doubly stochastic weights may be infeasible and optimization over directed networks thus builds on consensus with row and/or column stochastic weights. To this aim, the methods in [22]–[30] use push-sum consensus [31]–[33] that requires a division with a certain eigenvector of the corresponding weight matrix. In contrast, the methods in [24], [25] are based on surplus consensus [34] that circumvent eigenvector division by employing both row and column stochastic weights simultaneously.

Decentralized stochastic optimization can be divided into two types: (i) *streaming*, where an imprecise (stochastic) gradient is drawn from a underlying probability distribution at each node; or (ii) *batch*, where a finite collection of data samples is already available at each node and a stochastic gradient is computed from samples drawn randomly from the data. Early work on decentralized problems with streaming data can be found in [7], [35]–[41]. Stochastic optimization over batch data have garnered a strong activity in the centralized settings where modern methods hinge on certain variance-reduction techniques that leverage the finite-sum structure of the cost function to accelerate the standard stochastic gradient descent (SGD); related work includes SAG [42], SVRG [43], SAGA [44], Katyusha [45], and SARAH [46]. Existing variance-reduced decentralized optimization methods can be found in [47]–[52].

Although not discussed in this paper, second-order methods and algorithms based on the curvature of the cost functions can be found in [53]–[57]. Similarly, ADMM (alternating direction method of multipliers) and other primal-dual methods have also been used in decentralized optimization [58]–[66]. See also related work in [67]–[69], which considers methods based on dual gradients. Work to incorporate communication and computation imperfection and trade-offs, for example, time-varying and random graphs, asynchronous methods, quantization, and gradient sparsification can be found in [70]–[78].

B. *Outline of the Article*

We now describe the rest of this article. In Section II, we provide examples and preliminaries on convex functions, communication graphs, and nonnegative matrices. Section III discusses early work on decentralized gradient descent and shows its applicability to directed networks with the help of row and column stochastic weights. We then describe gradient tracking and introduce the **`AB`**/**`Push-Pull`** algorithm in Section IV, which further includes a sketch of the analysis, communication architectures, and accelerated methods. Section V describes how **`AB`**/**`Push-Pull`** provides a general framework to capture many first-order methods based on gradient tracking. We then begin the discussion on decentralized stochastic first-order methods over both undirected and directed graphs in Sections VI and VII. We show how gradient tracking alone is unable to ensure exact linear convergence in this setting (Section VI), which is subsequently achieved when gradient tracking is further combined with variance reduction (Section VII). Section VIII provides a detailed numerical study on the performance, speed-up, and convergence of related algorithms. Finally, Section IX concludes the article.

The goal of this article is to provide an in-depth overview of decentralized first-order methods and to further expose the reader with rigorous yet intuitive arguments to follow the technical analysis. In several remarks distributed throughout this article, we highlight the analysis techniques, practical aspects, and other salient features of the corresponding algorithms.

C. Notation

We use lowercase letters to denote scalars in $\mathbb{R}$, lowercase bold letters to denote vectors, and uppercase letters to denote matrices. For a vector $\mathbf{x} \in \mathbb{R}^p$, we denote its ith element by $[\mathbf{x}]_i$. For a set $\mathcal{S}$, we use $|\mathcal{S}|$ to denote its cardinality. For a sequence of real numbers $\{a_k\}_{k\geq 0}$, we denote $a_k \to a$ as $\lim_{k\to\infty} a_k = a$. The matrix I_p is the $p \times p$ identity, and $\mathbf{1}_p$ (resp. $\mathbf{0}_p$) is the p-dimensional column vector of all ones (resp. zeros). For two matrices X, Y, $X \otimes Y$ denotes their Kronecker product. We use $\|\cdot\|_2$ to denote the Euclidean norm of a vector. The spectral radius of a matrix X is denoted by $\rho(X)$ while its spectral norm is denoted by $|||X|||$. A strictly positive vector $\mathbf{x} \in \mathbb{R}^p$, denoted as $\mathbf{x} > 0$, is such that each of its elements is positive. For $\mathbf{w} := [w_1, \cdots, w_p]^\top > 0$ and an arbitrary vector $\mathbf{x} := [x_1, \cdots, x_p]^\top$, the weighted infinity norm of $\mathbf{x}$ is defined as $\|\mathbf{x}\|_\infty^{\mathbf{w}} = \max_i |x_i|/w_i$ and $|||\cdot|||_\infty^{\mathbf{x}}$ is the weighted matrix norm induced by the vector norm $\|\cdot\|_\infty^{\mathbf{w}}$.

II. Problem Formulation and Motivation

In this section, we describe the canonical form of the decentralized optimization problems and emphasize real-world scenarios where such problems are applicable and essential. Decentralized optimization is finite-sum minimization formulated over networked nodes. Formally, the goal of the nodes is to solve in a cooperative manner

$$\text{Problem P0:} \qquad \min_{\mathbf{x}\in\mathbb{R}^p} F(\mathbf{x}), \qquad F(\mathbf{x}) := \frac{1}{n}\sum_{i=1}^n f_i(\mathbf{x}),$$

where each f_i is only locally accessible and processed by node i and is not shared with any other node. The cooperation (information exchange) among the nodes is peer-to-peer without the existence of a central coordinator and is typically modeled as a communication graph, see Fig. 1. In many cases of practical interest, each local cost f_i can be further decomposed as a weighted sum over local data samples available at node i, i.e.,

$$f_i(\mathbf{x}) := \sum_{j=1}^{m_i} \zeta_{i,j} \cdot f_{i,j}(\mathbf{x}), \tag{1}$$

where $\zeta_{i,j} > 0$ is a weight assigned to each data sample. The paradigm of decentralized optimization preserves the privacy of local data and achieves data parallelism, thus acting effectively as a means for flexible parallel computation. We provide some illustrative examples of this formulation in the following Section II-A. For preliminaries on related technical concepts, see Section II-B.

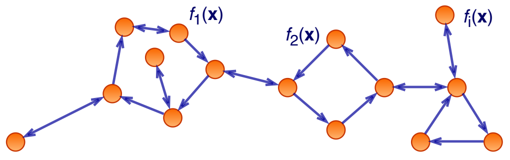


Fig. 1. Decentralized optimization over a general directed network.

A. Examples

The finite-sum minimization problems of the form P0 are quite prevalent in the areas on signal processing, control, and machine learning. We provide some examples below.

1) Signal-plus-noise model

In classical signal processing, we are often interested in finding an unknown signal $\mathbf{x} \in \mathbb{R}^p$ based on measurements $y_i = \mathbf{h}_i^\top \mathbf{x} + v_i$, obtained by a collection of sensors indexed by i, where $\mathbf{h}_i \in \mathbb{R}^p$ is the sensing matrix and $v_i \in \mathbb{R}$ is some unknown noise. Each sensor thus is tasked to find an $\mathbf{x}$ that minimizes the squared error $(y_i - \mathbf{h}_i^\top \mathbf{x})^2$. However, since the problem may be ill-conditioned and the collected measurements are noisy, collaboration among the sensors leads to a much more robust estimate. The resulting formulation is

$$\min_{\mathbf{x}\in\mathbb{R}^p} \frac{1}{n}\sum_{i=1}^n f_i(\mathbf{x}), \quad f_i(\mathbf{x}) := (y_i - \mathbf{h}_i^\top \mathbf{x})^2,$$

which is also known as the least-squares problems.

2) Linear models for binary classification

The decentralized nature of Problem P0 is more clear when we consider high-dimensional and potentially private or proprietary user data. For example, consider a set of n users interested in learning a classifier to distinguish between male and female faces. Each user i holds a collection of m_i images, vectorized as $\mathbf{z}_{i,j} \in \mathbb{R}^p$, that are labeled $y_{i,j} = +1$ for a male face, or $y_{i,j} = -1$ for a female face. The classification may be made via a linear classifier, i.e., a hyperplane $y = \mathbf{x}^\top \mathbf{z} + b$ that separates the data samples from two classes. Clearly, a classifier trained on the collection of all images across all users will have superior performance than any locally-trained classifier whose performance significantly depends on the size and quality of the local data samples. However, bringing all of the images to a central server is expensive depending on the resolution and size of the images and further requires sharing personal information. This discussion motivates the use of decentralized optimization framework to solve the following logistic regression problem:

$$\min_{\mathbf{x}\in\mathbb{R}^p, b\in\mathbb{R}} \frac{1}{n}\sum_{i=1}^n \frac{1}{m_i}\sum_{j=1}^{m_i} \widetilde{f}_{i,j}(\mathbf{x}) + \frac{\lambda}{2}\|\mathbf{x}\|^2,$$
$$\widetilde{f}_{i,j}(\mathbf{x}) := \sum_{j=1}^{m_i} \ln\left[1 + \exp\left(-(\mathbf{x}^\top \mathbf{z}_{i,j} + b)\right) y_{i,j}\right],$$

where the logistic loss $\widetilde{f}_{i,j}$ quantifies the error in the linear classifier and $\frac{\lambda}{2}\|\mathbf{x}\|^2$, for some $\lambda > 0$, is a regularization term to prevent the overfitting of the data.

3) Empirical risk minimization

Problem P0 also arises as an approximation of *expected risk minimization*, see e.g., [79] for additional details. In this context, the problem of interest is to find some model h, parameterized by $\mathbf{x} \in \mathbb{R}^p$, that maps an input $\mathbf{z} \in \mathbb{R}^{\mathbf{d}_z}$ to its corresponding output $\mathbf{y} \in \mathbb{R}^{\mathbf{d}_y}$. The setup requires defining a loss function $l(h(\mathbf{z};\mathbf{x}), \mathbf{y})$ that quantifies the mismatch between the model prediction $h(\mathbf{z};\mathbf{x})$, under the parameter $\mathbf{x}$, and the actual output data $\mathbf{y}$. Assuming that the data $(\mathbf{z}, \mathbf{y})$ belongs to an underlying distribution $\mathcal{P}$, the goal here is to find the optimal parameter $\mathbf{x}^*$ that minimizes the expected loss over $\mathcal{P}$, i.e., $\min_{\mathbf{x}\in\mathbb{R}^p} \mathbb{E}_{(\mathbf{z},\mathbf{y})\sim\mathcal{P}} l(h(\mathbf{z};\mathbf{x}), \mathbf{y})$. However, the distribution $\mathcal{P}$ is often intractable in practice and each node i usually has access to a large set of data samples $\{\mathbf{z}_{i,j}, \mathbf{y}_{i,j}\}_{j=1}^{m_i}$, which can be considered as the independent and identically distributed (i.i.d.) realizations from $\mathcal{P}$. The average loss across all nodes and all data thus serves as an appropriate surrogate for the expected risk and the corresponding problem is often called *empirical risk minimization*, i.e.,

$$\min_{\mathbf{x}\in\mathbb{R}^p} \frac{1}{n}\sum_{i=1}^{n}\frac{1}{m_i}\sum_{j=1}^{m_i} f_{i,j}(\mathbf{x}), \quad f_{i,j}(\mathbf{x}) := l(h(\mathbf{z}_{i,j};\mathbf{x}),\mathbf{y}_{i,j}).$$

This formulation captures a wide range of machine learning models, including deep neural networks.

B. Preliminaries

We now briefly describe some mathematical concepts that aid the technical discussion in this paper.

1) Convex functions

A convex function $f:\mathbb{R}^p\to\mathbb{R}$ is such that for any $0<\gamma<1$ and $\forall \mathbf{x}_1,\mathbf{x}_2\in\mathbb{R}^p$,

$$f(\gamma\mathbf{x}_1+(1-\gamma)\mathbf{x}_2)\leq \gamma f(\mathbf{x}_1)+(1-\gamma)f(\mathbf{x}_2). \tag{2}$$

The above definition says that a convex function always stays below a line that connects any two points on the function. If f is differentiable, an equivalent definition of convexity is that it lies above all of its tangents, i.e., $\forall \mathbf{x}_1,\mathbf{x}_2\in\mathbb{R}^p$,

$$f(\mathbf{x}_2)\geq f(\mathbf{x}_1)+\nabla f(\mathbf{x}_1)^\top(\mathbf{x}_2-\mathbf{x}_1), \tag{3}$$

where $\nabla f(\mathbf{x}_1)$ denotes the gradient (derivative) of f at $\mathbf{x}_1$. The convexity conditions above are general and do not guarantee the existence of a global minimum. The notion of strong convexity, as defined next, ensures that the global minimum of f exists and is unique. A function $f:\mathbb{R}^p\to\mathbb{R}$ is μ-strongly convex if $\forall \mathbf{x}_1,\mathbf{x}_2\in\mathbb{R}^p$,

$$f(\mathbf{x}_2)\geq f(\mathbf{x}_1)+\nabla f(\mathbf{x}_1)^\top(\mathbf{x}_2-\mathbf{x}_1)+\tfrac{\mu}{2}\|\mathbf{x}_2-\mathbf{x}_1\|_2^2, \tag{4}$$

for some $\mu>0$. It can be verified that strong convexity is stronger than (3) in the sense that it further imposes a quadratic lower bound. Finally, a function $f:\mathbb{R}^p\to\mathbb{R}$, not necessarily convex, is L-smooth if for some $L>0$ and $\forall \mathbf{x}_1,\mathbf{x}_2\in\mathbb{R}^p$,

$$\|\nabla f(\mathbf{x}_1)-\nabla f(\mathbf{x}_2)\|_2\leq L\|\mathbf{x}_1-\mathbf{x}_2\|_2, \tag{5}$$

which implies that

$$f(\mathbf{x}_2)\leq f(\mathbf{x}_1)+\nabla f(\mathbf{x}_1)^\top(\mathbf{x}_2-\mathbf{x}_1)+\tfrac{L}{2}\|\mathbf{x}_2-\mathbf{x}_1\|_2^2. \tag{6}$$

Clearly, an L-smooth function is controlled by a quadratic upper bound. We denote the class of functions that are both μ-strongly convex and L-smooth as $\mathcal{S}_{\mu,L}$. Note that each $f\in\mathcal{S}_{\mu,L}$ is subject to both the lower and upper quadratic bounds in (4) and (6); we thus always have $\mu\leq L$. The ratio $\kappa:=L/\mu$ is called the condition number of f and the functions with large κ are said to be ill-conditioned. A rather simple example of a function in this class $\mathcal{S}_{\mu,L}$ is $f(\mathbf{x})=\mathbf{x}^\top Q\mathbf{x}+\mathbf{b}^\top\mathbf{x}+c$, for $\mathbf{x}\in\mathbb{R}^p$ and a positive-definite matrix $Q\in\mathbb{R}^{p\times p}$; see [80] for more details on convex functions.

In this article, the algorithms in question are discussed under the assumption that $f_i\in\mathcal{S}_{\mu,L},\forall i\in\mathcal{V}$. Thus, the global cost is also such that $F:=\frac{1}{n}\sum_{i=1}^n f_i\in\mathcal{S}_{\mu,L}$ (unless explicitly mentioned otherwise) and we denote the unique minimizer of F as $\mathbf{x}^*$. We refer to the appropriate literature where the results are generalized to other classes, for example, L-smooth but possibly non-convex functions.

2) Communication Graph

We now define a graph to formally characterize the information exchange among the nodes in the communication network. Consider n nodes interacting over a potentially directed graph $\mathcal{G}=\{\mathcal{V},\mathcal{E}\}$, where $\mathcal{V}=\{1,\ldots,n\}$ is the set of nodes, and $\mathcal{E}\subseteq\mathcal{V}\times\mathcal{V}$ is a collection of ordered pairs (i,r) such that node r can send information to node i, i.e., $i\leftarrow r$. Since $\mathcal{G}$ is not necessarily undirected, $i\leftarrow r \not\Rightarrow i\rightarrow r$. Let $\mathcal{N}_i^{\text{in}}$ denote the set of incoming neighbors of node i, i.e., nodes that can send information to node i. Similarly, $\mathcal{N}_i^{\text{out}}$ denotes the set of outgoing neighbors, i.e., nodes that receive information from node i. We assume that $i\in\ \mathcal{N}_i^{\text{in}}\cap\mathcal{N}_i^{\text{out}}$. When the graph is undirected, we denote $\mathcal{N}_i:=\mathcal{N}_i^{\text{in}}=\mathcal{N}_i^{\text{out}}$. A directed graph is said to be strongly connected if there exists a directed path between any two nodes. Given a nonnegative matrix $M=\{m_{ir}\}\in\mathbb{R}^{n\times n}$, the directed graph induced by the matrix M is denoted by $\mathcal{G}_M=\{\{1,2,\ldots,n\},\mathcal{E}_M\}$, where $(i,r)\in\mathcal{E}_M$ if and only if $m_{ir}>0$. Conversely, given a directed graph $\mathcal{G}=\{\mathcal{V},\mathcal{E}\}$, we say $\underline{W}=\{w_{ir}\}$ is a weight matrix associated with $\mathcal{G}$ if $w_{ir}>0$ whenever there is a directed link from node r to node i.

3) Nonnegative matrices

A nonnegative (resp. positive) matrix is such that all of its elements are nonnegative (resp. positive). A matrix is row stochastic (resp. column stochastic) if it is nonnegative and all of its rows (resp. columns) sum to one and it is doubly stochastic if it is both row and column stochastic. The spectral radius of row, column, and doubly stochastic matrices is one, and one is also an eigenvalue of the corresponding matrix. A nonnegative matrix M is irreducible if its induced graph $\mathcal{G}_M$ is strongly connected and M is further primitive[1] if its trace is positive. By Perron-Frobenius theorem, for a primitive and row stochastic matrix $\underline{A}$, we denote its positive left eigenvector corresponding to the eigenvalue 1 as $\boldsymbol{\pi}_A$ such that $\boldsymbol{\pi}_A^\top\mathbf{1}_n=1$. Similarly, for a primitive and column stochastic matrix $\underline{B}$, we denote its positive right eigenvector corresponding to the eigenvalue 1 as $\boldsymbol{\pi}_B$ such that $\mathbf{1}_n^\top\boldsymbol{\pi}_B=1$. Clearly,

$$\underline{A}\mathbf{1}_n=\mathbf{1}_n,\ \boldsymbol{\pi}_A^\top\underline{A}=\boldsymbol{\pi}_A^\top, \qquad \underline{B}\boldsymbol{\pi}_B=\boldsymbol{\pi}_B,\ \mathbf{1}_n^\top\underline{B}=\mathbf{1}_n^\top,$$

and it can be shown that

$$\underline{A}^\infty:=\lim_{k\to\infty}\underline{A}^k=\mathbf{1}_n\boldsymbol{\pi}_A^\top, \qquad \underline{B}^\infty:=\lim_{k\to\infty}\underline{B}^k=\boldsymbol{\pi}_B\mathbf{1}_n^\top.$$

Additional details on these concepts can be found in [81].

III. Decentralized Gradient Descent

The classical first-order method to minimize a differentiable function $F:\mathbb{R}^p\to\mathbb{R}$ is the gradient descent algorithm [82]:

$$\mathbf{x}_{k+1}=\mathbf{x}_k-\alpha\cdot\nabla F(\mathbf{x}_k), \qquad \forall k\geq 0, \tag{7}$$

where $\{\mathbf{x}_k\}_{k\geq0}$ is a sequence of estimates of a minimizer of F and $\alpha>0$ is a constant step-size. When F is L-smooth (but not necessarily convex) and $\alpha\in(0,\frac{1}{L}]$, we have that $\|\nabla f(\mathbf{x}_k)\|\to 0$, i.e., gradient descent finds a critical point[2] of F asymptotically. When F is further μ-strongly convex, i.e., $F\in\mathcal{S}_{\mu,L}$ as considered in this article, then a

[1]Formally, a nonnegative matrix is primitive if it is irreducible and only has one non-zero eigenvalue of maximum modulus [81].

[2]A vector $\mathbf{x}^*\in\mathbb{R}^p$ is called a critical point of F if $\nabla F(\mathbf{x}^*)=\mathbf{0}_p$.

critical point $\mathbf{x}^*$ is further the unique global minimum of F. In this case, we have that $\|\mathbf{x}_k - \mathbf{x}^*\| \leq (1-\mu\alpha)^k \|\mathbf{x}_0 - \mathbf{x}^*\|$ for any $\alpha \in (0, \frac{1}{L}]$, i.e., gradient descent converges to $\mathbf{x}^*$ at a linear (on the log-scale) rate [80], [82]. In the rest of this article, we discuss various decentralized methods of gradient descent type that are suitable for many practical problems.

When gradient descent is implemented at node i, without any cooperation, to solve the decentralized optimization Problem P0, we note that (7) only finds the minimizer of f_i but not the minimizer of the global cost $F = \frac{1}{n}\sum_{i=1}^n f_i$, in general. A decentralized method thus must have a means of fusing information over the nodes in the network such that the estimate $\mathbf{x}_k^i$ at each node i is steered towards the global minimum $\mathbf{x}^*$ of F. In other words, a decentralized optimization algorithm requires two key ingredients: (i) *Agreement*: all nodes must agree on the same estimate; and, (ii) *Optimality*: the agreement must be on the global minimum. Agreement, required by any decentralized algorithm, is typically achieved with the help of average-consensus [13]–[15], when the underlying graph is undirected, or by push-sum [31]–[33] or surplus consensus [34] when the communication is directed. We next discuss decentralized optimization methods that build directly on top of these consensus algorithms.

A. Decentralized gradient descent: Undirected graphs

We start with the case of connected, undirected graphs where the construction of doubly stochastic (network) weight matrices in a decentralized manner is straightforward.[3] A well-known and perhaps the simplest decentralized solution for Problem P0 is the Decentralized Gradient Descent (**DGD**) [1], [2], [7] Let $\mathbf{x}_k^i \in \mathbb{R}^p$ denote the estimate of $\mathbf{x}^*$ at node i and iteration k. **DGD** runs the following iterations at each node i:

$$\mathbf{x}_{k+1}^i = \sum_{r\in\mathcal{N}_i} w_{ir}\mathbf{x}_k^r - \alpha_k \cdot \nabla f_i(\mathbf{x}_k^i), \qquad \forall k \geq 0, \qquad (8)$$

where w_{ir} is a weight that node i assigns to each $\mathbf{x}_k^r$ in its neighborhood and the weight matrix $\underline{W} = \{w_{ir}\} \in \mathbb{R}^{n\times n}$ is doubly stochastic. Note that when $\alpha_k = 0, \forall k$, **DGD** reduces to the classical average consensus, where $\mathbf{x}_k^i \to \frac{1}{n}\sum_{i=1}^n \mathbf{x}_0^i$, $\forall i$, i.e., all nodes agree on the average of their initial states.

To understand **DGD**, let us consider its vector-matrix form:

$$\mathbf{x}_{k+1} = W\mathbf{x}_k - \alpha_k \cdot \nabla \mathbf{f}(\mathbf{x}_k), \qquad (9)$$

where $W = \underline{W} \otimes I_p \in \mathbb{R}^{pn\times pn}$, $\mathbf{x}_k \in \mathbb{R}^{pn}$ concatenates the local $\mathbf{x}_k^i$'s, and $\nabla\mathbf{f}(\mathbf{x}_k) \in \mathbb{R}^{pn}$ concatenates the local gradients $\nabla f_i(\mathbf{x}_k^i)$'s. Assuming that $\alpha_k \to 0$, it can be shown that each $\mathbf{x}_k^i \to \overline{\mathbf{x}}_k$, where $\overline{\mathbf{x}}_k := \frac{1}{n}\sum_{i=1}^n \mathbf{x}_k^i$ is the mean iterate over the network, since **DGD** asymptotically reduces to average consensus. Moreover, since $\underline{W}$ is column stochastic, we get, after multiplying both sides of (9) by $\frac{1}{n}(\mathbf{1}_n^\top \otimes I_p)$, that

$$\overline{\mathbf{x}}_{k+1} = \overline{\mathbf{x}}_k - \alpha_k \cdot \frac{1}{n}\sum_{i=1}^n \nabla f_i(\mathbf{x}_k^i). \qquad (10)$$

Observe that $\frac{1}{n}\sum_{i=1}^n \nabla f_i(\mathbf{x}_k^i) \to \nabla F(\overline{\mathbf{x}}_k)$ as $\mathbf{x}_k^i \to \overline{\mathbf{x}}_k, \forall i$. Therefore, when $\alpha_k \to 0$ and the weight matrix $\underline{W}$ is doubly stochastic, we have that each $\mathbf{x}_k^i \to \overline{\mathbf{x}}_k$ and $\overline{\mathbf{x}}_k$ converges to the minimum of $F = \frac{1}{n}\sum_{i=1}^n f_i$, from (7), which guarantees simultaneously the agreement and optimality requirement of decentralized optimization. The rigorous analysis of **DGD** can be found in [2], [16], [70]. We make a few remarks as follows.

*Remark 1 (Performance of **DGD**–Rate/accuracy tradeoffs):* **DGD** converges sublinearly to the exact global minimum $\mathbf{x}^*$ of Problem P0 for decaying step-sizes such that $\alpha_k \to 0$. Under a constant step-size α, **DGD** converges linearly, however, to an inexact solution with accuracy $\mathcal{O}(\alpha)$. Clearly, a larger constant step-size leads to faster convergence albeit with worse accuracy; see [16], [83], [84] for detailed analyses. We will revisit this rate/accuracy tradeoff in Section IV and show that this issue can be fixed by a technique called gradient tracking.

Remark 2 (Consensus + innovation and diffusion learning): **DGD**-type algorithms of the form (8) are also known as "consensus + innovation" and "diffusion learning" in the context of signal estimation problems especially when the cost functions are quadratic; see [7], [85]–[87] and references therein.

B. Decentralized gradient descent: Directed graphs

We now consider strongly connected, directed graphs where the network weight matrices are either row stochastic or column stochastic but not doubly stochastic, in general. As a consequence, when the weight matrix in (8) is column stochastic but not row stochastic, the nodes do not agree since the right eigenvector corresponding to the eigenvalue of 1 is not $\mathbf{1}_n$ that is essential for agreement or consensus [25]. Similarly, when the weights are row stochastic but not column stochastic, the nodes agree however on a sub-optimal solution that is the minimum of a weighted average of local functions (and not the mean). We formally discuss these issues next.

1) **DGD** with column stochastic weights

Let $\underline{B} = \{b_{ir}\} \in \mathbb{R}^{n\times n}$ be a primitive, column stochastic weight matrix such that $\mathbf{1}_n^\top \underline{B} = \mathbf{1}_n^\top$ and $\underline{B}\boldsymbol{\pi}_B = \boldsymbol{\pi}_B$. Consider **DGD** (9) with $\underline{B}$, i.e.,

$$\mathbf{x}_{k+1} = B\mathbf{x}_k - \alpha_k \cdot \nabla\mathbf{f}(\mathbf{x}_k), \qquad (11)$$

where $B = \underline{B} \otimes I_p$. Recall that $\underline{B}^k \to \underline{B}^\infty := \boldsymbol{\pi}_B \mathbf{1}_n^\top$ and consider the convergence of (11) when $\alpha_k = 0, \forall k \geq 0$, i.e.,

$$\mathbf{x}_k \to B^\infty \mathbf{x}_0 = (\boldsymbol{\pi}_B\mathbf{1}_n^\top \otimes I_p)\mathbf{x}_0 = (\boldsymbol{\pi}_B \otimes I_p)(\mathbf{1}_n^\top \otimes I_p)\mathbf{x}_0,$$

which shows that $\mathbf{x}_k^i \to [\boldsymbol{\pi}_B]_i \sum_{r=1}^n \mathbf{x}_0^r, \forall i$. In other words, when the weights are only column stochastic, the nodes do not agree because of the non-identical elements in $\boldsymbol{\pi}_B$. We observe however that the scaled iterates $\mathbf{x}_k^i/(n[\boldsymbol{\pi}_B]_i)$ converge to the average of the initial states of the nodes, i.e., $\mathbf{x}_k^i/(n[\boldsymbol{\pi}_B]_i) \to \frac{1}{n}\sum_{r=1}^n \mathbf{x}_0^r, \forall i$, but since no node in the network has the knowledge of the eigenvector $\boldsymbol{\pi}_B$, another set of iterations is required to asymptotically estimate $[\boldsymbol{\pi}_B]_i$ at node i. The resulting algorithm is known as **`Gradient-Push`** [23], [88], [89], which takes the following form at each node i:

$$z_{k+1}^i = \sum_{r\in\mathcal{N}_i^{\text{in}}} b_{ir} z_k^r, \qquad (12a)$$

$$\mathbf{x}_{k+1}^i = \sum_{r\in\mathcal{N}_i^{\text{in}}} b_{ir}\mathbf{x}_k^r - \alpha_k \cdot \nabla f_i(\mathbf{w}_k^i), \qquad (12b)$$

$$\mathbf{w}_{k+1}^i = \frac{\mathbf{x}_{k+1}^i}{z_{k+1}^i}, \qquad (12c)$$

[3]Popular methods include Metroplis and Laplacian weights [70].

where $\mathbf{x}_0^i = \mathbf{w}_0^i \in \mathbb{R}^p$ is arbitrary and $z_0^i = 1, \forall i$. It is straightforward to verify that $z_k^i \to n[\boldsymbol{\pi}_B]_i, \forall i$, and thus if $\alpha_k \to 0$, $\mathbf{w}_k$ achieves agreement. On the other hand, the column stochasticity of $\underline{B}$ guarantees the optimality of the **Gradient-Push** as discussed earlier for (10).

*2) **DGD** with row stochastic weights*

Consider **DGD** now with row stochastic weights $\underline{A} = \{a_{ir}\}$:

$$\mathbf{x}_{k+1} = A\mathbf{x}_k - \alpha_k \cdot \nabla \mathbf{f}(\mathbf{x}_k), \tag{13}$$

where $A = \underline{A} \otimes I_p$. We have that $\underline{A}\mathbf{1}_n = \mathbf{1}_n$ and $\boldsymbol{\pi}_A^\top \underline{A} = \boldsymbol{\pi}_A^\top$, where $\boldsymbol{\pi}_A > 0$ and $\underline{A}^k \to \underline{A}^\infty := \mathbf{1}_n \boldsymbol{\pi}_A^\top$. In other words, the row stochasticity of the weight matrix leads to an agreement among the nodes. In particular, without gradient corrections, it is straightforward to show that $\mathbf{x}_k^i \to \sum_{r=1}^n [\boldsymbol{\pi}_A]_r \mathbf{x}_0^r$, at each node i. In (13), as $\alpha_k \to 0$, we thus have that $\mathbf{x}_k^i \to \sum_{r=1}^n [\boldsymbol{\pi}_A]_r \mathbf{x}_k^r := \widehat{\mathbf{x}}_k, \forall i$. Multiplying both sides of (13) by $(\boldsymbol{\pi}_A^\top \otimes I_p)$, we obtain

$$\widehat{\mathbf{x}}_{k+1} = \widehat{\mathbf{x}}_k - \alpha_k \cdot \sum_{r=1}^n [\boldsymbol{\pi}_A]_r \nabla f_r(\mathbf{x}_k^r). \tag{14}$$

From the above discussion, we conclude that each node approaches $\widehat{\mathbf{x}}_k$, which converges to the minimum of a *weighted average* of the local costs f_i. The algorithm to find the global minimum of F easily follows by dividing each ∇f_i by $[\boldsymbol{\pi}_A]_i$. Similar to gradient-push, separate iterations however are required to estimate the eigenvector $\boldsymbol{\pi}_A$ since it is not locally known at any node. The resulting algorithm [90], termed as **DGD-RS**, is given by

$$\mathbf{x}_{k+1}^i = \sum_{r \in \mathcal{N}_i^{\text{in}}} a_{ir}\mathbf{x}_k^r - \alpha_k \cdot \frac{\nabla f_i(\mathbf{x}_k^i)}{[\mathbf{e}_k^i]_i}, \tag{15a}$$

$$\mathbf{e}_{k+1}^i = \sum_{r \in \mathcal{N}_i^{\text{in}}} a_{ir}\mathbf{e}_k^r, \tag{15b}$$

where $\mathbf{x}_0^i \in \mathbb{R}^p$ is arbitrary and $\mathbf{e}_0^i \in \mathbb{R}^n$ is a vector of zeros with a one at the ith location. The iterations in (15b) asymptotically estimate the eigenvector $\boldsymbol{\pi}_A$ of $\underline{A}$. To see that, let $E_k = [\mathbf{e}_k^1 \;\ldots\; \mathbf{e}_k^n]^\top$, thus $E_0 = I_n$. We note that

$$E_{k+1} = \underline{A}E_k \to \underline{A}^\infty E_0 = \mathbf{1}_n \boldsymbol{\pi}_A^\top,$$

that is to say $\mathbf{e}_k^i \to \boldsymbol{\pi}_A, \forall i$. However, implementing these iterations requires each node to have a unique identifier in order to select the appropriate element from $\boldsymbol{\pi}_A$.

Remark 3 (Column and row stochastic weights over directed graphs): A column stochastic weight matrix $\underline{B} = \{b_{ir}\}$ is often constructed as $b_{ir} = 1/|N_i^{\text{out}}|, \forall r \in N_i^{\text{out}}$. This formulation requires each node to know its out degree, i.e., $|N_i^{\text{out}}|$. A row stochastic weight matrix $\underline{A} = \{a_{ir}\}$ can be easily constructed as $a_{ir} = 1/|N_i^{\text{in}}|, \forall r \in N_i^{\text{in}}$ since each node can locally assign weights to the information it receives.

Remark 4 (Average-consensus over directed graphs): It can be easily verified that (12a)–(12c) recover the average of the initial conditions $\mathbf{x}_0^i$'s when $\alpha_k = 0, \forall k$. This algorithm is well-known as push-sum that implements average-consensus with the help of column stochastic weights [31], [32]. Similarly, average-consensus with row stochastic weights can be obtained by choosing $\alpha_k = 0, \forall k$, and modifying (15a) as $\mathbf{x}_{k+1}^i = \sum_{r=1}^n a_{ir} \frac{\mathbf{x}_k^r}{[\mathbf{e}_k^i]_i}$; see also Section V-C.

Remark 5 (Eigenvector estimation): Based on the previous discussion, we note that over directed graphs there is a certain imbalance that is caused by not having $\mathbf{1}_n$ as either the left or the right eigenvector corresponding to the eigenvalue 1 of the corresponding weight matrices. When the weights are column stochastic, this imbalance manifests itself in disagreement of the estimates, whereas in the row stochastic case, this imbalance causes convergence to the minimum of a weighted sum of local cost functions. To overcome this imbalance, a division by the appropriate eigenvector elements is required that leads to separate iterations for eigenvector estimation. The eigenvector estimation, an iterative procedure in itself, may slow down the convergence of the corresponding algorithms especially when the underlying graph is not well-connected.

IV. The **AB**/**Push-Pull** Framework

All three algorithms discussed in the previous section, with doubly stochastic, column stochastic, and row stochastic weights, converge sub-linearly with decaying step-sizes, i.e., with $\alpha_k \to 0$. With a constant step-size, these methods converge linearly albeit to a sub-optimal solution. The reason for this sub-optimality is that $\mathbf{x}^*$ is not a fixed point when the step-size is a constant, as we explain in the following. Consider (8) with a constant step-size ($\alpha_k = \alpha, \forall k$) and with $\mathbf{x}_k^i = \mathbf{x}^*, \forall i$ and some k, then

$$\mathbf{x}_{k+1}^i = \sum_{r \in \mathcal{N}_i^{\text{in}}} w_{ir}\mathbf{x}^* - \alpha \cdot \nabla f_i(\mathbf{x}^*) = \mathbf{x}^* - \alpha \cdot \nabla f_i(\mathbf{x}^*) \neq \mathbf{x}^*,$$

because $\nabla f_i(\mathbf{x}^*) \neq \mathbf{0}_p$, in general. Recall that $\mathbf{x}^*$ minimizes $\sum_{i=1}^n f_i$ and thus $\sum_{i=1}^n \nabla f_i(\mathbf{x}^*) = \mathbf{0}_p$ holds for the sum but not necessarily for the component functions. Clearly, this issue disappears if ∇f_i at each node is replaced by ∇F but that is not possible since f_i's are distributed and private.

A recently introduced scheme that is referred to as *gradient tracking* overcomes the aforementioned steady-state error while keeping a constant step-size by replacing ∇f_i, at each node i, with an estimate of the global gradient ∇F [17], [18], [27], [28], [91]–[93]. Formally, a new variable $\mathbf{y}_k^i \in \mathbb{R}^p$ tracks $\sum_{i=1}^n \nabla f_i(\mathbf{x}_k^i)$ and the estimate $\mathbf{x}_k^i$ descends in the direction of $\mathbf{y}_k^i$. The task is thus to design an algorithm that tracks a time-varying function $\sum_{i=1}^n \nabla f_i(\mathbf{x}_k^i)$ such that its components are distributed over the network. To this aim, Dynamic Average Consensus (DAC) [94] is used that tracks the sum $\sum_{i=1}^n \mathbf{r}_k^i$ of time-varying functions $\mathbf{r}_k^i$, assuming that each $\mathbf{r}_k^i$ approaches a constant.

The resulting algorithm, Decentralized Gradient Descent with Gradient Tracking (**GT-DGD**), linearly converges to the global minimum $\mathbf{x}^*$ when each $f_i \in \mathcal{S}_{\mu,\ell}$, **GT-DGD** (16a)–(16b). **GT-DGD** is given by the following iterations: at each node i and $\forall k \geq 0$,

$$\mathbf{x}_{k+1}^i = \sum_{r \in \mathcal{N}_i^{\text{in}}} w_{ir}\mathbf{x}_k^r - \alpha \cdot \mathbf{y}_k^i, \tag{16a}$$

$$\mathbf{y}_{k+1}^i = \sum_{r \in \mathcal{N}_i^{\text{in}}} w_{ir}\mathbf{y}_k^r + \nabla f_i(\mathbf{x}_{k+1}^i) - \nabla f_i(\mathbf{x}_k^i), \tag{16b}$$

where $\mathbf{x}_0^i$'s in $\mathbb{R}^p$ are arbitrary and $\mathbf{y}_0^i = \nabla f_i(\mathbf{x}_0^i), \forall i$. Here, the fusion weights $\{w_{ir}\}$ are chosen such that $\underline{W} =$

$\{w_{ir}\}$ is primitive and doubly stochastic and thus **GT-DGD** is only applicable to connected undirected (or balanced directed) graphs. Intuitively, if we assume that $\mathbf{x}_k^i \rightarrow \mathbf{x}^*$ at each i, then $\sum_{i=1}^n \nabla f_i(\mathbf{x}_k^i)$ approaches a constant and we have that $\mathbf{y}_k^i \rightarrow \sum_{i=1}^n \nabla f_i(\mathbf{x}_k^i)$ at each node i. Thus, the iterates obtained via equation (16a) asymptotically descend in the direction of global minimum. However, **GT-DGD** is restricted to undirected graphs because the weights $\underline{W}$ are doubly stochastic. Here, we recall that the row stochasticity of $\underline{W}$ leads to agreement, while the column stochasticity leads to optimality. Although row stochasticity and column stochasticity are both required, we may ask whether they must hold simultaneously for the weights in (16a) and (16b).

In particular, (16a) may be implemented with row stochastic weights (that are not necessarily column stochastic) leading to an agreement, while (16b) may be implemented with column stochastic weights (that are not necessarily row stochastic) ensuring optimality. This observation leads to the **AB/Push-Pull** algorithm[4] described formally in Algorithm 1, written with the help of row and column stochastic weights, $\underline{A} = \{a_{ir}\}$a and $\underline{B} = \{b_{ir}\}$, respectively. Since the implementation of **AB/Push-Pull** does not requite doubly stochastic weights, the algorithm is applicable to both directed and undirected graphs. When each $f_i \in \mathcal{S}_{\mu,\ell}$, **AB/Push-Pull** converges linearly to the global minimum $\mathbf{x}^*$ of F [3], [4].

Algorithm 1 **AB/Push-Pull**: At each node i

```
Require: $\mathbf{x}_0^i \in \mathbb{R}^p, \mathbf{y}_0^i = \nabla f_i(\mathbf{x}_0^i), \alpha > 0$.
1: for $k = 0, 1, 2, \cdots$ do
2:     State update: $\mathbf{x}_{k+1}^i = \sum_{r \in \mathcal{N}_i^{\text{in}}} a_{ir}\mathbf{x}_k^r - \alpha \cdot \mathbf{y}_k^i$
3:     Gradient tracking update:
       $\mathbf{y}_{k+1}^i = \sum_{r \in \mathcal{N}_i^{\text{in}}} b_{ir}\mathbf{y}_k^r + \nabla f_i(\mathbf{x}_{k+1}^i) - \nabla f_i(\mathbf{x}_k^i)$
4: end for
```

Remark 6 (Uncoordinated Step-sizes): The **AB/Push-Pull** algorithm is applicable to the case when each node i chooses a distinct step-size α_i. Convergence requires the maximum step-size $\max_i \alpha_i$ to be positive and sufficiently small while all other step-sizes can in fact be chosen as 0; see [30], [95], [96] for additional details. For the sake of argument, assume that node 1 in an n-node network chooses an appropriate step-size $\alpha_1 > 0$, while $\alpha_i = 0$, for $i = 2, \ldots, n$, then node 1 implements a descent in its state update while all other nodes implement only the average-consensus part. The descent direction $\mathbf{y}_k^1$ at node 1 however comes from mixing information among all nodes via the gradient tracking update and $\mathbf{x}_k^1$ at node 1 thus asymptotically descends in the global direction. Because the rest of the network updates $\mathbf{x}_k^i, i = 2, \ldots, n$ without the gradient correction, we observe that the overall scheme operates in a leader-follower mode where node 1 acts as the leader while the remaining follower nodes asymptotically converge to the leader state; see [97] for details on the leader-follower algorithm.

A. Analysis

In this section, we briefly describe the main ideas to establish the linear convergence of **AB/Push-Pull**. To proceed, we write **AB/Push-Pull** in a compact form as follows:

$$\mathbf{x}_{k+1} = A\mathbf{x}_k - \alpha\mathbf{y}_k, \tag{17}$$
$$\mathbf{y}_{k+1} = B\mathbf{y}_k + \nabla\mathbf{f}(\mathbf{x}_{k+1}) - \nabla\mathbf{f}(\mathbf{x}_k), \tag{18}$$

where $A = \underline{A} \otimes I_p$, $B = \underline{B} \otimes I_p$, and the global vectors $\mathbf{x}_k, \mathbf{y}_k$, and $\nabla\mathbf{f}(\mathbf{x}_k)$ concatenate the local vectors $\mathbf{x}_k^i$'s, $\mathbf{y}_k^i$'s, and $\nabla f_i(\mathbf{x}_k^i)$'s, respectively. Following **DGD**, we may show that $\mathbf{x}_k \rightarrow \mathbf{1}_n \otimes \overline{\mathbf{x}}_k$, where $\overline{\mathbf{x}}_k$ is some weighted network average, and $\overline{\mathbf{x}}_k \rightarrow \mathbf{x}^*$. However, there are two issues here. First, the weight matrices are not doubly stochastic and hence contraction in the Euclidean norm is not applicable. In particular, $\|W - W^\infty\|_2 < 1$ for a primitive doubly stochastic matrix but not necessarily for a row or a column stochastic matrix. Second, the descend is in the direction of $\mathbf{y}_k$, and $\mathbf{x}_k$ is coupled to the gradients via $\mathbf{y}_k$. The formal analysis thus requires an alternate approach and is described in the following.

Step 1–Contracting norms: Our approach to establish convergence of **AB/Push-Pull** is to first find contracting norms for the row and column stochastic matrices. In this context, note that since both $\underline{A}$ and $\underline{B}$ are primitive and stochastic, we use their non-$\mathbf{1}_n$ eigenvectors (corresponding to eigenvalue 1), $\boldsymbol{\pi}_A > 0$ and $\boldsymbol{\pi}_B > 0$, respectively, to define weighted Euclidean norms as follows: $\forall \mathbf{x} \in \mathbb{R}^n$,

$$\|\mathbf{x}\|_{\boldsymbol{\pi}_A} := \sqrt{\pi_A^1 x_1^2 + \cdots + \pi_A^n x_n^2} = \|\text{diag}(\sqrt{\boldsymbol{\pi}_A})\mathbf{x}\|_2\,,$$
$$\|\mathbf{x}\|_{\boldsymbol{\pi}_B} := \sqrt{\tfrac{1}{\pi_B^1} x_1^2 + \cdots + \tfrac{1}{\pi_B^n} x_n^2} = \left\|\text{diag}(\sqrt{\boldsymbol{\pi}_B})^{-1}\mathbf{x}\right\|_2.$$

Subsequently, we denote $|||\cdot|||_{\boldsymbol{\pi}_A}$ and $|||\cdot|||_{\boldsymbol{\pi}_B}$ as the matrix norms induced by $\|\cdot\|_{\boldsymbol{\pi}_A}$ and $\|\cdot\|_{\boldsymbol{\pi}_B}$, respectively [81], i.e.,

$$|||\,X\,|||_{\boldsymbol{\pi}_A} = \left|\left|\left|\,\text{diag}(\sqrt{\boldsymbol{\pi}_A})X\ \text{diag}(\sqrt{\boldsymbol{\pi}_A})^{-1}\,\right|\right|\right|_2,$$
$$|||\,X\,|||_{\boldsymbol{\pi}_B} = \left|\left|\left|\,\text{diag}(\sqrt{\boldsymbol{\pi}_B})^{-1}X\ \text{diag}(\sqrt{\boldsymbol{\pi}_B})\,\right|\right|\right|_2,$$

$\forall X \in \mathbb{R}^{n\times n}$. With the help of these induced matrix norms, it can be shown that [98]

$$\sigma_A := |||\,A - A^\infty\,|||_{\boldsymbol{\pi}_A \otimes \mathbf{1}_p} < 1,$$
$$\sigma_B := |||\,B - B^\infty\,|||_{\boldsymbol{\pi}_B \otimes \mathbf{1}_p} < 1.$$

Step 2–Establish an LTI error dynamics: With the help of the aforementioned contracting norms for the two weight matrices A and B, we next describe the three errors embedded in **AB/Push-Pull** in their respective norms:

(i) the agreement error $\|\mathbf{x}_k - A^\infty\mathbf{x}_k\|_{\boldsymbol{\pi}_A \otimes I_p}$;
(ii) the optimality gap $\|A^\infty\mathbf{x}_k - \mathbf{1}_n \otimes \mathbf{x}^*\|_2$; and,
(iii) the gradient tracking error $\|\mathbf{y}_k - B^\infty\mathbf{y}_k\|_{\boldsymbol{\pi}_B \otimes I_p}$.

These errors can be written in a vector

$$\mathbf{t}_k = \begin{bmatrix} \|\mathbf{x}_k - A^\infty\mathbf{x}_k\|_{\boldsymbol{\pi}_A \otimes I_p} \\ \|A^\infty\mathbf{x}_k - \mathbf{1}_n \otimes \mathbf{x}^*\|_2 \\ \|\mathbf{y}_k - B^\infty\mathbf{y}_k\|_{\boldsymbol{\pi}_B \otimes I_p} \end{bmatrix}$$

that follows:

$$\mathbf{t}_{k+1} \leq J_\alpha \mathbf{t}_k, \tag{19}$$

where

$$J_\alpha = \begin{bmatrix} \sigma_A & 0 & 0 \\ 0 & 1 & 0 \\ a_6 & 0 & \sigma_B \end{bmatrix} + \alpha \begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & -n\mu(\boldsymbol{\pi}_A^\top\boldsymbol{\pi}_B) & a_6 \\ a_7 & a_8 & a_9 \end{bmatrix},$$

for some constants $a_1, \ldots, a_9$; see [3], [4] for details.

[4]Algorithm 1 appeared simultaneously as **AB** in [3] and as *push-pull* in [4]; see Section IV-B for push and pull aspects of the underlying communication.

Step 3–Linear convergence: Clearly, if $\mathbf{t}_k \to \mathbf{0}_3$, then $\mathbf{x}_k^i$'s reach agreement and the agreement is on the global minimum $\mathbf{x}^*$ of F, while the rate at which $\mathbf{t}_k \to \mathbf{0}_3$ establishes the convergence rate of **AB/Push-Pull**. Given that $\mathbf{t}_k$ follows the LTI system in (19), linear convergence off **AB/Push-Pull** to $\mathbf{x}^*$ can be established by showing that the spectral radius $\rho(J_\alpha)$ of J_α is less than one. There are several ways to obtain $\rho(J_\alpha) < 1$ for the matrix J_α given above and we present the argument from [3]. First note that when $\alpha = 0$, J_0 has three eigenvalues: $\sigma_A, \sigma_B, 1$, two of which are strictly less than 1 as given by Step 1. Next, since the eigenvalues of a matrix are a continuous function of its elements, it can be shown that as the step-size α increases, the eigenvalue of 1 strictly decreases. In particular, denote by $q(\alpha)$ the eigenvalues of J_α as a function of α, we have that

$$\left.\frac{d\,q(\alpha)}{d\alpha}\right|_{\alpha=0,q=1} = -n\mu(\boldsymbol{\pi}_A^\top \boldsymbol{\pi}_B),$$

which is strictly negative because $\boldsymbol{\pi}_A$ and $\boldsymbol{\pi}_B$ are both strictly positive, where n is the number of nodes and $\mu > 0$ is the strong convexity constant of F. In other words, the eigenvalue $q(\alpha) = 1$ decreases as a function of α and $\rho(J_\alpha) < 1$ for a sufficiently small $\alpha > 0$.

B. Architectures

The **AB/Push-Pull** algorithm unifies different types of decentralized and distributed architectures, including decentralized, centralized, and semi-centralized architecture [4], [95]. To illustrate, let graphs $\mathcal{G}_{\underline{A}}$ and $\mathcal{G}_{\underline{B}}$ be induced by the two matrices $\underline{A}$ and $\underline{B}$, respectively, i.e., $a_{ir} > 0$ (resp. $b_{ir} > 0$) if and only if there exists a link from node r to node i in graph $\mathcal{G}_{\underline{A}}$ (reps. $\mathcal{G}_{\underline{B}}$). Note that when implementing the **AB/Push-Pull** algorithm, we have the flexibility to design two different graphs $\mathcal{G}_{\underline{A}}$ and $\mathcal{G}_{\underline{B}}$, rather than restricting to using one single graph that is commonly considered in the distributed optimization literature (see e.g. [27]). Such flexibility is key to the unifying property of the **AB/Push-Pull** algorithm. Formally, we require the following condition on the graphs $\mathcal{G}_{\underline{A}}$ and $\mathcal{G}_{\underline{B}^\top}$ (the graph $\mathcal{G}_{\underline{B}^\top}$ is the graph $\mathcal{G}_{\underline{B}}$ with all its edges reversed).

Assumption 1: The graphs $\mathcal{G}_{\underline{A}}$ and $\mathcal{G}_{\underline{B}^\top}$ each contain at least one spanning tree. Moreover, there exists at least one node that is a root of spanning trees for both $\mathcal{G}_{\underline{A}}$ and $\mathcal{G}_{\underline{B}^\top}$.[5]

In light of the above condition, we explain how **AB/Push-Pull** unifies different types of distributed architecture using examples from [4]. First, for the fully decentralized architecture, suppose we have a directed and strongly connected graph $\mathcal{G}$. We can let $\mathcal{G}_{\underline{A}} = \mathcal{G}_{\underline{B}} = \mathcal{G}$ and design the weights for $\underline{A}$ and $\underline{B}$ accordingly. Then we have a typical peer-to-peer network structure which is fully decentralized.

For the case of (semi)-centralized architectures, which is less straightforward, we consider a simple, four-node star network $\mathcal{G}$, as shown in Fig. 2. Clearly $\mathcal{G}_{\underline{A}}$ and $\mathcal{G}_{\underline{B}^\top}$ are identical spanning tree with node 1 being their common root. Under

[5]This condition is weaker than assuming both $\mathcal{G}_{\underline{A}}$ and $\mathcal{G}_{\underline{B}}$ are strongly connected but it requires some restrictions on graph topology. Note however that the union of $\mathcal{G}_{\underline{A}}$ and $\mathcal{G}_{\underline{B}}$ is still required to be strongly connected, which guarantees the essential information exchange among the nodes.

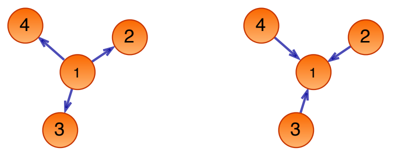


Fig. 2. On the left is the graph $\mathcal{G}_{\underline{A}}$ and on the right is the graph $\mathcal{G}_{\underline{B}}$.

this setting, the weight matrices $\underline{A}$ and $\underline{B}$ in **AB/Push-Pull** can be chosen as

$$\underline{A} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0.5 & 0.5 & 0 & 0 \\ 0.5 & 0 & 0.5 & 0 \\ 0.5 & 0 & 0 & 0.5 \end{bmatrix}, \quad \underline{B} = \begin{bmatrix} 1 & 0.5 & 0.5 & 0.5 \\ 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0.5 \end{bmatrix}.$$

It can be seen from Fig. 2 that the central node 1's information regarding $\mathbf{x}_k^1$ is pulled by the entire network through $\mathcal{G}_{\underline{A}}$; the other nodes only passively receive and use node 1's information. Meanwhile, node 1 has been pushed information regarding $\mathbf{y}_k^i$ ($i = 2, 3, 4$) from nodes 2, 3, and 4 through $\mathcal{G}_{\underline{B}}$; the other nodes only actively comply with the request from node 1. This motivates the algorithm's name ***Push-Pull*** in [4], [95]. Although nodes 2, 3, and 4 are updating $\mathbf{y}_k^i$'s accordingly, they do not have to contribute to the optimization process: due to the weights in the last three rows of $\underline{B}$, the values of $\mathbf{y}_k^i$'s for nodes 2, 3, and 4 will decrease to 0 geometrically fast. As a result, in this special case, the local stepsize α for nodes 2, 3, and 4 can be set to 0. Without loss of generality, suppose $f_1(\mathbf{x}) = 0, \forall \mathbf{x} \in \mathbb{R}^p$. Then the **AB/Push-Pull** algorithm represents a typical centralized gradient method for minimizing $\sum_{i=2}^4 f_i(x)$ where the master node 1 utilizes the slave nodes 2, 3, and 4 to compute the gradient information in a distributed fashion.

Taking the above as a toy example for illustrating the centralized architecture, note that node 1 can be replaced by a strongly connected subnet in $\mathcal{G}_{\underline{A}}$ and $\mathcal{G}_{\underline{B}}$, respectively. Similarly, all the other nodes can be replaced by subnets as long as the information from the master layer in these subnets can be diffused to all the slave layer nodes in $\mathcal{G}_{\underline{A}}$, and the information from all the slave layer nodes can be diffused to the master layer in $\mathcal{G}_{\underline{B}}$. The concept of rooted trees can be used to understand the specific requirements on connectivities of slave subnets. In general, the nodes are referred to as leaders if their roles in the network are similar to the role of node 1 in Fig. 2; and the other nodes are referred to as followers. After replacing the individual nodes by subnets, all the subnets have decentralized network structures, while the leader subnet and the follower subnets form a master-slave relationship. This is why such an architecture is called semi-centralized.

C. Acceleration

Given the simplicity of **AB/Push-Pull**, it is natural to consider momentum-based improvements. One immediate extension, based on the Polyak's heavy-ball method [82], is **ABm** [96] that is given by the following equations:

$$\mathbf{x}_{k+1}^i = \sum_{r\in\mathcal{N}_i^{\text{in}}} a_{ir}\mathbf{x}_k^r - \alpha \cdot \mathbf{y}_k^i + \beta \cdot (\mathbf{x}_k^i - \mathbf{x}_{k-1}^i), \tag{20a}$$

$$\mathbf{y}_{k+1}^i = \sum_{r\in\mathcal{N}_i^{\text{in}}} b_{ir}\mathbf{y}_k^r + \nabla f_i(\mathbf{x}_{k+1}^i) - \nabla f_i(\mathbf{x}_k^i), \tag{20b}$$

where $\mathbf{x}_k^i - \mathbf{x}_{k-1}^i$ is the heavy-ball momentum added at each node, α and β are the step-size and momentum parameters, respectively. It is shown in [96] that **ABm** converges linearly for sufficiently small step-size and momentum parameters, however, acceleration over **AB** is only shown numerically and a detailed theoretical analysis remains an open problem. Alternatively, the use of Nesterov's momentum [80] is also studied in the context of **AB/Push-Pull**. The corresponding algorithm **ABN** [99] is given by the following equations:

$$\mathbf{s}_{k+1}^i = \sum_{r\in\mathcal{N}_i^{\text{in}}} a_{ir}\mathbf{x}_k^r - \alpha\cdot\mathbf{y}_k^i, \tag{21a}$$

$$\mathbf{x}_{k+1}^i = \mathbf{s}_{k+1}^i + \beta_k\cdot(\mathbf{s}_{k+1}^i - \mathbf{s}_k^i), \tag{21b}$$

$$\mathbf{y}_{k+1}^i = \sum_{r\in\mathcal{N}_i^{\text{in}}} b_{ir}\mathbf{y}_k^r + \nabla f_i\big(\mathbf{x}_{k+1}^i\big) - \nabla f_i\big(\mathbf{x}_k^i\big), \tag{21c}$$

where β_k is the momentum parameter. When the set of weights above are both doubly stochastic, a closely related form of **ABN** is studied in [100] where acceleration is analytically shown. The analysis of **ABN** however remains an open problem and acceleration is only shown numerically in [99].

V. AB/Push-Pull: A general framework

We now cast **AB/Push-Pull** as a general framework that unifies much of the existing work on decentralized first-order methods with gradient tracking. Clearly, **GT-DGD** (16a)–(16b) is a special case of **AB/Push-Pull** when both sets of weights are further doubly stochastic. We note that **DGD** also follows from **AB/Push-Pull** when A is replaced by a doubly stochastic matrix W and B is chosen as identity I_{np}. Below, we relate **AB/Push-Pull** to other well-known algorithms in the literature; see [96] for additional discussion.

A. *GT-DGD with column stochastic weights*

We recall **AB/Push-Pull** written compactly in (17)–(18), where $A = \underline{A}\otimes I_p$ is row stochastic $B = \underline{B}\otimes I_p$ is column stochastic, while both A and B are in addition primitive. Define Π_A to be a diagonal matrix with $\boldsymbol{\pi}_A\otimes\mathbf{1}_p$ on its main diagonal where $\boldsymbol{\pi}_A^\top\underline{A} = \boldsymbol{\pi}_A^\top$ and $\underline{A}\mathbf{1}_n = \mathbf{1}_n$. Clearly, we have $\Pi_A = \text{diag}(\boldsymbol{\pi}_A)\otimes I_p$. We first note that $\Pi_A A\Pi_A^{-1}$ is column stochastic, i.e.,

$$\mathbf{1}_{np}^\top\cdot\Pi_A A\Pi_A^{-1} = \mathbf{1}_{np}^\top(\text{diag}(\boldsymbol{\pi}_A)\otimes I_p)(\underline{A}\otimes I_p)\Pi_A^{-1} = \mathbf{1}_{np}^\top.$$

Similarly, it can be verified that $\Pi_A A\Pi_A^{-1}\cdot\boldsymbol{\pi}_A\otimes\mathbf{1}_p = \boldsymbol{\pi}_A\otimes\mathbf{1}_p$. With the help of the invertible matrix Π_A, we define a state transformation $\widetilde{\mathbf{x}}_k = \Pi_A\mathbf{x}_k$ and use (17)–(18) to obtain

$$\widetilde{\mathbf{x}}_{k+1} = \widetilde{B}\widetilde{\mathbf{x}}_k - \alpha\Pi_A\cdot\mathbf{y}_k, \tag{22}$$

$$\mathbf{y}_{k+1} = B\mathbf{y}_k + \nabla\mathbf{f}(\Pi_A^{-1}\widetilde{\mathbf{x}}_{k+1}) - \mathbf{f}(\Pi_A^{-1}\widetilde{\mathbf{x}}_k), \tag{23}$$

where $\widetilde{B} := \Pi_A A\Pi_A^{-1}$ is column stochastic as we noted earlier. This state transformation shows that a decentralized optimization algorithm with only column stochastic weights (B and $\widetilde{B}$) is naturally embedded in **AB/Push-Pull**. However, such an algorithm additionally requires the eigenvector vector $\boldsymbol{\pi}_A$ of the weights $\widetilde{B}$ in (22) to implement (23). Since $\boldsymbol{\pi}_A^\top$ is not locally known to any node, implementing (22) and (23) require estimating this non-$\mathbf{1}_n$ right eigenvector of the column stochastic $\widetilde{B}$; clearly not required in **AB/Push-Pull** because of the row stochasticity of A. The resulting decentralized algorithm at each node i is given by

$$\mathbf{x}_{k+1}^i = \sum_{r\in\mathcal{N}_i^{\text{in}}} \widetilde{b}_{ir}\mathbf{x}_k^r - \alpha\cdot\mathbf{y}_k^i, \tag{24a}$$

$$z_{k+1}^i = \sum_{r\in\mathcal{N}_i^{\text{in}}} \widetilde{b}_{ir}z_k^r, \tag{24b}$$

$$\mathbf{y}_{k+1}^i = \sum_{r\in\mathcal{N}_i^{\text{in}}} b_{ir}\mathbf{y}_k^r + \nabla f_i\Big(\frac{\mathbf{x}_{k+1}^i}{z_{k+1}^i}\Big) - \nabla f_i\Big(\frac{\mathbf{x}_k^i}{z_k^i}\Big), \tag{24c}$$

where $\{\widetilde{b}_{ir}\}$ and $\{b_{ir}\}$ are (possibly different) column stochastic weights; the algorithm is initialized with an arbitrary $\mathbf{x}_0^i\in\mathbb{R}^p$ and $\mathbf{y}_0^i = \nabla f_i(\mathbf{x}_0^i), z_0^i = 1$. The above algorithm essentially adds gradient tracking to gradient-push [23], [88], [89] described earlier in (12a)–(12c).

Remark 7: The algorithm in (24a)–(24c) is well-known as **ADDOPT** [28] and **Push-DIGing** [27], both of which add push-sum consensus to **GT-DGD** implemented over directed graphs with column stochastic weights. It is easy to see that the general idea of push-sum consensus, explored over the **DGD** framework in [23], [88], [89] or over the **GT-DGD** framework as in **ADDDOPT/Push-DIGing** [27], [28], essentially results from a state transformation in **AB/Push-Pull**.

B. *GT-DGD with row stochastic weights*

Following the previous discussion, we now perform a state transformation on the $\mathbf{y}_k$-update such that the column stochastic weight matrix B in **AB/Push-Pull** transforms into a row stochastic matrix. Defining $\Pi_B = \text{diag}(\boldsymbol{\pi}_B)\otimes I_p$, it is easy to verify that $\widetilde{A} := \Pi_B^{-1}B\Pi_B = \{\widetilde{a}_{ir}\}$ is row stochastic, i.e.,

$$\Pi_B^{-1}B\Pi_B\mathbf{1}_{np} = \mathbf{1}_{np},\quad (\boldsymbol{\pi}_B^\top\otimes\mathbf{1}_p)\Pi_B^{-1}B\Pi_B \quad = \boldsymbol{\pi}_B^\top\otimes\mathbf{1}_p.$$

The applicable state transformation $\widetilde{\mathbf{y}}_k := \Pi_B^{-1}\mathbf{y}_k$ leads to

$$\mathbf{x}_{k+1}^i = \sum_{r\in\mathcal{N}_i^{\text{in}}} a_{ir}\mathbf{x}_k^r - \alpha\cdot\widetilde{\mathbf{y}}_k^i, \tag{25a}$$

$$\mathbf{e}_{k+1}^i = \sum_{r\in\mathcal{N}_i^{\text{in}}} \widetilde{a}_{ir}\mathbf{e}_k^r, \tag{25b}$$

$$\widetilde{\mathbf{y}}_{k+1}^i = \sum_{r\in\mathcal{N}_i^{\text{in}}} \widetilde{a}_{ir}\mathbf{y}_k^r + \frac{\nabla f_i(\mathbf{x}_{k+1}^i)}{[\mathbf{e}_{k+1}^i]_i} - \frac{\nabla f_i(\mathbf{x}_k^i)}{[\mathbf{e}_k^i]_i}, \tag{25c}$$

after adding eigenvector estimation to estimate the non-$\mathbf{1}_n$ eigenvector vector of $\underline{B}$ corresponding to the eigenvalue 1, where $[\mathbf{e}_k^i]_i$ is the ith element of $\mathbf{e}_k^i\in\mathbb{R}^n$ at node i. The algorithm is initialized with an arbitrary $\mathbf{x}_0^i\in\mathbb{R}^p$ and $\mathbf{y}_0^i = \nabla f_i(\mathbf{x}_0^i)$, while $\mathbf{e}_0^i\in\mathbb{R}^n$ is a vector of zeros with a one at the ith location. The resulting algorithm is formally studied in [29], [30] as **FROST** and is a gradient tracking extension of **DGD** with row stochastic weights [90], described earlier in (15a)–(15b).

C. *Average-consensus*

We now interpret **AB/Push-Pull** only for consensus problems and show that it subsumes average-consensus over strongly connected graphs as a special case. To show this, we choose the cost function at each node i as

$$f_i(\mathbf{x}) = \tfrac{1}{2}\|\mathbf{x} - \boldsymbol{\upsilon}_i\|^2, \tag{26}$$

for some $\boldsymbol{v}_i \in \mathbb{R}^p$. Clearly, the minimum of $F = \sum_{i=1}^n f_i$ is achieved at $\mathbf{x}^* = \frac{1}{n}\sum_{i=1}^n \boldsymbol{v}_i$. Thus, **AB/Push-Pull** naturally leads to the following average-consensus algorithm by noting that $\nabla\mathbf{f}(\mathbf{x}_{k+1}) - \nabla\mathbf{f}(\mathbf{x}_k) = \mathbf{x}_{k+1} - \mathbf{x}_k$ for the local functions described in (26):

$$\begin{bmatrix} \mathbf{x}_{k+1} \\ \mathbf{y}_{k+1} \end{bmatrix} = \begin{bmatrix} A & -\alpha I_{np} \\ A - I_{np} & B - \alpha I_{np} \end{bmatrix} \begin{bmatrix} \mathbf{x}_k \\ \mathbf{y}_k \end{bmatrix}, \qquad (27)$$

where $\mathbf{x}_0^i = \boldsymbol{v}_i$ and $\mathbf{y}_i^0 = 0,\ \forall i$. From the linear convergence of **AB/Push-Pull**, as shown in Section IV-A, it follows that the above equations converge linearly to the average of $\boldsymbol{v}_i$'s. What is surprising is that the above form is closely related to a well-known algorithm for average-consensus over directed graphs known as *surplus consensus* [34]. In particular, surplus consensus is obtained from the above equations after a state transformation with diag$\,(I_{np}, -I_{np})$.

Remark 8 (Consensus over arbitrary graphs): Following the discussion here, choosing the local functions f_i's as (26) in **GT-DGD** [17], [18], or in **ADDOPT/Push-DIGing** [27], [28], or in **FROST** [29], [30], leads to average-consensus, that utilizes gradient tracking, with only doubly stochastic, column stochastic, or row stochastic weights. The protocol that results directly from **AB/Push-Pull** is surplus consensus, while the one resulting from **FROST** has been considered in [33]. Following the state transformations of Sections V-A and V-B, it is straightforward to verify that the algorithm in [33] is in fact related to surplus consensus after a state transformation.

Remark 9: Recall that **DGD** (8) adds a gradient correction to consensus with doubly stochastic weights and is applicable to undirected graphs. Its extension, **GP** (12a)–(12c), to directed graphs adds gradient correction to push-sum consensus [23], [88], [89]. Similarly, we may use (27), or equivalently surplus consensus [34], as the consensus layer and add a gradient correction to enable decentralized optimization over directed graphs. This idea was explored in [24], [25], however, the convergence rate restrictions are similar to that of **DGD**.

VI. Decentralized Stochastic Optimization

We now discuss stochastic first-order methods in Section VI and VII with the help of local cost functions with a finite-sum structure of (1). The discussion in Section VI, in particular, easily extends to the case when local functions do not have the finite sum structure; see Remark 14.

Recall from (1) that each node i possesses a total of m_i data samples leading to m_i component functions $\{f_{i,j_i}\}$'s in the local cost f_i. The protocols discussed so far are batch operations, i.e., they use the entire local f_i and thus all local data samples to compute the descent direction $\sum_{j_i=1}^{m_i} \nabla f_{i,j_i}$, at each node i. Computing m_i gradients at each node i can be taxing particularly when the data is high-dimensional and the cost functions are non-trivial. Computationally-efficient schemes thus rely on stochastic policies to randomly select a small subset of data from the local batch, over which the descent direction is computed. The centralized algorithm based on this idea is the well-known Stochastic Gradient Descent (SGD); see [11] for a comprehensive review. Assuming that the data is not distributed ($n = 1$), and the goal is to minimize $F = \frac{1}{m}\sum_{j=1}^m f_j$, the centralized **SGD** is given by

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \cdot \nabla f_{\tau_k}(\mathbf{x}_k), \qquad \forall k \geq 0, \qquad (28)$$

where τ_k is an index chosen uniformly at random from the set $\{1, \ldots, m\}$ at each k, and m is the total number of component functions.

It can be verified that ∇f_{τ_k} is an unbiased gradient estimate, i.e., $\mathbb{E}[\nabla f_{\tau_k}(\mathbf{x})] = \nabla F(\mathbf{x}), \forall \mathbf{x} \in \mathbb{R}^p$, Moreover, it is assumed that each stochastic gradient ∇f_{τ_k} has bounded variance, i.e.,

$$\mathbb{E}[\|\nabla f_{\tau_k}(\mathbf{x}_k) - \nabla F(\mathbf{x}_k)\|_2^2] < \sigma^2, \qquad \forall k \geq 0, \mathbf{x} \in \mathbb{R}^p.$$

Under an additional assumption that $F \in \mathcal{S}_{\mu,\ell}$, it can be shown that with a constant step-size $\alpha \in \left(0, \frac{1}{\ell}\right]$, the mean squared error $\mathbb{E}\left[\|\mathbf{x}_k - \mathbf{x}^*\|_2^2\right]$ decays linearly, at the rate of $(1-\mu\alpha)^k$, to a neighborhood of $\mathbf{x}^*$. Formally, we have [11],

$$\mathbb{E}\left[\|\mathbf{x}_k - \mathbf{x}^*\|_2^2\right] \leq (1-\mu\alpha)^k + \frac{\alpha\sigma^2}{\mu}, \qquad \forall k \geq 0. \qquad (29)$$

The above equation says that the optimality gap decreases to $\frac{\alpha\sigma^2}{\mu}$ at a linear rate with the exponent $1-\alpha\mu$, where μ is the strong convexity constant of the function F. Since $\alpha \in (0, \frac{1}{\ell}]$ and $\mu \leq \ell$, the exponent is always less than 1. The constant steady-state error given by $\frac{\alpha\sigma^2}{\mu}$ implies the "inexact convergence" of **SGD** due to the persistent gradient noise, i.e., the variance σ^2 does not vanish. A diminishing step-size, $\mathcal{O}(\frac{1}{k})$, overcomes this issue and leads to exact convergence albeit at a slower rate. For example, with $\alpha_k = \frac{1}{\mu(k+1)}$, we have $\forall k \geq 0$,

$$\mathbb{E}\left[\|\mathbf{x}_k - \mathbf{x}^*\|_2^2\right] \leq \frac{\max\left\{\frac{2\sigma^2}{\mu^2}, \|\mathbf{x}_0 - \mathbf{x}^*\|_2^2\right\}}{k+1}, \qquad (30)$$

which goes to zero [11] for smooth and strongly convex functions, i.e., $F \in \mathcal{S}_{\mu,\ell}$. In other words, to reach an ε-accuracy of $\mathbf{x}^*$, i.e., $\mathbb{E}\left[\|\mathbf{x}_k - \mathbf{x}^*\|^2\right] \leq \varepsilon$, the **SGD** method with decaying step-sizes requires $\mathcal{O}(\kappa^2\varepsilon^{-1})$ component gradient evaluations. Note that this rate is independent of the sample size since **SGD** only computes one gradient per iteration. In the following, we discuss the decentralized versions of **SGD**.

*Remark 10 (**GD** vs. **SGD**):* We note that gradient descent (**GD**) (7) computes m component gradients per iteration, while **SGD** (28) computes only one gradient. Due to this, **SGD** requires $\mathcal{O}(\kappa^2\varepsilon^{-1})$ component gradient evaluations to reach an ε-accuracy of the global minimum $\mathbf{x}^*$, and is independent of the size m of the data, versus $\mathcal{O}(m\kappa\ln\varepsilon^{-1})$ required by **GD** that uses the entire batch. It can be argued that **SGD** is often more preferable in the big data regimes where m is very large and low-precision solutions may suffice.

A. Decentralized Stochastic Gradient Descent (DSGD): Undirected Graphs

We now go back to Problem P0 where the data is distributed over $n > 1$ nodes and each node i possesses m_i component functions indexed by j_i. Similar to **DGD**, each node i samples from its local full batch and implements the following protocol known as **DSGD** [2], [7], [35]:

$$\mathbf{x}_{k+1}^i = \sum_{r \in \mathcal{N}_i^{\text{in}}} w_{ir}\mathbf{x}_k^r - \alpha_k \cdot \nabla f_{i,\tau_k^i}(\mathbf{x}_k), \qquad \forall k \geq 0, \quad (31)$$

where τ_k^i is a distinct index at each node i and iteration k, chosen uniformly at random from the local index set $\{1,\ldots,m_i\}$, and the weights are such that $\underline{W}=\{w_{ir}\}$ is doubly stochastic. Since the weight matrix is doubly stochastic, the underlying graphs are restricted to be undirected or balanced directed.

As in the centralized case, it is straightforward to verify that $\mathbb{E}[\nabla f_{i,\tau_k^i}(\mathbf{x})]=\nabla f_i(\mathbf{x}),\forall i$ and $\forall \mathbf{x}\in\mathbb{R}^p$, and we assume that each local stochastic gradient has a bounded variance, i.e.,

$$\mathbb{E}[\|\nabla f_{i,\tau_k^i}(\mathbf{x}_k)-\nabla f_i(\mathbf{x}_k)\|_2^2]<\sigma^2, \qquad \forall i,k\geq 0,\mathbf{x}\in\mathbb{R}^p.$$

Assuming that each local function is smooth and strongly convex, i.e., $f_i\in\mathcal{S}_{\mu,l},\forall i$, and that λ is the second largest singular value of $\underline{W}$, it can be shown that [101]: under a constant step-size, $\alpha_k=\alpha\in\left(0,\mathcal{O}\left(\frac{(1-\lambda)\mu}{\ell^2}\right)\right],\forall k$, the mean squared error $\mathbb{E}[\|\mathbf{x}_k^i-\mathbf{x}^*\|_2^2]$ decays linearly, at a rate of $(1-\mathcal{O}(\mu\alpha))^k$, to a neighborhood of $\mathbf{x}^*$ such that

$$\limsup_{k\to\infty}\frac{1}{n}\sum_{i=1}^n\mathbb{E}\left[\left\|\mathbf{x}_k^i-\mathbf{x}^*\right\|_2^2\right] = \mathcal{O}\left(\frac{\alpha\sigma^2}{n\mu}+\frac{\ell^2}{\mu^2}\frac{\alpha^2\sigma^2}{1-\lambda}+\frac{\ell^2}{\mu^2}\frac{\alpha^2 b}{(1-\lambda)^2}\right), \quad (32)$$

where $b:=\frac{1}{n}\sum_{i=1}^n\|\nabla f_i(\mathbf{x}^*)\|_2^2$. With a diminishing step-size $\alpha_k=\mathcal{O}(\frac{1}{k})$, **DSGD** achieves an exact convergence [75], [102]–[105], such that

$$\frac{1}{n}\sum_{i=1}^n\mathbb{E}\left[\left\|\mathbf{x}_k^i-\mathbf{x}^*\right\|_2^2\right]=\mathcal{O}\left(\frac{1}{k}\right), \qquad \forall k\geq 0. \quad (33)$$

DSGD for smooth and nonconvex functions is studied in [38].

*Remark 11 (**SGD** vs. **DSGD**):* With a decaying step-size, **DSGD** asymptotically achieves a network-independent convergence rate that is exactly the same as the centralized **SGD** (see [104], [105]). The network topology only affects the transient time to reach this asymptotic rate (see [104] for a precise statement). With a constant step-size in both cases, the convergence becomes linear but there is a steady-state error. The steady-state error (32) of **DSGD** depends on two additional terms as compared to **SGD** (29). Both additional terms arise from the decentralized nature of **DSGD** and include the factor $(1-\lambda)$ that encodes the connectivity of the underlying (undirected) graph, i.e., $\lambda<1$ for connected graphs and is small for well-connected graphs. However, the last term further includes the constant b that captures the average deviation between the local and global solutions. In particular, assume $\mathbf{x}_i^*$ to be the locally optimal solution such that $\nabla f_i(\mathbf{x}_i^*)=\mathbf{0}_p,\forall i$. We have $b\leq\frac{\ell}{n}\sum_i\|\mathbf{x}^*-\mathbf{x}_i^*\|^2$. Clearly, b is large when data distributions at the nodes are diverse and the local optimal is far from the global. This discrepancy motivates the use of gradient tracking technique in **DSGD** as we discuss next.

B. *DSGD with Gradient Tracking (GT-DSGD): Undirected Graphs*

As we describe in Section IV, the performance of **DGD** (8) improves with the addition of gradient tracking in **GT-DGD** (16a)-(16b). Similarly, we can add gradient tracking in **DSGD** to obtain **GT-DSGD** described as follows:

$$\mathbf{x}_{k+1}^i=\sum_{r\in\mathcal{N}_i^{\text{in}}}w_{ir}\mathbf{x}_k^r-\alpha_k\cdot\mathbf{y}_k^i, \quad (34a)$$

$$\mathbf{y}_{k+1}^i=\sum_{r\in\mathcal{N}_i^{\text{in}}}w_{ir}\mathbf{y}_k^r+\nabla f_{i,\tau_{k+1}^i}(\mathbf{x}_{k+1}^i)-\nabla f_{i,\tau_k^i}(\mathbf{x}_k^i), \quad (34b)$$

which has been recently studied in [106]–[108]. Following the same idea as in **GT-DGD**, the descend direction $\mathbf{y}_k^i$ is now estimated with the help of local *stochastic* gradients. Under the same assumptions of smoothness, strong convexity, and bounded variance as in **DSGD**, the convergence of **GT-DSGD** is summarized in the following [107]. With a constant step-size, $\alpha\in\left(0,\mathcal{O}\left(\frac{(1-\lambda)\mu}{\ell^2}\right)\right],\forall k$, $\mathbb{E}[\|\mathbf{x}_k^i-\mathbf{x}^*\|_2^2]$ decays linearly at a rate of $(1-\mathcal{O}(\mu\alpha))^k$ to a neighborhood of $\mathbf{x}^*$ such that

$$\limsup_{k\to\infty}\frac{1}{n}\sum_{i=1}^n\mathbb{E}\left[\left\|\mathbf{x}_k^i-\mathbf{x}^*\right\|_2^2\right]=\mathcal{O}\left(\frac{\alpha\sigma^2}{n\mu}+\frac{\ell^2}{\mu^2}\frac{\alpha^2\sigma^2}{(1-\lambda)^3}\right). \quad (35)$$

With a decaying step-size $\alpha_k=\mathcal{O}(\frac{1}{k})$, we have that

$$\frac{1}{n}\sum_{i=1}^n\mathbb{E}\left[\left\|\mathbf{x}_k^i-\mathbf{x}^*\right\|_2^2\right]=\mathcal{O}\left(\frac{1}{k}\right), \qquad \forall k\geq 0. \quad (36)$$

*Remark 12 (**DSGD** vs. **GT-DSGD**):* With a constant step-size, the performances of **DSGD** and **GT-DSGD** have distinct features. On the one hand, the dependence on b does not exist in **GT-DSGD**. This is intuitive because recall from Section IV that each descend direction $\mathbf{y}_k^i$ is towards an estimate of the global descend; see also Remark 11. However, since the estimate involves stochastic gradients (and not the local full batch), the persistent variance σ^2 contributes to a steady-state error. On the other hand, the steady-state error in **DSGD** has a better network dependence $\mathcal{O}((1-\lambda)^{-2})$ when compared with **GT-DSGD** $\mathcal{O}((1-\lambda)^{-3})$. In short, when the underlying communication graph is relatively well-connected, **GT-DSGD** that is independent of the difference of the local and the global optimal solutions may be preferred over **DSGD**.

C. *Decentralized Stochastic Optimization: Directed Graphs*

When the underlying communication is over arbitrary strongly connected directed graphs, we are unable to implement doubly stochastic weights $\underline{W}=\{w_{ir}\}$ in **DSGD** and **GT-DSGD**. An immediate extension is to use push-sum with **DSGD** (that uses column stochastic weights) to obtain Stochastic Gradient Push (SGP) [37], [39] that replaces the local full batch $\sum_{j_i}\nabla f_{i,j_i}$ in the algorithm described in (12a)–(12c) with a stochastic gradient [39]. A similar technique can be implemented with only row stochastic weights by writing the algorithm described in (15a)–(15b) with stochastic gradients. For the sake of brevity, we only describe the **SAB** algorithm here that is a stochastic variant of **AB/Push-Pull**. The basic algorithm is given by

$$\mathbf{x}_{k+1}^i=\sum_{r\in\mathcal{N}_i^{\text{in}}}a_{ir}\mathbf{x}_k^r-\alpha\cdot\mathbf{y}_k^i, \quad (37a)$$

$$\mathbf{y}_{k+1}^i=\sum_{r\in\mathcal{N}_i^{\text{in}}}b_{ir}\mathbf{y}_k^r+\nabla f_{i,\tau_{k+1}^i}(\mathbf{x}_{k+1}^i)-\nabla f_{i,\tau_k^i}(\mathbf{x}_k^i), \quad (37b)$$

where $\underline{A} = \{a_{ir}\}$ is row stochastic and $\underline{B} = \{b_{ir}\}$ is column stochastic. **SAB** has been recently introduced in [98] where it is shown that **SAB** converges linearly with a constant step-size. However, an explicit characterization of the rate and the analysis for the decaying step-sizes remain open problems.

*Remark 13 (Generalization of **SAB**):* Clearly, following the discussion in Section V, one can develop stochastic extensions of **ADDOPT/Push-DIGing** [27], [28] and **FROST** [30], where the former only use column stochastic weights while the latter only uses row stochastic weights. These extensions rely on eigenvector estimation as described before. Over directed graphs, these variants have the promise of improved performance when compared to the Stochastic Gradient Push [39] because of the additional gradient tracking component. However, a formal analysis of **SAB** and the extensions based on row- and column stochastic weights remain open problems.

Remark 14 (Stochastic first-order oracle): All methods discussed in this Section, i.e., **SGD**, **DSGD**, **GT-DSGD**, **SAB**, and their variants, are applicable to more general expected risk minimization problems, which do not have the finite-sum structure (1) on the local costs f_i's. In particular, each local stochastic gradient can be replaced with a noisy sample of the true gradient $\nabla f_i(\mathbf{x}_k^i)$, instead of a randomly chosen component function $\nabla f_{\tau_k^i}(\mathbf{x}_k^i)$. Formally, it is assumed that each node is able to call a Stochastic First-order Oracle (SFO), i.e., at each iteration k and node i, given $\mathbf{x}_k^i \in \mathbb{R}^p$ as the input, the SFO returns a stochastic gradient in the form of $\mathbf{g}_i(\mathbf{x}_k^i, \xi_k^i) \in \mathbb{R}^p$, where ξ_k^i's are random vectors, $\forall i, k \geq 0$. The stochastic gradients satisfy the following assumptions: The set of random vectors $\{\xi_k^i\}_{k\geq 0, i\in\mathcal{V}}$ are independent of each other, and

(1) $\mathbb{E}_{\xi_k^i}\left[\mathbf{g}_i(\mathbf{x}_k^i, \xi_k^i) | \mathbf{x}_k^i\right] = \nabla f_i(\mathbf{x}_k^i)$,
(2) $\mathbb{E}_{\xi_k^i}\left[\left\|\mathbf{g}_i(\mathbf{x}_k^i, \xi_k^i) - \nabla f_i(\mathbf{x}_k^i)\right\|_2^2 | \mathbf{x}_k^i\right] \leq \sigma^2$.

See [11], [109], [110] for additional details and discussion[6].

VII. Decentralized Stochastic First-Order Methods with Variance Reduction

This section focuses on problems where the local costs f_i's have a finite-sum structure (1). In the centralized settings, the corresponding problem $\min F = \frac{1}{m}\sum_j f_j$, i.e., Problem P0 with one node ($n = 1$), has been a topic of significant research recently. **SGD** (28) that has served as a promising solution of such problems converges with a steady-state error as shown in (29) with a constant step-size. This steady state error is explicitly given by $\frac{\alpha\sigma^2}{\mu}$ and is a consequence of the variance σ^2 of the stochastic gradient. Since the variance is persistent, a natural way to avoid it is to replace the stochastic gradient in (28) with a more refined estimator of the full batch gradient. Various variance-reduced methods are developed in this context and have a key property that the variance of the gradient estimator goes to zero asymptotically.

A popular variance reduction method is **SAGA** [44] that replaces the descent direction ∇f_{τ_k} in **SGD** (28) with with the following estimator of the batch gradient [44], i.e.,

$$\mathbf{g}_k = \nabla f_{\tau_k}(\mathbf{x}_k) - \nabla f_{\tau_k}(\widehat{\mathbf{x}}_{\tau_k}) + \frac{1}{m}\sum_{j=1}^m \nabla f_j(\widehat{\mathbf{x}}_j), \tag{38a}$$
$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha \cdot \mathbf{g}_k, \tag{38b}$$

where $\widehat{\mathbf{x}}_j$ is the most recent iterate where ∇f_j was evaluated and τ_k is chosen uniformly at random from the index set $\{1, \ldots, m\}$, at each iteration k. The gradient estimator $\mathbf{g}_k$ is implemented as follows. For an arbitrary $\mathbf{x}_0 \in \mathbb{R}^p$, compute $\nabla f_1(\mathbf{x}_0), \nabla f_2(\mathbf{x}_0), \ldots, \nabla f_m(\mathbf{x}_0)$ and store them in a table. At each $k \geq 0$, draw an index τ_k, compute $\mathbf{g}_k$ and replace the τ_k-th element of the table with $\nabla f_{\tau_k}(\mathbf{x}_k)$ while other entries in the table remain unchanged. It can be shown that the **SAGA** estimator is unbiased [44], i.e., $\mathbb{E}[\mathbf{g}_k] = \nabla F(\mathbf{x}_k)$, and has the property that

$$\mathbb{E}[\|\mathbf{g}_k - \nabla F(\mathbf{x}_k)\|_2^2] \to 0, \tag{39}$$

as $\mathbf{x}_k \to \mathbf{x}^*$. Under the assumption that each component function is smooth and strongly convex, i.e., $f_j \in \mathcal{S}_{\mu,L}, \forall j$, it can be shown that with $\alpha = \frac{1}{3\ell}$, we have [44],

$$\mathbb{E}\left[\|\mathbf{x}_k - \mathbf{x}^*\|_2^2\right] \leq C\left(1 - \min\left\{\tfrac{1}{4m}, \tfrac{\mu}{3\ell}\right\}\right)^k, \quad \forall k \geq 0, \tag{40}$$

for some constant $C > 0$. In other words, **SAGA** achieves ε-accuracy of the global minimum $\mathbf{x}^*$ with $\mathcal{O}\left(\max\{m, \kappa\}\ \ln\varepsilon^{-1}\right)$ component gradient evaluations, where recall that $\kappa = \frac{\ell}{\mu}$ is the condition number of the global cost F.

*Remark 15 (**SGD** vs **SAGA**):* We note that **SAGA**, particularly because of the variance reduction (40), achieves linear convergence that is independent of the variance. It is sometimes argued [11] that when low-precision solutions suffice, the overhead of **SAGA**, in terms of gradient storage and slow convergence in the beginning, may not be advantageous enough in comparison with the simplicity of **SGD** iterations that achieve a fairly decent solution very fast.

Remark 16 (Other variance-reduction methods): Other well-known and popular variance reduction techniques include **SAG** [42], **SAGA** [44], **SVRG** [43], **S2GD** [111], **SARAH** [46] and **SPIDER** [112]. These methods mostly differ in terms of the gradient estimator and lead to different computation and storage trade-offs. See [11] for detailed comparison.

A. Decentralized Gradient Tracking and Variance Reduction: GT-SAGA

Following the discussion so far, it is natural to extend decentralized stochastic methods with the help of both gradient tracking and variance reduction. Recall that the convergence in **DSGD**, **GT-DSGD**, and **SAB** is inexact and adding variance reduction to **GT-DSGD** or **SAB** potentially removes their steady-state error. The protocol at node i is summarized as:

(i) *Variance reduction*: Draw a local stochastic gradient and update the estimate $\mathbf{g}_k^i$ of the local full batch;
(ii) *Gradient tracking*: Fuse the local full batch estimates $\mathbf{g}_k^i$'s over nearby nodes to obtain $\mathbf{y}_{k+1}^i$;
(iii) *Local descent*: Update $\mathbf{x}_{k+1}^i$ with the new descent $\mathbf{y}_{k+1}^i$.

We can interpret the variance reduction step as intra-node fusion where each node estimates its own local full batch, while the gradient tracking step as the inter-node fusion where the nodes fuse (the estimate of) their local full gradients over

[6]The bounded variance condition is sometimes relaxed to (see e.g., [101]) $\mathbb{E}_{\xi_k^i}\left[\left\|\mathbf{g}_i(\mathbf{x}_k^i, \xi_k^i) - \nabla f_i(\mathbf{x}_k^i)\right\|_2^2 | \mathbf{x}_k^i\right] \leq M_{\mathbf{x}}\|\mathbf{x}_k^i\|^2 + \sigma^2$, for some $M_{\mathbf{x}} > 0$.

the network. In essence, the variability in the gradient selection process is removed by performing these two fusions and the overall descend is in the global direction, asymptotically.

Following **AB/Push-Pull**, the resulting algorithm over directed graphs, termed as **AB-SAGA**, is described as follows. Each node i starts with an arbitrary $\mathbf{x}_0^i \in \mathbb{R}^p$, maintains a local gradient table $\left[\nabla f_{i,1}(\mathbf{x}_0^i),\ \ldots\ ,\nabla f_{i,m_i}(\mathbf{x}_0^i)\right]$, and computes $\mathbf{y}_0^i = \nabla f_i(\mathbf{x}_0^i)$. Each node i then iteratively performs the following steps at each $k \geq 0$:

(i) Perform the $\mathbf{x}_k$-update:

$$\mathbf{x}_{k+1}^i = \sum_{r\in\mathcal{N}_i} a_{ir}\mathbf{x}_k^r - \alpha\mathbf{y}_k^i, \tag{41}$$

where $\underline{A} = \{a_{ir}\}$ is row stochastic;

(ii) Draw τ_k^i uniformly at random from $\{1,\ldots,m_i\}$;

(iii) Variance reduction:

$$\mathbf{g}_{k+1}^i = \nabla f_{i,\tau_k^i}\big(\mathbf{x}_{k+1}^i\big) - \nabla\widehat{f}_{i,\tau_k^i} + \frac{1}{m_i}\textstyle\sum_{j_i=1}^{m_i}\nabla\widehat{f}_{i,j_i}, \tag{42}$$

where $\widehat{f}_{i,\tau_k^i}$ is the τ_k^i-th element in the gradient table;

(iv) Gradient tracking:

$$\mathbf{y}_{k+1}^i = \sum_{r\in\mathcal{N}_i} b_{ir}\mathbf{d}_k^r + \mathbf{g}_{k+1}^i - \mathbf{g}_k^i, \tag{43}$$

where $\underline{B} = \{b_{ir}\}$ is column stochastic;

(v) Replace $\widehat{f}_{i,\tau_k^i}$ with $\nabla f_{i,\tau_k^i}(\mathbf{x}_{k+1}^i)$ in the gradient table.

The analysis of **AB-SAGA** described above remains an open problem for the general case of row stochastic and column stochastic weight matrices, $\underline{A} = \{a_{ir}\}$ and $\underline{B} = \{b_{ir}\}$.

A special case when both weights are doubly stochastic was introduced recently as **GT-SAGA** in [52], [113] that is applicable to undirected or to balanced directed graphs. Similar to the centralized **SAGA** [44], **GT-SAGA** converges linearly to $\mathbf{x}^*$ with a constant step-size. Formally, when each $f_{i,j_i} \in \mathcal{S}_{\mu,\ell}$, and the step-size is chosen such that $\alpha = \min\left\{\mathcal{O}\left(\frac{1}{\mu M}\right), \mathcal{O}\left(\frac{m}{M}\frac{(1-\lambda)^2}{\kappa L}\right)\right\}$, we have that $\forall k \geq 0$,

$$\frac{1}{n}\sum_{i=1}^{n}\mathbb{E}\left[\left\|\mathbf{x}_k^i - \mathbf{x}^*\right\|_2^2\right] \leq R\left(1 - \min\left\{\mathcal{O}\left(\tfrac{1}{M}\right), \mathcal{O}\left(\tfrac{m}{M}\tfrac{(1-\lambda)^2}{\kappa^2}\right)\right\}\right)^k, \tag{44}$$

where $m = \min_i\{m_i\}, M = \max_i\{m_i\}$, and $R > 0$ is some constant [52], [113]. In other words, **GT-SAGA** achieves ε-accuracy of $\mathbf{x}^*$ in

$$\mathcal{O}\left(\max\left\{M, \tfrac{M}{m}\tfrac{\kappa^2}{(1-\lambda)^2}\right\}\ln\varepsilon^{-1}\right)$$

parallel local component gradient computations.

*Remark 17 (**DSGD**, **GT-DSGD**, and **GT-SAGA**):* Recapping the performances of the decentralized stochastic first-order methods over undirected graphs, we note that **GT-SAGA** does not have a steady-state error due to variance-reduction [52], [113]. However, the gradients tables result in an increased storage cost $\mathcal{O}(pm_i)$ at each node i. Given a desired accuracy, applicable trade-offs between the convergence benefits of the **GT-SAGA** versus the simplicity of other methods can be formulated; see also [114] for a detailed review.

Remark 18 (Other decentralized VR methods): Decentralized VR methods that do not explicitly employ gradient tracking include **DSA** [47] that combines **EXTRA** [19] with **SAGA** [44], **diffusion-AVRG** [48] that combines exact diffusion [20] and **AVRG** [115], **DSBA** [49] that adds proximal mapping [116] to each iteration of **DSA**, and **ADFS** [51] that applies an accelerated randomized proximal coordinate gradient method [117] to the dual formulation of Problem P0. We note that when $M \approx m$ is very large, **GT-SAGA** improves upon the convergence rate of these methods in terms of the joint dependence on κ and $M \approx m$, with the exception of **DSBA** and **ADFS**. Both **DSBA** and **ADFS** achieve better iteration complexity, however, at the expense of computing the proximal mapping of a component function at each iteration. Although the computation of this proximal mapping is efficient for certain function classes, it can be very expensive for general functions. VR methods that incorporate gradient tracking include **GT-SVRG** [52], [118] and **GT-SARAH** [119] where the former has certain advantages over **GT-SAGA** when it comes to the storage cost while **GT-SARAH** considers nonconvex problems.

*Remark 19 (Linear speedup of **GT-SAGA**):* We note that **GT-SAGA** has a low per-iteration computation cost and achieves a linear convergence to $\mathbf{x}^*$. In particular, it reaches ε-accuracy of $\mathbf{x}^*$ in $\mathcal{O}\left(\max\left\{M, \frac{M}{m}\frac{\kappa^2}{(1-\lambda)^2}\right\}\ln\varepsilon^{-1}\right)$ parallel local component gradient computations. This exact linear convergence makes it particularly favorable in comparison with **DSGD** and **GT-DSGD** when high-precision solutions are desired. Interestingly, when each node has a large data set such that $M \approx m > \frac{\kappa^2}{1-\lambda^2}$, the complexity of **GT-SAGA** becomes $\mathcal{O}(M\ \ln\varepsilon^{-1})$, independent of the network, and is thus n times faster than that of centralized **SAGA** where it is $\mathcal{O}(nM\ \ln\varepsilon^{-1})$ [44]. Clearly, in this "big-data" regime, **GT-SAGA** acts effectively as a means for parallel computation and achieves linear speed-up compared with centralized variance-reduced methods. We also numerically show the linear speedups of decentralized algorithms in Section VIII.

VIII. Numerical Experiments

In this section, we compare the performance of the decentralized first-order methods discussed in this article with the help of linear and logistic regression problems.

1) Decentralized state estimation in sensor networks

Consider a network of sensors deployed to estimate a state vector $\mathbf{x} \in \mathbb{R}^p$, e.g., the location of a target. Each sensor i obtains its local measurements $\mathbf{y}_i \in \mathbb{R}_i^p$ as $\mathbf{y}_i = H_i\mathbf{x} + \mathbf{n_i}$, where $H_i \in \mathbb{R}^{p_i\times p}$ is the local sensing matrix and $\mathbf{n}_i \in \mathbb{R}^p$ is the random noise. The nodes cooperate to solve the following decentralized optimization problem: $\min_{\mathbf{x}\in\mathbb{R}^p}\sum_{i=1}^n\|\mathbf{y}_i - H_i\mathbf{x}\|^2$. We randomly generate an undirected graph of 100 nodes using the nearest neighbor rule, i.e., two nodes are connected if they lie within a communication radius. The directed geometric graph is obtained by making half of the links in the undirected geometric graph to be one-directional. Two sample graphs are shown in Fig. 3. We set the dimension of the state $\mathbf{x} \in \mathbb{R}^p$ to be $p = 100$ and generate the sensing matrices and the state vector from Gaussian distribution with zero-mean and a standard deviation of 10. The maximum rank of the sensing matrices does not exceed 20, i.e., no node is able to recover the state on its own.

In Fig. 5, we compare **DGD**, **GT-DGD**, and **AB/Push-Pull** over an undirected graph. Note that doubly stochastic weights are used for **DGD** and **GT-DGD** while row and column stochastic weights are used in **AB/Push-Pull**. The same constant step-size $\alpha = 10^{-5}$ is used for all of the algorithms. As discussed before, **DGD** with a constant step-size has inexact convergence while with a decaying step-size has exact but sub-linear convergence. Both **GT-DGD** and **AB/Push-Pull** converge linearly to the exact solution due to gradient tracking. In Fig. 6, we compare Gradient-Push (**GP**), **ADDOPT/Push-DIGing** (with only column stochastic weights), **FROST** (with only row stochastic weights) and **AB/Push-Pull** over a directed graph. **GP** being the extension of **DGD** with push-sum consensus has inexact convergence with a constant step-size while all of the other algorithms are based on gradient tracking and converge linearly.

2) *Decentralized training over multiple machines*

We next consider decentralized logistic regression over the directed exponential graphs [39], shown in Fig. 8, where the goal is to classify hand-written digits 3 and 8 from the MNIST dataset [120]. The class of directed exponential graphs is weight-balanced (therefore admits doubly stochastic weights), sparsely-connected (low-communication per node), and possesses a strong algebraic connectivity. The nodes cooperatively solve the following smooth and strongly convex problem: $\min_{\mathbf{w},b} F(\mathbf{w}, b) = \frac{1}{nm}\sum_{i=1}^{n}\sum_{j_i=1}^{m} f_{i,j_i}(\mathbf{w}, b)$, with

$$f_{i,j_i}(\mathbf{w}, b) = \ln\left[1 + \exp\left\{-(\mathbf{w}^\top \mathbf{x}_{j_i} + b)y_{j_i}\right\}\right] + \frac{\lambda}{2}\|\mathbf{w}\|_2^2,$$

where $\mathbf{x}_{j_i} \in \mathbb{R}^{784}$ is the feature vector and y_{j_i} is the corresponding binary label.

In our setup, a total number of 11968 training samples are evenly distributed among all nodes. Each training sample is normalized to a unit vector. We set the regularization parameter $\lambda = \frac{1}{nm}$ [42]. The global minimum is found by centralized Nesterov gradient descent. We plot the average residual $\frac{1}{n}\sum_{i=1}^{n}\left(F(x_k^i) - F^*\right)$ across all nodes for comparison. The hyper-parameters for all algorithms are tuned manually for best performances.

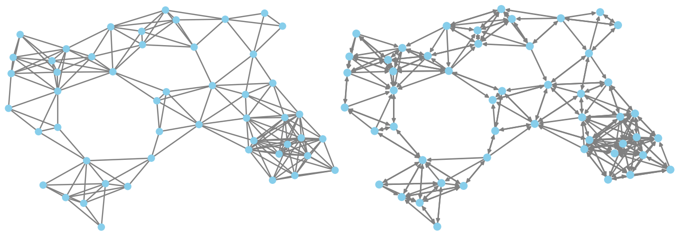


Fig. 3. An undirected geometric graph (left) and a directed geometric graph (right) generated by nearest neighbor rule.

We first show the performance over the directed exponential graph with $n = 8$ nodes (thus $m = 1496$). Fig. 6 compares **DGD**, **GT-DGD**, **GT-DSGD**, **GT-DSGD**, and **GT-SAGA**, all with constant step-sizes, where one unit on the horizontal axis (or epoch) represents $m = 1496$ parallel component gradient evaluations, i.e., one effective pass of the local data. It can be observed that in the first few epochs, stochastic gradient methods, **DSGD** and **GT-DSGD**, make very fast progress and significantly outperform their deterministic counterparts, **DGD** and **GT-DGD**, and therefore are particularly favorable for problems where low-precision solutions suffice. Over time, however, stochastic methods drastically slow down and full gradient methods that make steady progress across iterations typically achieve high accuracy faster. Clearly, **GT-SAGA** that uses both gradient tracking and variance-reduction exhibits consistent fast linear convergence to the global minimum and outperforms all other methods.

Finally, we study the speedup achieved by the decentralized stochastic methods over their centralized counterparts in Fig. 7, where we compare **DSGD** and **GT-DSGD** with centralized **SGD** and **GT-SAGA** with centralized **SAGA**. Each iteration of centralized **SGD** and **SAGA** evaluates one component gradient while each iteration of **DSGD**, **GT-DSGD** and **GT-SAGA** evaluates n component gradients in parallel, i.e., one at each node. The speedup can be described as the the ratio of the number of iterations taken by the centralized method over the corresponding decentralized method to achieve the same accuracy (10^{-3} for comparisons with **SGD** and 10^{-15} for comparison with **SAGA**). We conduct the experiment over the directed exponential graphs with $4, 8, 6$, and 32 nodes shown in Fig. 8. It can be observed that a linear speedup is achieved for all three decentralized approaches. In other words, the decentralized methods are approximately n times faster than their centralized counterparts and thus are particularly favorable when data can be processed in parallel.

IX. Conclusions

In this article, we study decentralized first-order methods with full and stochastic gradients over undirected and directed graphs. We show that most of the existing work on decentralized methods based on gradient tracking can be cast in a unifying framework provided by the **AB/Push-Pull** algorithm. On the stochastic front, we discuss how gradient tracking is unable to achieve exact linear convergence, which can be recovered with the addition of a recently introduced variance-reduction technique. We provide numerical results to illustrate the convergence behavior and performance of the corresponding methods.

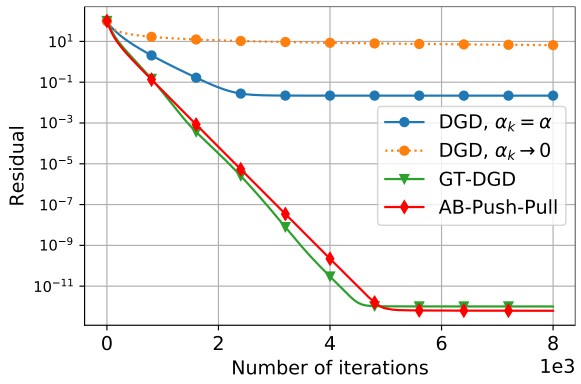


Fig. 4. Decentralized state estimation over undirected geometric graph.

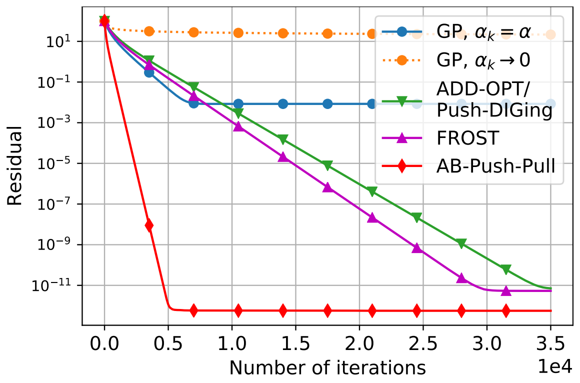


Fig. 5. Decentralized state estimation over directed geometric graph.

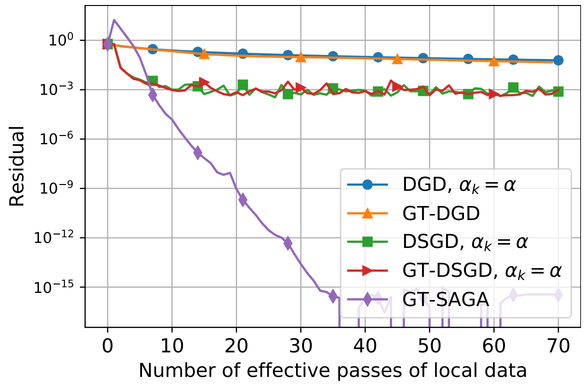


Fig. 6. Decentralized logistic regression over the directed exponential graph with $n = 8$ nodes.

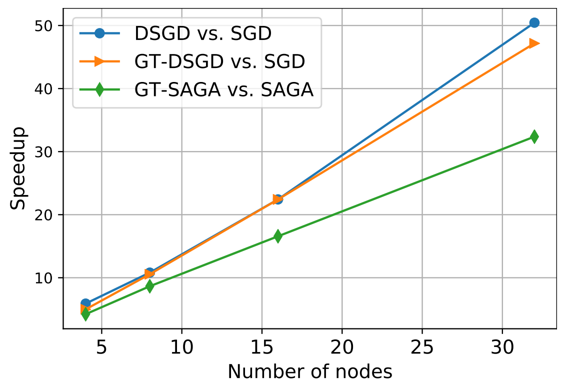


Fig. 7. Linear speedup: **DSGD** and **GT-DSGD** vs. **SGD** to achieve an accuracy of 10^{-3}; **GT-SAGA** vs. **SAGA** to achieve an accuracy of 10^{-15}.

References

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

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

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

[4] S. Pu, W. Shi, J. Xu, and A. Nedić, "A push-pull gradient method for distributed optimization in networks," in *58th IEEE Conference on Decision and Control*, 2018, pp. 3385–3390.

[5] J. Tsitsiklis, D. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE Trans. on Autom. Control*, vol. 31, no. 9, pp. 803–812, 1986.

[6] P. A. Forero, A. Cano, and G. B. Giannakis, "Consensus-based distributed support vector machines," *Journal of Machine Learning Research*, vol. 11, no. May, pp. 1663–1707, 2010.

[7] S. Kar, J. M. F. Moura, and K. Ramanan, "Distributed parameter estimation in sensor networks: Nonlinear observation models and imperfect communication," *IEEE Trans. on Information Theory*, vol. 58, no. 6, pp. 3575–3605, 2012.

[8] J. F. C. Mota, J. M. F. Xavier, P. M. Q. Aguiar, and M. Püschel, "Distributed basis pursuit," *IEEE Trans. on Sig. Process.*, vol. 60, no. 4, pp. 1942–1956, Apr. 2012.

[9] H.-T. Wai, Z. Yang, Z. Wang, and M. Hong, "Multi-agent reinforcement learning via double averaging primal-dual optimization," *arXiv:1806.00877*, 2018.

[10] V. Cevher, S. Becker, and M. Schmidt, "Convex optimization for big data: Scalable, randomized, and parallel algorithms for big data analytics," *IEEE Sig. Process. Mag.*, vol. 31, no. 5, pp. 32–43, 2014.

[11] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *SIAM Review*, vol. 60, no. 2, pp. 223–311, 2018.

[12] T. Yang, X. Yi, J. Wu, Y. Yuan, D. Wu, Z. Meng, Y. Hong, H. Wang, Z. Lin, and K. H. Johansson, "A survey of distributed optimization," *Annual Reviews in Control*, 2019.

[13] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Trans. on Autom. Control*, vol. 49, no. 9, pp. 1520–1533, Sep. 2004.

[14] A. Olshevsky and J. N. Tsitsiklis, "Convergence speed in distributed consensus and averaging," *SIAM Journal on Control and Optimization*, vol. 48, no. 1, pp. 33–55, 2009.

[15] S. Safavi and U. A. Khan, "Revisiting finite-time distributed algorithms via successive nulling of eigenvalues," *IEEE Signal Processing Letters*, vol. 22, no. 1, pp. 54–57, Jan. 2015.

[16] K. Yuan, Q. Ling, and W. Yin, "On the convergence of decentralized gradient descent," *SIAM J. on Optim.*, vol. 26, no. 3, pp. 1835–1854, Sep. 2016.

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

[18] G. Qu and N. Li, "Harnessing smoothness to accelerate distributed optimization," *IEEE Trans. on Control of Network Systems*, vol. 5, no. 3, pp. 1245–1260, 2017.

[19] 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.

[20] K. Yuan, B. Ying, X. Zhao, and A. H. Sayed, "Exact diffusion for distributed optimization and learning Part I: Algorithm development," *IEEE Trans. on Sig. Process.*, vol. 67, no. 3, pp. 708–723, 2018.

[21] K. Yuan, B. Ying, X. Zhao, and A. H. Sayed, "Exact diffusion for distributed optimization and learning Part II: Convergence analysis," *IEEE Trans. on Sig. Process.*, vol. 67, no. 3, pp. 724–739, 2018.

[22] K. I. Tsianos, *The role of the Network in Distributed Optimization Algorithms: Convergence Rates, Scalability, Communication/Computation Tradeoffs and Communication Delays*, Ph.D. thesis, Dept. Elect. Comp. Eng. McGill University, 2013.

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

[24] C. Xi, Q. Wu, and U. A. Khan, "On the distributed optimization over directed networks," *Neurocomputing*, vol. 267, pp. 508–515, Dec. 2017.

[25] C. Xi and U. A. Khan, "Distributed subgradient projection algorithm over directed graphs," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3986–3992, Oct. 2016.

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

[27] A. Nedić, 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.

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

[29] C. Xi, V. S. Mai, R. Xin, E. Abed, and U. A. Khan, "Linear convergence in optimization over directed graphs with row-stochastic matrices," *IEEE Transactions on Automatic Control*, vol. 63, no. 10, pp. 3558–3565, Oct. 2018.

[30] R. Xin, C. Xi, and U. A. Khan, "FROST – Fast row-stochastic optimization with uncoordinated step-sizes," *EURASIP Journal on Advances in Signal Processing–Special Issue on Optimization, Learning, and Adaptation over Networks*, Jan. 2019.

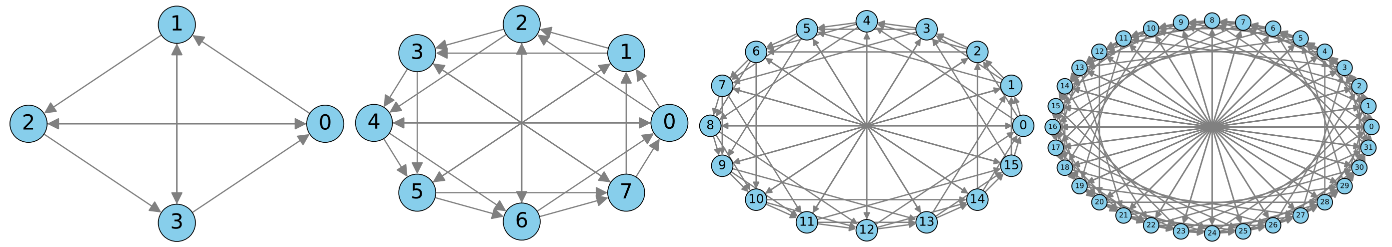


Fig. 8. Directed exponential graphs with $4, 8, 16$, and 32 nodes. These graphs are balanced and thus admit doubly stochastic weights.

[31] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information," in *44th Annual IEEE Symposium on Foundations of Computer Science*, Oct. 2003, pp. 482–491.

[32] F. Benezit, V. Blondel, P. Thiran, J. Tsitsiklis, and M. Vetterli, "Weighted gossip: Distributed averaging using non-doubly stochastic matrices," in *IEEE International Symposium on Information Theory*, Jun. 2010, pp. 1753–1757.

[33] A. Priolo, A. Gasparri, E. Montijano, and C. Sagues, "A distributed algorithm for average consensus on strongly connected weighted digraphs," *Automatica*, vol. 50, no. 3, pp. 946–951, 2014.

[34] K. Cai and H. Ishii, "Average consensus on general strongly connected digraphs," *Automatica*, vol. 48, no. 11, pp. 2750–2761, 2012.

[35] S. S. Ram, A. Nedić, and V. V. Veeravalli, "Distributed stochastic subgradient projection algorithms for convex optimization," *Journal of Optim. Theory and Appl.*, vol. 147, no. 3, pp. 516–545, 2010.

[36] G. Morral, P. Bianchi, and G. Fort, "Success and failure of adaptation-diffusion algorithms for consensus in multi-agent networks," in *53rd IEEE Conference on Decis. and Contr.* IEEE, 2014, pp. 1476–1481.

[37] A. Nedić and A. Olshevsky, "Stochastic gradient-push for strongly convex functions on time-varying directed graphs," *IEEE Trans. on Autom. Control*, vol. 61, no. 12, pp. 3936–3947, 2016.

[38] X. Lian, C. Zhang, H. Zhang, C. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent," in *NeurIPS*, 2017, pp. 5330–5340.

[39] M. Assran, N. Loizou, N. Ballas, and M. Rabbat, "Stochastic gradient push for distributed deep learning," *arXiv:1811.10792*, 2018.

[40] J. Chen and A. H Sayed, "On the learning behavior of adaptive networks—Part I: transient analysis," *IEEE Transactions on Information Theory*, vol. 61, no. 6, pp. 3487–3517, 2015.

[41] S. Pu and A. Garcia, "A flocking-based approach for distributed stochastic optimization," *Operations Research*, vol. 1, pp. 267–281, 2018.

[42] M. Schmidt, N. Le Roux, and F. Bach, "Minimizing finite sums with the stochastic average gradient," *Mathematical Programming*, vol. 162, no. 1-2, pp. 83–112, 2017.

[43] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," in *Advances in neural information processing systems*, 2013, pp. 315–323.

[44] A. Defazio, F. Bach, and S. Lacoste-Julien, "SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives," in *NeurIPS*, 2014, pp. 1646–1654.

[45] Z. Allen-Zhu, "Katyusha: The first direct acceleration of stochastic gradient methods," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 8194–8244, 2017.

[46] L. M. Nguyen, J. Liu, K. Scheinberg, and M. Takáč, "SARAH: a novel method for machine learning problems using stochastic recursive gradient," in *ICML*. JMLR. org, 2017, pp. 2613–2621.

[47] A. Mokhtari and A. Ribeiro, "DSA: decentralized double stochastic averaging gradient algorithm," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2165–2199, 2016.

[48] K. Yuan, B. Ying, J. Liu, and A. H. Sayed, "Variance-reduced stochastic learning by networked agents under random reshuffling," *IEEE Trans. on Sig. Process.*, vol. 67, no. 2, pp. 351–366, 2018.

[49] Z. Shen, A. Mokhtari, T. Zhou, P. Zhao, and H. Qian, "Towards more efficient stochastic decentralized learning: Faster convergence and sparse communication," *arXiv:1805.09969*, 2018.

[50] Z. Wang and H. Li, "Edge-based stochastic gradient algorithm for distributed optimization," *IEEE Trans. Knowl. Data Eng.*, 2019.

[51] H. Hendrikx, F. Bach, and L. Massoulié, "Asynchronous accelerated proximal stochastic gradient for strongly convex distributed finite sums," *arXiv:1901.09865*, 2019.

[52] R. Xin, U. A. Khan, and S. Kar, "Variance-reduced decentralized stochastic optimization with accelerated convergence," *arXiv:1912.04230*, Dec. 2019.

[53] A. Mokhtari, W. Shi, Q. Ling, and A. Ribeiro, "A decentralized second-order method with exact linear convergence rate for consensus optimization," *IEEE Trans. on Sig. and Information Processing over Networks*, vol. 2, no. 4, pp. 507–522, 2016.

[54] M. Eisen, A. Mokhtari, and A. Ribeiro, "Decentralized quasi-newton methods," *IEEE Trans. on Sig. Process.*, vol. 65, no. 10, pp. 2613–2628, May 2017.

[55] A. Mokhtari, Q. Ling, and A. Ribeiro, "Network newton distributed optimization methods," *IEEE Trans. on Sig. Process.*, vol. 65, no. 1, pp. 146–161, Jan 2017.

[56] H.-T. Wai, N. M. Freris, A. Nedić, and A. Scaglione, "SUCAG: Stochastic unbiased curvature-aided gradient method for distributed optimization," *arXiv:1803.08198*, 2018.

[57] F. Mansoori and E. Wei, "Superlinearly convergent asynchronous distributed network newton method," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE, 2017, pp. 2874–2879.

[58] J. F. C. Mota, J. M. F. Xavier, P. M. Q. Aguiar, and M. Puschel, "D-ADMM: A communication-efficient distributed algorithm for separable optimization," *IEEE Trans. on Sig. Process.*, vol. 61, no. 10, pp. 2718–2723, May 2013.

[59] W. Shi, Q. Ling, K Yuan, G Wu, and W Yin, "On the linear convergence of the admm in decentralized consensus optimization," *IEEE Trans. on Sig. Process.*, vol. 62, no. 7, pp. 1750–1761, April 2014.

[60] E. Wei and A. Ozdaglar, "On the o (1/k) convergence of asynchronous distributed alternating direction method of multipliers," in *2013 IEEE Global Conference on Sig. and Information Processing*. IEEE, 2013, pp. 551–554.

[61] M. Maros and J. Jaldén, "On the Q-linear convergence of distributed generalized ADMM under non-strongly convex function components," *IEEE Trans. Signal Inf. Process. Netw.*, 2019.

[62] H. Sun and M. Hong, "Distributed non-convex first-order optimization and information processing: Lower complexity bounds and rate optimal algorithms," *IEEE Trans.s on Signal Process.*, vol. 67, no. 22, pp. 5912–5928, 2019.

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

[64] D. Jakovetić, "A unification and generalization of exact distributed first-order methods," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 5, no. 1, pp. 31–46, 2018.

[65] M. Hong, D. Hajinezhad, and M. Zhao, "Prox-pda: The proximal primal-dual algorithm for fast distributed nonconvex optimization and learning over networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 1529–1538.

[66] M. Hong, Z. Luo, and M. Razaviyayn, "Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems," *SIAM Journal on Optimization*, vol. 26, no. 1, pp. 337–364, 2016.

[67] C. A. Uribe, S. Lee, A. Gasnikov, and A. Nedić, "A dual approach for optimal algorithms in distributed optimization over networks," *arXiv:1809.00710*, 2018.

[68] M. Maros and J. Jaldén, "A geometrically converging dual method for distributed optimization over time-varying graphs," *arXiv:1810.05760*, 2018.

[69] K. Scaman, F. Bach, S. Bubeck, Y. T. Lee, and L. Massoulié, "Optimal algorithms for smooth and strongly convex distributed optimization in networks," in *ICML*. JMLR. org, 2017, pp. 3027–3036.

[70] A. Nedić, A. Olshevsky, and M. G. Rabbat, "Network topology and communication-computation tradeoffs in decentralized optimization," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 953–976, 2018.

[71] B. Li, S. Cen, Y. Chen, and Y. Chi, "Communication-efficient distributed optimization in networks with gradient tracking," *arXiv:1909.05844*, 2019.

[72] D. Jakovetic, J. M. F. Xavier, and José M. F. Moura, "Convergence rates of distributed nesterov-like gradient methods on random networks.," *IEEE Trans. Sig. Process.*, vol. 62, no. 4, pp. 868–882, 2014.

[73] F. Saadatniaki, R. Xin, and U. A. Khan, "Optimization over time-varying directed graphs with row and column-stochastic matrices," *IEEE Transactions on Automatic Control*, Oct. 2018, Arxiv: https://arxiv.org/abs/1810.07393, *under review*.

[74] J. Xu, S. Zhu, Y. C. Soh, and L. Xie, "Convergence of asynchronous distributed gradient methods over stochastic networks," *IEEE Trans. on Autom. Control*, vol. 63, no. 2, pp. 434–448, 2018.

[75] A. Olshevsky, I. Paschalidis, and A. Spiridonoff, "Robust asynchronous stochastic gradient-push: asymptotically optimal and network-independent performance for strongly convex functions," *arXiv:1811.03982*, 2018.

[76] A. Reisizadeh, A. Mokhtari, H. Hassani, and R. Pedarsani, "An exact quantized decentralized gradient descent algorithm," *IEEE Trans. on Sig. Process.*, vol. 67, no. 19, pp. 4934–4947, Oct 2019.

[77] A. Berahas, R. Bollapragada, N. S. Keskar, and E. Wei, "Balancing communication and computation in distributed optimization," *IEEE Transactions on Automatic Control*, 2018.

[78] J. Zhang and K. You, "Asyspa: An exact asynchronous algorithm for convex optimization over digraphs," *IEEE Transactions on Automatic Control*, 2019.

[79] L. Bottou, F. E Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *SIAM Review*, vol. 60, no. 2, pp. 223–311, 2018.

[80] Y. Nesterov, *Lectures on convex optimization*, vol. 137, Springer, 2018.

[81] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, Cambridge, 1985.

[82] B. Polyak, *Introduction to optimization*, Optimization Software, 1987.

[83] J. Chen and A. H. Sayed, "On the learning behavior of adaptive networks—part i: Transient analysis," *IEEE Transactions on Information Theory*, vol. 61, no. 6, pp. 3487–3517, 2015.

[84] J. Chen and A. H. Sayed, "On the learning behavior of adaptive networks—part ii: Performance analysis," *IEEE Transactions on Information Theory*, vol. 61, no. 6, pp. 3518–3548, 2015.

[85] U. A. Khan and J. M. F. Moura, "Distributing the Kalman filter for large-scale systems," *IEEE Trans. on Sig. Process.*, vol. 56, no. 10, pp. 4919–4935, Oct. 2008.

[86] U. A. Khan, S. Kar, A. Jadbabaie, and J. M.F. Moura, "On connectivity, observability, and stability in distributed estimation," in *49th IEEE Conf. of Decis. and Contr.*, Atlanta, GA, Dec. 2010, pp. 6639–6644.

[87] S. Kar and J. M. F. Moura, "Consensus+ innovations distributed inference over networks: cooperation and sensing in networked systems," *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 99–109, 2013.

[88] K. I. Tsianos, S. Lawlor, and M. G. Rabbat, "Push-sum distributed dual averaging for convex optimization," in *51st IEEE Annual Conference on Decision and Control*, Maui, Hawaii, Dec. 2012, pp. 5453–5458.

[89] A. Nedić and A. Olshevsky, "Distributed optimization over time-varying directed graphs," in *52nd IEEE Annual Conference on Decision and Control*, Dec. 2013, pp. 6855–6860.

[90] V. S. Mai and E. H. Abed, "Distributed optimization over weighted directed graphs using row stochastic matrix," in *IEEE American Control Conference*, July 2016, pp. 7165–7170.

[91] P. Di Lorenzo and G. Scutari, "Next: In-network nonconvex optimization," *IEEE Trans. on Sig. and Information Processing over Networks*, vol. 2, no. 2, pp. 120–136, 2016.

[92] J. Xu, S. Zhu, Y. C. Soh, and L. Xie, "Convergence of asynchronous distributed gradient methods over stochastic networks," *IEEE Transactions on Automatic Control*, vol. 63, no. 2, pp. 434–448, 2017.

[93] G. Scutari and Y. Sun, "Distributed nonconvex constrained optimization over time-varying digraphs," *Mathematical Programming*, vol. 176, no. 1-2, pp. 497–544, 2019.

[94] M. Zhu and S. Martínez, "Discrete-time dynamic average consensus," *Automatica*, vol. 46, no. 2, pp. 322–329, 2010.

[95] S. Pu, W. Shi, J. Xu, and A. Nedić, "Push-pull gradient methods for distributed optimization in networks," *IEEE Transactions on Automatic Control*, 2020.

[96] R. Xin and U. A. Khan, "Distributed heavy-ball: A generalization and acceleration of first-order methods with gradient tracking," *IEEE Transactions on Automatic Control*, Sep. 2019, to appear.

[97] S. Safavi and U. A. Khan, "Leader-follower consensus in mobile sensor networks," *IEEE Signal Processing Letters*, vol. 22, no. 12, pp. 2249–2253, Dec. 2015.

[98] U. A. Khan R. Xin, A. K. Sahu and S. Kar, "Distributed stochastic optimization with gradient tracking over strongly-connected networks," in *58th IEEE Conference of Decision and Control*, Nice, France, 2019.

[99] R. Xin, D. Jakovetić, and U. A. Khan, "Distributed Nesterov gradient methods over arbitrary graphs," *IEEE Signal Processing Letters*, vol. 26, no. 18, pp. 1247–1251, Jun. 2019.

[100] G. Qu and N. Li, "Accelerated distributed nesterov gradient descent," *IEEE Trans. on Autom. Control*, 2019.

[101] K. Yuan, S. A. Alghunaim, B. Ying, and A. H. Sayed, "On the performance of exact diffusion over adaptive networks," *arXiv:1903.10956*, 2019.

[102] D. Jakovetic, D. Bajovic, A. K. Sahu, and S. Kar, "Convergence rates for distributed stochastic optimization over random networks," in *IEEE Conference on Decision and Control*, 2018, pp. 4238–4245.

[103] A. Koloskova, S. U Stich, and M. Jaggi, "Decentralized stochastic optimization and gossip algorithms with compressed communication," *arXiv:1902.00340*, 2019.

[104] S. Pu, A. Olshevsky, and I. C. Paschalidis, "A sharp estimate on the transient time of distributed stochastic gradient descent," *arXiv:1906.02702*, 2019.

[105] S. Pu, A. Olshevsky, and I. C. Paschalidis, "Asymptotic network independence in distributed stochastic optimization for machine learning," *IEEE Signal Processing Magazine*, 2020, to appear.

[106] S. Pu and A. Nedić, "A distributed stochastic gradient tracking method," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 963–968.

[107] S. Pu and A. Nedić, "Distributed stochastic gradient tracking methods," *Mathematical Programming*, 2020.

[108] J. Zhang and K. You, "Decentralized stochastic gradient tracking for empirical risk minimization," *arXiv:1909.02712*, 2019.

[109] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro, "Robust stochastic approximation approach to stochastic programming," *SIAM Journal on optimization*, vol. 19, no. 4, pp. 1574–1609, 2009.

[110] S. Ghadimi and G. Lan, "Stochastic first-and zeroth-order methods for nonconvex stochastic programming," *SIAM Journal on Optimization*, vol. 23, no. 4, pp. 2341–2368, 2013.

[111] J. Konečnỳ, J. Liu, P. Richtárik, and M. Takáč, "Mini-batch semi-stochastic gradient descent in the proximal setting," *IEEE J. Sel. Topics Signal Process.*, vol. 10, no. 2, pp. 242–255, 2015.

[112] C. Fang, C. J. Li, Z. Lin, and T. Zhang, "Spider: Near-optimal non-convex optimization via stochastic path-integrated differential estimator," in *NeurIPS*, 2018, pp. 689–699.

[113] R. Xin, U. A. Khan, and S. Kar, "Variance-reduced decentralized stochastic optimization with gradient tracking–Part I: GT-SAGA," *Arxiv: 1909.11774*, Sep. 2019.

[114] R. Xin, S. Kar, and U. A. Khan, "An introduction to decentralized stochastic optimization with gradient tracking," *IEEE Sig. Process. Magazine, arXiv:1907.09648*, 2019, under review.

[115] B. Ying, K. Yuan, and A. H. Sayed, "Variance-reduced stochastic learning under random reshuffling," *arXiv:1708.01383*, 2017.

[116] A. Defazio, "A simple practical accelerated method for finite sums," in *Advances in neural information processing systems*, 2016, pp. 676–684.

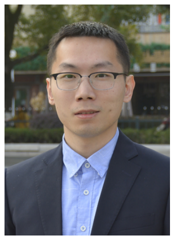
[117] Q. Lin, Z. Lu, and L. Xiao, "An accelerated randomized proximal coordinate gradient method and its application to regularized empirical risk minimization," *SIAM J. Optim.*, vol. 25, no. 4, pp. 2244–2273, 2015.

[118] R. Xin, U. A. Khan, and S. Kar, "Variance-reduced decentralized stochastic optimization with gradient tracking–Part II: GT-SVRG," *arXiv:1910.04057*, Oct. 2019.

[119] H. Sun, S. Lu, and M. Hong, "Improving the sample and communication complexity for decentralized non-convex optimization: A joint gradient estimation and tracking approach," *arXiv:1910.05857*, 2019.

[120] Y. LeCun, "The MNIST database of handwritten digits," *http://yann. lecun. com/exdb/mnist/*, 1998.

Ran Xin received his B.S. degree in Mathematics and Applied Mathematics from Xiamen University, China, in 2016, and M.S. degree in Electrical and Computer Engineering from Tufts University in 2018. Currently, he is a Ph.D. candidate in the Electrical and Computer Engineering Department at Carnegie Mellon University. His research interests include first-order methods for convex and nonconvex optimization, particularly in networks.



Shi Pu is currently an assistant professor in the Institute for Data and Decision Analytics, The Chinese University of Hong Kong, Shenzhen, China. He received a B.S. Degree in Engineering Mechanics from Peking University, in 2012, and a Ph.D. Degree in Systems Engineering from the University of Virginia, in 2016. He was a postdoctoral associate at the University of Florida, from 2016 to 2017, a postdoctoral scholar at Arizona State University, from 2017 to 2018, and a postdoctoral associate at Boston University, from 2018 to 2019. His research interests include distributed optimization, network science, machine learning, and game theory.

Angelia Nedić has a Ph.D. from Moscow State University, Moscow, Russia, in Computational Mathematics and Mathematical Physics (1994), and a Ph.D. from Massachusetts Institute of Technology, Cambridge, USA in Electrical and Computer Science Engineering (2002). She has worked as a senior engineer in BAE Systems North America, Advanced Information Technology Division at Burlington, MA. Currently, she is a faculty member of the school of Electrical, Computer and Energy Engineering at Arizona State University at Tempe. Prior to joining Arizona State University, she has been a Willard Scholar faculty member at the University of Illinois at Urbana-Champaign. She is a recipient (jointly with her co-authors) of the Best Paper Awards at the Winter Simulation Conference 2013 and at the International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt) 2015. Her general research interest is in optimization, large scale complex systems dynamics, variational inequalities and games.

Usman A. Khan is an Associate Professor of Electrical and Computer Engineering (ECE) at Tufts University, Medford, MA, USA. His research interests include statistical signal processing, network science, and decentralized optimization over autonomous multiagent systems. Recognition of his work includes the prestigious National Science Foundation (NSF) Career award, an IEEE journal cover, three best student paper awards in IEEE conferences, and several news articles including two in *IEEE Spectrum*. He received his B.S. degree in 2002 from University of Engineering and Technology, Pakistan, M.S. degree in 2004 from University of Wisconsin-Madison, USA, and Ph.D. degree in 2009 from Carnegie Mellon University, USA, all in ECE. Dr. Khan is an *IEEE Senior Member* and has been an elected full member of the *Sensor Array and Multichannel Technical Committee* with the *IEEE Signal Processing Society* since 2019 where he was an Associate member from 2010 to 2019. He was an elected full member of the *IEEE Big Data special interest group* from 2017 to 2019. He was an Editor of the *IEEE Transactions on Smart Grid* from 2014 to 2017 and is currently an Associate Editor of the *IEEE Control System Letters*, *IEEE Transactions Signal and Information Processing over Networks*, and *IEEE Open Journal of Signal Processing*. He is the Lead Guest Editor for the *Proceedings of the IEEE Special Issue on Optimization for Data-driven Learning and Control* and a Guest Associate Editor for *IEEE Control System Letters Special Issue on Learning and Control* both to appear in 2020. He is the Technical Area Chair for the Networks track in *IEEE 2020 Asilomar Conference on Signals Systems and Computers*.

Authors Response: 0007-SIP-2020-PIEEE
A general framework for decentralized optimization

Ran Xin, Shi Pu, Angelia Nedić, and Usman A. Khan

I. General comments

We thank the editors and reviewers for the time and effort invested in providing detailed and constructive suggestions. Please see our point-by-point reply in the next sections.

II. Reply to Reviewer 1's comments

This is a nice review article on decentralized first-order optimization methods, in both deterministic and stochastic settings. The Authors have done a very good job in presenting and explaining the state of the art in this area, and this is important because decentralized (stochastic) gradient methods have a relatively long history, and the literature in the area is very extensive. This can be also seen by the relatively long list of references in the paper. In addition to the above, I find that the "general framework" that the authors propose, which is mainly concentrated around the AB/Push-Pull scheme is interesting, and can assist a newcomer in the area understand the different approaches treated from an elevated point of view. From this perspective, I think that this work has educational merit, as well. One particular element of the paper that I find helpful is that the authors very openly describe current open problems, which of course can be the basis for new results in the area.

Reply: We thank the reviewer for these constructive comments and all the valuable suggestions.

One suggestion that I have is that the authors talk more, at least in the introduction, about decentralized consensus-based first-order methods which are based on the Alternating Direction Method of Multipliers (ADMM), and provide a good selection of related references. There has been a lot of work on this topical area and with great results, including very nice applications, such as in estimation problems, matrix completion and nonlinear filtering. Please add a related discussion at least in the Introduction, including those references. Also, it might be worth referring to diffusion-based algorithms and related applications, again in the Introduction, as well. These would provide a very clear overview of the area to the reader (together with the related references).

Reply: We have added relevant references and discussion on ADMM and diffusion-based methods, please see Section I and Remark 2.

But in general, this work, although (to be clear) does *not* present new results in the area of first-order decentralized optimization, but rather unifies many of the existing results in the area, is indeed solid, and it was a pleasure to read. I think this can be a nice reference paper in the area. I would therefore recommend this paper for acceptance subject to minor revisions, so that my suggestions above is successfully addressed.

Reply: The reviewer is correct that the paper does not have any new results. This is in spirit of the Proceedings of the IEEE and our goal is to write a reader-friendly overview paper that provides deep insights into the fundamental design and analysis of decentralized optimization algorithms.

III. Reply to Reviewer 2's comments

The paper "A general framework for decentralized optimization with first-order methods" provides a deep comparison of most known decentralized first-order methods developed for solving optimal control and estimation problems. Decentralized approaches are nowadays recognized more and more important due to the availability of large-scale data often organised over undirected or directed graphs. The authors provide a formal description of existing methods and, additionally, of the AB/Push-Pull introduced by themselves less than two years ago. Then, they develop a comparison of all methods and show that the last one can embed all other methods in a unified framework. Although the paper does not provide novel algorithms or methodologies to solve optimal control or estimation problems, it proposes an important review of existing methods and provides a rewriting of the AB/Push-Pull algorithms in view of applications and integration with previous methods. The paper is very well written and the formalization of methods introduced can be easily used to be applied in many contexts. Because of the above considerations, the paper can be published on the Proceedings of the IEEE.

Reply: We thank the reviewer for these constructive comments and all the valuable suggestions. The reviewer is correct that the paper does not have any new results. This is in spirit of the Proceedings of the IEEE and our goal is to write a reader-friendly overview paper that provides deep insights into the fundamental design and analysis of decentralized optimization algorithms.