Distributed Nesterov Gradient Methods Over Arbitrary Graphs
Ran Xin ©, Dušan Jakovetić ©, and Usman A. Khan ©

Abstract—In this letter, we introduce a distributed Nesterov gradient method, $AB\!N$, that does not require doubly stochastic weights. Instead, the implementation is based on a simultaneous application of both row- and column-stochastic weights that makes $AB\!N$ applicable to arbitrary (strongly-connected) graphs. Since constructing column-stochastic weights needs additional information (the number of outgoing neighbors), not available in certain communication protocols, we derive a variation, $FROZEN$, that only requires row-stochastic weights, but at the expense of additional iterations for eigenvector estimation. We numerically study these algorithms for various objective functions and network parameters and show that the proposed distributed Nesterov gradient methods achieve acceleration compared to the current state-of-the-art methods for distributed optimization.

Index Terms—Directed graphs, distributed algorithms, machine learning, optimization methods.

I. INTRODUCTION

DISTRIBUTED optimization has recently seen a surge of interest particularly with the emergence of modern signal processing and machine learning applications. A well-studied problem in this domain is finite sum minimization, i.e., $\min_{x} \sum_{i} f_{i}(x)$, where each $f_{i} : \mathbb{R}^{p} \rightarrow \mathbb{R}$ is a smooth and convex function locally available at an agent $i$. Since the $f_{i}$’s depend on data that may be private to each agent and communicating large data is impractical, developing distributed solutions of the above problem is fundamentally important. Related work has been a topic of significant research in the areas of signal processing and control [1]–[4], and more recently has also found coverage in the machine learning literature [5]–[10].

Since the focus is on distributed implementation, the inter-agent information exchange is modeled by a graph and significant work has focused on various graph topologies. Distributed optimization algorithms typically require two key steps: (i) consensus, i.e., reaching agreement among the agents; and (ii) optimality, i.e., showing that the agreement is on the optimal solution. Naturally, consensus algorithms are used as the basic building block of distributed optimization on top of which a gradient correction is added to steer the agreement to the optimal solution. Initial work thus follows closely the progress in consensus algorithms, see e.g., [5], [6], [11]–[15].

Early work on consensus assumes doubly-stochastic (DS) weights [16], [17], which require the underlying graphs to be undirected (or balanced). The subsequent work on distributed optimization over undirected graphs includes [11]–[14], where the convergence is sublinear, and [18]–[20] with linear convergence. For directed (and unbalanced) graphs, it is not possible to construct DS weights, i.e., the weights can be chosen such that they sum to 1 either only on incoming edges or only on outgoing edges. Distributed Optimization over digraphs [21]–[28] thus has been built on consensus with non-DS weights [29]–[31]. Required now is division with an additional variable that estimates the Perron eigenvector of the underlying weight matrix, see e.g., [21], [22], [25]–[28]. Such division may cause conservatism and stability issues [32].

Recently, we introduced the $AB$ algorithm that removes the need of Perron eigenvector estimation by utilizing both row-stochastic (RS) and column-stochastic (CS) weights [33]. It is shown in [33] that $AB$ converges linearly to the optimal for smooth and strongly-convex functions. In this letter, we propose accelerated distributed optimization over arbitrary strongly-connected graphs by extending $AB$ with Nesterov’s momentum. We first propose $AB\!N$ that uses both RS and CS weights. Construction of CS weights requires each agent to know at least its out-degree, which may not be possible in broadcast-type communication scenarios. To address this challenge, we propose the $FROZEN$ algorithm that only uses RS weights. We show that $FROZEN$ can be derived from $AB\!N$ with the help of a simple state transformation. Finally, we note that a rigorous theoretical analysis is beyond the scope of this letter and we present extensive simulations to highlight and verify different aspects of the proposed methods.

We now describe the rest of the letter. Section II formulates the problem and recaps the $AB$ algorithm. Section III describes the two methods, $AB\!N$ and $FROZEN$, and Section IV provides simulations comparing the proposed methods with the state-of-the-art methods for both general convex and strongly-convex functions over various digraphs.

II. PROBLEM FORMULATION AND PRELIMINARIES

Consider $n$ agents connected over a digraph, $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, \ldots, n\}$ is the set of agents and $\mathcal{E}$ is the collection of edges, $(i, j), i, j \in \mathcal{V}$, such that $j \rightarrow i$. We define $N_{i}^{in}$ as the collection of in-neighbors of agent $i$, i.e., the set of agents that can send information to agent $i$. Similarly, $N_{i}^{out}$ is the set of out-neighbors of agent $i$. Note that both $N_{i}^{in}$ and $N_{i}^{out}$...
include node $i$. The agents solve the following unconstrained optimization problem:

$$ P_1: \min_{x \in \mathbb{R}^p} F(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x), $$

where each $f_i : \mathbb{R}^p \rightarrow \mathbb{R}$ is private to agent $i$. We formalize the set of assumptions as follows.

**Assumption 1:** The graph, $G$, is strongly-connected but not necessarily undirected.

**Assumption 2:** Each local objective, $f_i$, is $\mu$-strongly-convex, i.e., $\forall i \in V$ and $\forall x, y \in \mathbb{R}^p$, we have, for some $\mu > 0$,

$$ f_i(y) \geq f_i(x) + \nabla f_i(x) (y - x) + \frac{\mu}{2} \| x - y \|^2. $$

**Assumption 3:** Each local objective, $f_i$, is $L$-smooth, i.e., its gradient is Lipschitz-continuous: $\forall i \in V$ and $\forall x, y \in \mathbb{R}^p$, we have, for some $L > 0$,

$$ \| \nabla f_i(x) - \nabla f_i(y) \| \leq L \| x - y \|. $$

Note that $\mu \leq L$. Let $F_{\mu,L}^{1,1}$ be the class of functions satisfying Assumption 3 and let $F_{\mu,L}^{1,1}$ be the class of functions that satisfy both Assumptions 2 and 3 [34]. In this letter, we propose distributed algorithms to solve Problem $P_1$ for both function classes, i.e., $F \in F_{\mu,L}^{1,1}$ and $F \in F_{\mu,L}^{1,1}$. We assume that Problem $P_1$ is solvable in the class $F_{\mu,L}^{1,1}$.

### A. Centralized Optimization: Nesterov’s Gradient Method

The gradient descent algorithm is given by

$$ x_{k+1} = x_k - \alpha \nabla F(x_k), $$

where $k$ is the iteration index and $\alpha$ is the step-size. It is well known [34], [35] that the oracle complexity of this method to achieve an $\epsilon$-accuracy is $O\left( \left( \frac{k}{\epsilon} \right)^\frac{1}{2} \right)$ for the function class $F_{\mu,L}^{1,1}$ and $O(k \log \frac{1}{\epsilon})$ for the function class $F_{\mu,L}^{1,1}$, where $k \triangleq \frac{\mu}{\alpha}$ is the condition number of $F$. There are gaps between the lower oracle complexity bounds of the function class $F_{\mu,L}^{1,1}$ and $F_{\mu,L}^{1,1}$ and the upper complexity bounds of gradient descent [34]. These gaps are closed by the seminal work [34] by Nesterov, which accelerates the convergence by adding a certain momentum to gradient descent. The centralized Nesterov’s gradient method [34] iteratively updates two variables $x_k, y_k \in \mathbb{R}^p$, initialized arbitrarily with $x_0 = y_0$, as follows:

$$ y_{k+1} = x_k - \frac{1}{L} \nabla F(x_k), $$

$$ x_{k+1} = y_{k+1} + \beta_k (y_{k+1} - y_k), $$

where $\beta_k$ is the momentum parameter. For the function class $F_{\mu,L}^{1,1}$, choosing $\beta_k = \frac{k}{k+3}$ leads to an optimal oracle complexity of $O\left( \frac{1}{\sqrt{\epsilon}} \right)$, while for the function class $F_{\mu,L}^{1,1}$, $\beta_k = \frac{\sqrt{\epsilon} - \sqrt{\pi}}{\sqrt{L} + \sqrt{\pi}}$ leads to an optimal oracle complexity of $O\left( \sqrt{\pi} \log \frac{1}{\epsilon} \right)$.

### B. Distributed Optimization: The $\mathcal{AB}$ Algorithm

Most existing work on distributed optimization [1]–[3], [11]–[14], [18]–[20] is restricted to undirected graphs. The work on directed graphs [21], [22], [25]–[28] requires Perron eigenvector estimation [29], [30]. Recently, the $\mathcal{AB}$ algorithm was introduced in [33] that does not require Perron eigenvector estimation and is described next. Consider two distinct sets of weights, $\{a_{ij}\}$ and $\{b_{lj}\}$, for each agent $i$ such that

$$ a_{ij} = \begin{cases} > 0, & j \in \mathcal{N}_i^m, \\ 0, & \text{otherwise} \end{cases}, \quad \sum_{j=1}^{n} a_{ij} = 1, \forall i, $$

$$ b_{lj} = \begin{cases} > 0, & l \in \mathcal{N}_i^\text{out}, \\ 0, & \text{otherwise} \end{cases}, \quad \sum_{l=1}^{m} b_{lj} = 1, \forall i. $$

Note that the weight matrix, $A = \{a_{ij}\}$, is row-stochastic (RS), while $B = \{b_{lj}\}$ is column-stochastic (CS). Using these weights, the $\mathcal{AB}$ algorithm performs the following iterations on two variables, $x^i_k, s^i_k \in \mathbb{R}^p$, at each agent $i$ [33]:

$$ x^i_{k+1} = \sum_{j \in \mathcal{N}_i^m} a_{ij} x^j_k - \alpha s^i_k, \quad (1a) $$

$$ s^i_{k+1} = \sum_{j \in \mathcal{N}_i^\text{out}} b_{lj} s^j_{k+1} + \nabla f_i (x^i_{k+1}) - \nabla f_i (x^i_k), \quad (1b) $$

where $x^i_k \in \mathbb{R}^p$ is arbitrary and $s^i_0 = \nabla f_i (x^i_0)$. The intuition behind the above equations is as follows: (1a) essentially is a descent method where the descent direction is $s^i_k$, instead of $\nabla f_i (x^i_k)$ as used in the earlier methods [11]–[14]. (1b), on the other hand, is gradient tracking, i.e., $s^i_k \rightarrow \frac{1}{m} \sum_{l=1}^{m} \nabla f_l (x^l_k)$, [18], [19], [26], [33], and thus (1a) takes a step in the global descent direction asymptotically. It is shown in [33] that $\mathcal{AB}$ converges linearly to the optimal solution for the function class $F_{\mu,L}^{1,1}$.

The construction of RS weights, $A$, is rather trivial as each agent $i$ can arbitrarily assign them. A common method to design CS weights is for each agent $i$ to transmit $s^i_k / |\mathcal{N}_i^\text{out}|$ to its outgoing neighbors in $\mathcal{N}_i^\text{out}$. This strategy, however, requires the knowledge of the out-degree at each agent $i$. Algorithms related to $\mathcal{AB}$ include [18], [19], [26], [36] over undirected graphs where both weight matrices, $A$ and $B$, are doubly-stochastic. Extensions of $\mathcal{AB}$ include: non-identical local step-sizes and heavy-ball momentum [32]; time-varying/random graphs [37], [38]; analysis for non-convex functions [39]. There is no prior work on Nesterov’s gradient method that is applicable to arbitrary strongly-connected graphs.

### III. DISTRIBUTED NESTEROV GRADIENT METHODS

In this section, we introduce two distributed Nesterov gradient methods applicable to arbitrary strongly-connected graphs.

#### A. The $\mathcal{ABN}$ Algorithm

Each agent, $i \in V$, maintains three variables: $x^i_k, y^i_k$ and $s^i_k$, all in $\mathbb{R}^p$, where $x^i_k$ and $y^i_k$ are the local estimates of the global minimizer and $s^i_k$ is used to track the average of local gradients. $\mathcal{ABN}$ is described in Algorithm 1.

A valid choice for $b_{lj}$’s at each $i$ is to choose them as $1/|\mathcal{N}_i^\text{out}|$, which does not require knowing the outgoing nodes but only the out-degree. For the function class $F_{\mu,L}^{1,1}$, $\beta$ is a constant; for the function class $F_{\mu,L}^{1,1}$, we choose $\beta_k = \frac{k}{k+3}, \forall k \geq 0$ [34].

#### B. The FROZEN Algorithm

Note that $\mathcal{ABN}$ requires the knowledge of out-degrees to implement column-stochastic (CS) weights, $\{b_{lj}\}$’s. To avoid CS weights, we now develop a distributed Nesterov gradient method
The method we described may not be applicable in some contexts. We first consider a distributed binary classification problem over arbitrary graphs, see [37], [38], [41]. Let $x_k, y_k, s_k, \nabla f(x_k)$ denote the concatenated vectors of $x_k^i, y_k^i, s_k^i, \nabla f(x_k^i)$, respectively. Then $ABN$ can be compactly written as follows:

$$y_{k+1} = Ax_k - \alpha s_k,$$  
(3a)

$$x_{k+1} = y_{k+1} + \beta_k (y_{k+1} - y_k),$$  
(3b)

$$s_{k+1} = Bs_k + \nabla f(x_{k+1}) - \nabla f(x_k),$$  
(3c)

where $A = A \otimes I_p, B = B \otimes I_p$, and $\otimes$ is the Kronecker product. Since $A$ is already RS, we seek a transformation that makes $B$ a RS matrix. Since $B$ is CS, we denote its right Perron eigenvector as $v$ such that $Bv = v$. Let $\text{diag}(v)$ denote a diagonal matrix with $v$ on its main diagonal. With the help of $V = \text{diag}(v) \otimes I_p$, we define a state transformation, $s_k = V^{-1}s_k$, and rewrite $ABN$ as

$$y_{k+1} = Ax_k - \alpha Vs_k,$$  
(4a)

$$x_{k+1} = y_{k+1} + \beta_k (y_{k+1} - y_k),$$  
(4b)

$$s_{k+1} = As_k + V^{-1} \left( \nabla f(x_{k+1}) - \nabla f(x_k) \right),$$  
(4c)

where $\tilde{A} = V^{-1}BV$ can be verified to be RS, such that

$$\tilde{A}1_{n_p} = 1_{n_p}, \quad (v \otimes 1_p)^\top \tilde{A} = (v \otimes 1_p)^\top.$$  

Since $v$ is the right Perron eigenvector of $B$, it is not locally known to any agent and thus the above equations are not practically possible to implement. We thus add an independent Perron eigenvector estimation algorithm to the above set equations and obtain $FROZEN$ (Fast Row-stochastic OptimiZation with momentum) described in Algorithm 2. The momentum parameter is chosen the same way as in $ABN$.

In the above algorithm, $e_i \in \mathbb{R}^{n_i}$ is a vector of zeros with a 1 at the $i$th location and $[\cdot]_i$ denotes the $i$th element of a vector. We note from (5a) that $[27], [28], [40]:$

$$\lim_{k \to \infty} v_k = v,$$

where $v$ is the left Perron vector of $\tilde{A} = \{\tilde{a}_{ij}\}$. Therefore, $FROZEN$, (5a)-(5d), asymptotically approaches to (4a)-(4c). We note that although the weights assignment in $FROZEN$ is straightforward, this flexibility comes at a price: (i) each agent must maintain an additional $n$-dimensional vector, $v_k^i$; (ii) additional iterations are required for Perron eigenvector estimation in (5a); and, (iii) the initial condition $v_0 = e_i$ requires each agent to have and know a unique identifier. However, as discussed earlier, $ABN$ may not be applicable in some communication protocols. Finally, we note that when $\beta_k = 0, \forall k$, $FROZEN$ reduces to FROST [27], [28].

Generalizations and extensions: The method we described to convert $ABN$ to $FROZEN$ leads to another variant of $ABN$ with only CS weights, see [33] for details. The resulting methods add Nesterov’s momentum to ADD-OPT/Push-DIGing [25], [26]. It is further straightforward to conceive an implementation of $ABN$ and $FROZEN$ over time-varying/random graphs, see e.g., the related work in [37], [38] on non-accelerated methods. Asynchronous schemes may also be derived following the methodologies studied in [41], [42]. Finally, we note that a rigorous theoretical analysis of $ABN$ and $FROZEN$ is beyond the scope of this letter. We thus rely on simulations to highlight and verify different aspects of the proposed methods.

IV. NUMERICAL RESULTS

In this section, we numerically verify the convergence of the proposed algorithms, $ABN$ and $FROZEN$, and compare them with existing (first-order) methods for distributed optimization. To this aim, we generate digraphs with $n = 30$ nodes using nearest-neighbor rules. The weights are designed as follows: $a_{ij} = 1/|N_{in}^i|$ and $b_{ij} = 1/|N_{out}^i|, \forall i,j$. We first compare $ABN$ and $FROZEN$ with the following methods over strongly-connected digraphs: ADD-OPT/Push-DIGing [25], [26], FROST [28], and $AB$ [33]. For comparison, we use the average residual, $\frac{1}{n} \sum_{i=1}^n \|x_i - \bar{x}\|_2$, versus the number of iterations as the performance measure.

Strongly-convex case: We first consider a distributed binary classification problem using logistic regression: each agent $i$ has access to $m_i$ training samples, $(c_{ij}, y_{ij}) \in \mathbb{R}^p \times \{-1, 1\}$, where $c_{ij}$ contains $p$ features of the $j$th training data at agent $i$, and $y_{ij}$ is the corresponding binary label. The agents minimize $F = \sum_{i=1}^n f_i(w, b)$, where $w \in \mathbb{R}^p, b \in \mathbb{R}$, and each $f_i$ is given by

$$f_i(w, b) = \sum_{j=1}^{m_i} \ln[1 + e^{-(c_{ij}^\top w + b)y_{ij}}] + \frac{\lambda}{2m} \|w\|^2_2.$$

Algorithm 1: $ABN$

At each agent $i$:

**Initialize:** Arbitrary $x_i^0 \in \mathbb{R}^p$ and $s_i^0 = \nabla f_i(x_i^0)$

**Choose:** $(a_{ij})$ with $\sum_j a_{ij} = 1$, and $(b_{ij})$ with $\sum_i b_{ij} = 1$

For $k = 0, 1, \ldots$, do

**Transmit:** $x_i^k$ and $b_{ij} s_i^k$ to each $l \in \mathcal{N}_i^{\text{out}}$

**Compute:**

$$y_{i+1} = \sum_{j \in \mathcal{N}_i} a_{ij} x_j^i - \alpha s_i^k,$$  
(2a)

$$x_{i+1} = y_{i+1} + \beta (y_{i+1} - y_i),$$  
(2b)

$$s_{i+1} = \sum_{j \in \mathcal{N}_i} b_{ij} s_j^i + \nabla f_i(x_{i+1}) - \nabla f_i(x_i^i),$$  
(2c)

end

Algorithm 2: $FROZEN$

At each agent $i$:

**Initialize:** Arbitrary $x_i^0 \in \mathbb{R}^p$, $s_i^0 = \nabla f_i(x_i^0)$, $v_0^i = e_i$

**Choose:** Two sets of row-stochastic weights, $(a_{ij}), (\tilde{a}_{ij})$

For $k = 1, \ldots$, do

**Broadcast:** $x_i^k, v_k^i, s_i^k$

**Compute:**

$$v_{k+1}^i = \sum_{j \in \mathcal{N}_i} \tilde{a}_{ij} v_{k}^j,$$  
(5a)

$$y_{k+1}^i = \sum_{j \in \mathcal{N}_i} a_{ij} x_j^i - \alpha s_k^i,$$  
(5b)

$$x_{k+1}^i = y_{k+1}^i + \beta_k (y_{k+1}^i - y_k^i),$$  
(5c)

$$s_{k+1}^i = \sum_{j \in \mathcal{N}_i} \tilde{a}_{ij} s_j^i + \frac{\nabla f_i(x_{k+1}^i)}{|v_{k+1}^i|} - \frac{\nabla f_i(x_k^i)}{|v_k^i|},$$  
(5d)

end
The feature vectors, $c_{ij}$’s, are randomly generated from two distinct Gaussian distributions corresponding to two different labels, +1 and −1. We set $p = 10$, $m_i = 20$, $\forall i$, and $\lambda = 10$. The performance comparison is shown in Fig. 1, where the step-size and momentum parameters are manually optimized. Recall that although $FROZEN$ is slower than $ABN$ due to Perron eigenvector estimation, it is applicable to broadcast protocols as it only requires row-stochastic weights.

**Non strongly-convex case:** We next choose $f_i$’s to be smooth, convex but not strongly-convex: $f_i(x) = u(x) + b_i^T x$, where $b_i$’s are randomly generated such that $b_n = -\sum_{i=1}^{n-1} b_i$, and $u(x)$ is chosen as follows [19]:

$$u(x) = \begin{cases} \frac{1}{4} x^4, & |x| \leq 1, \\ |x| - \frac{1}{4}, & |x| > 1. \end{cases}$$

Note that $F = \sum_i f_i$ is not strongly-convex as $F''(x^*) = 0$. The performance comparison is shown in Fig. 2, where the momentum parameter is chosen as $\beta_k = \frac{k-1}{k-3}$ for $ABN$ and $FROZEN$ while the step-sizes are manually optimized.

**Influence of graph sparsity:** We now study the influence of graph sparsity with the help of the logistic regression problem discussed earlier. We fix the number of nodes to $n = 30$ and randomly generate three nearest-neighbor digraphs, $G_1$, $G_2$ and $G_3$, with decreasing sparsity, see Fig. 3; the performance comparison is shown in Fig. 4. It can be seen that both $ABN$ and $FROZEN$ approach to the centralized Nesterov gradient descent (CNGD) as graph sparsity decreases.

**Acceleration of $ABN$ and $FROZEN$:** Finally, we study the acceleration achieved by $ABN$ and $FROZEN$ as the condition number of the global objective function increases. To this aim, we choose each local objective as a quadratic function, i.e., $f_i(x) = \frac{1}{2} x^T Q_i x + b_i^T x$, where $Q_i \in \mathbb{R}^{p \times p}$ is diagonal and the vector $b_i$ is randomly chosen. We tune the diagonal elements of $Q_i$’s to achieve the desired condition numbers and plot the corresponding number of iterations needed to reach an accuracy of $10^{-3}$ of the optimal solution for $AB$, $ABN$, and $FROZEN$. The results are shown in Figs. 5 and 6. We note that both $ABN$ and $FROZEN$ are more robust than $AB$ when the condition number increases. The trends for $ABN$ and $FROZEN$ are very similar except that $FROZEN$ has a larger scale due to the Perron eigenvector estimation.

**V. CONCLUSIONS**

In this letter, we present accelerated methods for distributed optimization based on Nesterov’s momentum over arbitrary graphs. The fundamental algorithm, $ABN$, uses both row- and column-stochastic weights. When column-stochastic weights are not possible to construct, we provide an alternate algorithm, $FROZEN$, that only uses row-stochastic weights, however, at the expense of slower convergence. Although a theoretical analysis is beyond the scope of this letter, we provide an extensive set of numerical results to study the behavior of the proposed methods for both convex and strongly-convex cases.
REFERENCES

[1] M. Rabbat and R. Nowak, “Distributed optimization in sensor networks,” in Proc. 3rd Int. Symp. Inf. Process. Sens. Netw., Berkeley, CA, USA, Apr. 2004, pp. 20–27.

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

[3] A. Mokhtari, W. Shi, Q. Ling, and A. Ribeiro, “A decentralized second-order method with exact linear convergence rate for consensus optimization,” IEEE Trans. Signal Inf. Process. Netw., vol. 2, no. 4, pp. 507–522, Dec. 2016.

[4] S. Safavi, U. A. Khan, S. Kar, and J. M. F. Moura, “Distributed localization: A linear theory,” Proc. IEEE, vol. 106, pp. 1204–1223, Jul. 2018.

[5] P. A. Forero, A. Cano, and G. B. Giannakis, “Consensus-based distributed support vector machines,” J. Mach. Learn. Res., vol. 11, pp. 1663–1707, 2010.

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

[7] H.-T. Wai, Z. Yang, Z. Wang, and M. Hong, “Multi-agent reinforcement learning via double averaging primal-dual optimization,” in Proc. Ann. Conf. Neural Inf. Process. Syst., 2018, pp. 9649–9660.

[8] 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 Proc. Ann. Conf. Neural Inf. Process. Syst., 2017, pp. 5330–5340.

[9] M. Hong, D. Hajinezhad, and M. Zhao, “Prox-PDA: The proximal primal-dual algorithm for fast distributed nonconvex optimization and learning over networks,” in Proc. Int. Conf. Mach. Learn., 2017, pp. 1529–1538.

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

[11] S. S. Ram, A. Nedić, and V. V. Veeravalli, “Distributed subgradient projection algorithm for convex optimization,” in IEEE Int. Conf. Acoust., Speech, Signal Process., Taipei, Taiwan, Apr. 2009, pp. 3653–3656.

[12] A. Nedić and A. Ozdaglar, “Distributed subgradient methods for multi-agent optimization,” IEEE Trans. Autom. Control, vol. 54, no. 1, pp. 48–61, Jan. 2009.

[13] K. Srivastava and A. Nedić, “Distributed asynchronous constrained stochastic optimization,” IEEE J. Sel. Topics Signal Process., vol. 5, no. 4, pp. 772–790, Aug. 2011.

[14] S. Lee and A. Nedić, “Distributed random projection algorithm for convex optimization,” IEEE J. Sel. Topics Signal Process., vol. 7, no. 2, pp. 221–234, Apr. 2013.

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

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

[17] L. Xiao and S. Boyd, “Fast linear iterations for distributed averaging,” Syst. Control Lett., vol. 53, no. 1, pp. 65–78, 2004.

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

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

[20] 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.

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

[22] C. Xi and U. A. Khan, “Distributed subgradient projection algorithm over directed graphs,” IEEE Trans. Autom. Control, vol. 62, no. 8, pp. 3986–3992, Oct. 2016.

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

[24] C. Xi, R. Xin, and U. A. Khan, “ADD-OPT: Accelerated distributed directed optimization,” IEEE Trans. Autom. Control, vol. 63, no. 5, pp. 1329–1339, May 2017.

[25] 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.

[26] C. Xi, V. S. Mai, R. Xin, E. H. Abed, and U. A. Khan, “Linear convergence in optimization over directed graphs with row-stochastic matrices,” IEEE Trans. Autom. Control, vol. 63, no. 10, pp. 3558–3565, Oct. 2018.

[27] R. Xin, C. Xi, and U. A. Khan, “FROST—Fast row-stochastic optimization with uncoordinated step-sizes,” EURASIP J. Adv. Signal Process., vol. 2019, 2019, Art. no. 1.

[28] D. Kempe, A. Dobra, and J. Gehrke, “Gossip-based computation of aggregate information,” in Proc. 44th Annu. IEEE Symp. Found. Comput. Sci., Oct. 2003, pp. 482–491.

[29] B. Polyak, Introduction to Optimization. New York, NY, USA: Springer, 1987.

[30] Y. Nesterov, Introductory Lectures on Convex Optimization: A Basic Course, vol. 87. New York, NY, USA: Springer, 2013.

[31] A. Nedić and A. Ozdaglar, “Distributed asynchronous constrained stochastic optimization,” IEEE J. Sel. Topics Signal Process., vol. 5, no. 4, pp. 772–790, Aug. 2011.

[32] S. Lee and A. Nedić, “Distributed random projection algorithm for convex optimization,” IEEE J. Sel. Topics Signal Process., vol. 7, no. 2, pp. 221–234, Apr. 2013.

[33] R. Xin, C. Xi, and U. A. Khan, “Distributed heavy-ball: A generalization and acceleration of first-order methods with gradient tracking,” 2018, arXiv:1808.02942.

[34] R. Xin and U. A. Khan, “A linear algorithm for optimization over directed graphs with geometric convergence,” IEEE Control Syst. Lett., vol. 2, no. 3, pp. 325–330, Jul. 2018.

[35] A. Nedić and A. Ozdaglar, “Distributed asynchronous constrained stochastic optimization,” IEEE J. Sel. Topics Signal Process., vol. 5, no. 4, pp. 772–790, Aug. 2011.

[36] S. Lee and A. Nedić, “Distributed random projection algorithm for convex optimization,” IEEE J. Sel. Topics Signal Process., vol. 7, no. 2, pp. 221–234, Apr. 2013.

[37] S. Safavi and U. A. Khan, “Revisiting finite-time distributed algorithms via successive nulling of eigenvalues,” IEEE Signal Process. Lett., vol. 22, no. 1, pp. 54–57, Jan. 2015.

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

[39] L. Xiao and S. Boyd, “Fast linear iterations for distributed averaging,” Syst. Control Lett., vol. 53, no. 1, pp. 65–78, 2004.

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

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