# Push-SAGA: A decentralized stochastic algorithm with variance reduction over directed graphs

Muhammad I. Qureshi[†], Ran Xin[‡], Soummya Kar[‡], and Usman A. Khan[†]
[†]Tufts University, Medford, MA, USA, [‡]Carnegie Mellon University, Pittsburgh, PA, USA

***Abstract*— In this paper, we propose Push-SAGA, a decentralized stochastic first-order method for finite-sum minimization over a directed network of nodes. Push-SAGA combines node-level variance reduction to remove the uncertainty caused by stochastic gradients, network-level gradient tracking to address the distributed nature of the data, and push-sum consensus to tackle directed information exchange. We show that Push-SAGA achieves linear convergence to the exact solution for smooth and strongly convex problems and is thus the first linearly-convergent stochastic algorithm over arbitrary strongly connected directed graphs. We also characterize the regime in which Push-SAGA achieves a linear speed-up compared to its centralized counterpart and achieves a network-independent convergence rate. We illustrate the behavior and convergence properties of Push-SAGA with the help of numerical experiments on strongly convex and non-convex problems.**

***Index Terms*— Stochastic optimization, variance reduction, first-order methods, decentralized algorithms, directed graphs**

## I. Introduction

We consider decentralized finite-sum minimization over a network of $n$ nodes, i.e.,

$$\mathbf{P}: \quad \min_{\mathbf{z}\in\mathbb{R}^p} F(\mathbf{z}) := \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{z}), \quad f_i(\mathbf{z}) := \frac{1}{m_i}\sum_{j=1}^{m_i} f_{i,j}(\mathbf{z}),$$

where each local cost $f_i : \mathbb{R}^p \to \mathbb{R}$, private to node $i$, is further decomposable into $m_i$ component cost functions. Problems of this nature commonly arise in many training and inference tasks where large-scale data is available over several geographically distributed nodes and the information exchange among the nodes is not bidirectional. Examples include robotic networks and the Internet of Things, where multiple robots or devices possess heterogeneous training data and may have different communication and battery resources resulting into one-directional communication.

This paper describes **Push-SAGA**, a *stochastic, first-order* method with a low per-iteration computation complexity, where the nodes communicate over *directed graphs* that are particularly amenable to efficient, resource-constrained network design and often result from severing costly communication links. Existing decentralized stochastic gradient methods over general directed graphs suffer from the variance of the stochastic gradients and the disparity between the local $f_i$ and global costs $F = \sum_i f_i$. To overcome these challenges, **Push-SAGA** utilizes *variance reduction*, locally at each node, to remove the uncertainty caused by the stochastic gradients, and *gradient tracking*, at the network level, to track the global gradient; see Fig. 1. Since the underlying graph is directed, **Push-SAGA** further uses the push-sum to enable agreement among the nodes with network weight matrices that are not necessarily doubly stochastic.

Decentralized stochastic gradient descent (**DSGD**) over undirected graphs can be found in [1]–[3]. The steady-state error in DSGD, due to the local vs. global cost dissimilarity, is further removed in **GT-DSGD** [4], [5], with the help of gradient tracking [6]–[10]. For $L$-smooth and $\mu$-strongly convex problems, **DSGD** and **GT-DSGD** converge linearly to an inexact solution with a constant stepsize (or sublinearly to the exact solution with a decaying stepsize) due to the variance of the stochastic gradients. Linear convergence to the exact solution is shown with the help of variance reduction [11]–[13]; see related work in [14]–[18]. Refs. [19], [20] propose a combination of gradient tracking and variance reduction to achieve optimal rates for decentralized stochastic non-convex problems. However, all of these methods require doubly stochastic weights and thus are not applicable to directed graphs. Related work that does not use doubly stochastic weights includes stochastic gradient push (**SGP**) [21]–[23], that extends **DSGD** to directed graphs with the help of push-sum consensus [24], and **S-ADDOPT** [25] that adds gradient tracking to **SGP**. For smooth and strongly convex problems, both **SGP** and **S-ADDOPT**, similar to their undirected counterparts **DSGD** and **GT-DSGD**, converge linearly to an *inexact* solution with a constant stepsize and sublinearly to the exact solution with decaying stepsizes. Of relevance are also **GP** [26] and **Push-DIGing/ADDOPT** [10], [27], which are non-stochastic counterparts of **SGP** and **S-ADDOPT** as they use local full (batch) gradients at each node.

The main contributions of this paper include: (i) We show that **Push-SAGA** converges linearly to the global optimizer of $\mathbf{P}$ for smooth and strongly convex problems; (ii) We characterize the performance of **Push-SAGA** in terms of an explicit directivity constant that quantifies the directed nature of the communication and reduces to 1 for undirected graphs; (iii) We develop regimes of practical relevance where **Push-SAGA** exhibits linear speedup and network-independent convergence.

The rest of the paper is organized as follows. We develop **Push-SAGA** in Section II whereas Section III describes the main results and contributions. Section IV provides the convergence analysis, while Section V contains numerical experiments on strongly convex and non-convex problems.

The authors acknowledge the support of NSF under awards CCF-1513936, CMMI-1903972, and CBET-1935555.

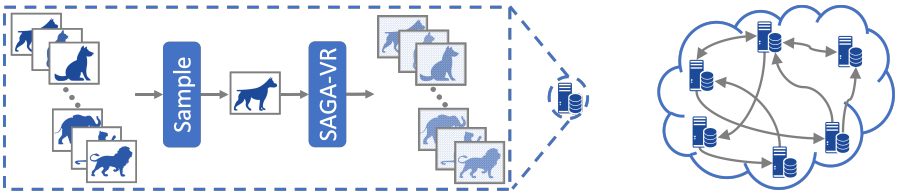


Fig. 1. (Left) Node-level: Each node computes the gradient at a random data sample and then estimates the local batch gradient with the help of variance reduction. (Right) Network-level: The nodes implement global gradient tracking with the help of inter-node fusion and push-sum.

## II. Motivation and Algorithm Development

In order to motivate **Push-SAGA**, we first describe **DSGD**, a decentralized extension of **SGD**, and its performance with a constant stepsize $\alpha$. Let $\mathbf{z}^*$ denote the global minimum of Problem $\mathbf{P}$ and let $\mathbf{x}_i^k \in \mathbb{R}^p$ denote the **DSGD** estimate of $\mathbf{z}^*$ at node $i$ and iteration $k$. Each node $i$ updates $\mathbf{x}_i^k$ as

$$\mathbf{x}_i^{k+1} = \sum_{r=1}^{n} w_{ir}\mathbf{x}_r^k - \alpha \cdot \nabla f_{i,s_i^k}(\mathbf{x}_i^k), \qquad k \geq 0, \quad (1)$$

where $\underline{W} = \{w_{ir}\} \in \mathbb{R}^{n\times n}$ is a network weight matrix such that $w_{ij} \neq 0$, if and only if, node $j$ can send information to node $i$, and $s_i^k$ is chosen uniformly at random from the set $\{1, \ldots, m_i\}$ at each iteration $k$. Assuming that each local cost is $L$-smooth and $\mu$-strongly convex and that each local stochastic gradient has a bounded variance, i.e.,

$$\mathbb{E}_{s_i^k}[\|\nabla f_{i,s_i^k}(\mathbf{x}_i^k) - \nabla f_i(\mathbf{x}_i^k)\|_2^2 \mid \mathbf{x}_i^k] \leq \sigma^2, \qquad \forall i, k,$$

it can be shown that, for a certain constant stepsize $\alpha$, the error $\mathbf{e}_i^k := \mathbb{E}[\|\mathbf{x}_i^k - \mathbf{z}^*\|_2^2]$, at each node $i$, decays at a linear rate of $(1 - \mathcal{O}(\mu\alpha))^k$ to a steady-state given by [28]

$$\limsup_{k\to\infty} \tfrac{1}{n}\sum_{i=1}^{n} \mathbf{e}_i^k = \mathcal{O}\Big( \tfrac{\alpha}{n\mu}\sigma^2 + \tfrac{\alpha^2 L^2}{\mu^2(1-\lambda)}\sigma^2 + \tfrac{\alpha^2 L^2}{\mu^2(1-\lambda)^2}\eta \Big), \quad (2)$$

where $\eta := \frac{1}{n}\sum_{i=1}^{n} \|\nabla f_i(\mathbf{z}^*)\|_2^2$ and $(1-\lambda)$ is the spectral gap of $\underline{W}$. Equation (2) is based on a constant stepsize $\alpha$ that leads to a linear but inexact convergence and our goal is to recover linear convergence to the exact solution.

We now consider the error terms in (2). The first two terms both depend on the variance $\sigma^2$ introduced due to the stochastic gradient and vanish as $\sigma^2 \to 0$; a variance reduction scheme that replaces the local stochastic gradients $\nabla f_{i,s_i^k}$, in **DSGD** (1), with an estimate of the local batch gradient $\sum_j \nabla f_{i,j}$ thus potentially removes this variance. The last term in (2) involves $\eta$, which quantifies the disparity between the local costs $f_i$'s and the global cost $F$ (recall that $\nabla F(\mathbf{z}^*) = 0$). A mechanism that uses the local gradient estimators (from the variance reduction step) to learn the global gradient thus removes $\eta$; this is realized with the help of dynamic average consensus [7]. In summary, adding local variance reduction and global gradient tracking to **DSGD** potentially leads to linear convergence for smooth and strongly convex problems. However, the weights $w_{ir}$ in **DSGD** are such that $\underline{W} = \{w_{ir}\}$ is doubly stochastic, which in general requires the underlying communication graph to be undirected.

In directed graphs, the weight matrix may either be row stochastic or column stochastic, in general, but not both at the same time. Consequently, the proposed method **Push-SAGA** uses primitive, column stochastic weights $\underline{B} = \{b_{ir}\} \in \mathbb{R}^{n\times n}$, for which it can be verified that the nodes do not reach agreement, i.e., $\underline{B}\mathbf{1}_n \neq \mathbf{1}_n$, where $\mathbf{1}_n$ is a column vector of $n$ ones. In fact, assuming $\boldsymbol{\pi}$ to be right eigenvector of $\underline{B}$ corresponding to the eigenvalue 1, the iterations $\mathbf{x}^{k+1} = \underline{B}\mathbf{x}^k \to \underline{B}^\infty\mathbf{x}^0 = \boldsymbol{\pi}\mathbf{1}_n^\top\mathbf{x}^0$, which only leads to an agreement among the components of $\mathbf{x}^k$, when its $i$-th component $\mathbf{x}_i^k$ is scaled by the $i$-th component $\boldsymbol{\pi}_i$ of $\boldsymbol{\pi}$. This asymmetry, caused by the non-$\mathbf{1}_n$ (right) eigenvector of $\underline{B}$, is removed with the help of the push-sum correction. In particular, push-sum estimates the non-$\mathbf{1}_n$ eigenvector $\boldsymbol{\pi}$ with $\mathbf{y}^{k+1} = \underline{B}\mathbf{y}^k$, $\mathbf{y}^0 = \mathbf{1}_n$; subsequently, each component of $\mathbf{x}^k$ is scaled by the corresponding component of $\mathbf{y}^k$ to obtain an agreement in the $\mathbf{x}_i^k/\mathbf{y}_i^k$ iterate. We note that $\mathbf{y}_i^k > 0, \forall k$, from the Perron Frobenius theorem [29].

**Algorithm Description:** The proposed **Push-SAGA** algorithm is formally described next, see also Fig. 1.

**Algorithm 1** **Push-SAGA** at each node $i$

**Require:** $\mathbf{x}_i^0 = \mathbf{z}_i^0 \in \mathbb{R}^p$, $\mathbf{w}_i^0 = \mathbf{g}_i^0 = \nabla f_i(\mathbf{z}_i^0)$, $\alpha > 0$,
Grad. table: $\{\nabla f_{i,j}(\mathbf{z}_i^0)\}_{j=1}^{m_i}$, $\{\mathbf{v}_{i,j}^1 = \mathbf{z}_i^0\}_{j=1}^{m_i}$

**for** $k = 0, 1, 2, \cdots$ **do**
  $\mathbf{x}_i^{k+1} \leftarrow \sum_{r=1}^{n} b_{ir}\mathbf{x}_r^k - \alpha \cdot \mathbf{w}_i^k$
  $y_i^{k+1} \leftarrow \sum_{r=1}^{n} b_{ir} y_r^k$
  $\mathbf{z}_i^{k+1} \leftarrow \mathbf{x}_i^{k+1}/y_i^{k+1}$
  **Select** $s_i^{k+1}$ uniformly at random from $\{1, \cdots, m_i\}$
  $\mathbf{g}_i^{k+1} \leftarrow \nabla f_{i,s_i^{k+1}}(\mathbf{z}_i^{k+1}) - \nabla f_{i,s_i^{k+1}}(\mathbf{v}_{i,s_i^{k+1}}^{k+1})$
    $+\frac{1}{m_i}\sum_{j=1}^{m_i} \nabla f_{i,j}(\mathbf{v}_{i,j}^{k+1})$
  $\nabla f_{i,s_i^{k+1}}(\mathbf{z}_i^{k+1}) \leftarrow \nabla f_{i,s_i^{k+1}}(\mathbf{v}_{i,s_i^{k+1}}^{k+1})$ in grad. table
  $\mathbf{w}_i^{k+1} \leftarrow \sum_{r=1}^{n} b_{ir}\mathbf{w}_r^k + \mathbf{g}_i^{k+1} - \mathbf{g}_i^k$
  **if** $j = s_i^{k+1}$, **then** $\mathbf{v}_{i,j}^{k+2} \leftarrow \mathbf{z}_i^{k+1}$, **else** $\mathbf{v}_{i,j}^{k+2} \leftarrow \mathbf{v}_{i,j}^{k+1}$
  **end if**
**end for**

We note that **Push-SAGA** has three main components: (i) *Variance reduction*, which utilizes the **SAGA**-based gradient estimator [12] to estimate the local batch gradient $\nabla f_i$ at each node $i$ from locally sampled gradients; (ii) *Gradient tracking*, which is based on dynamic average consensus [7] to estimate the global gradient $\nabla F$ from the local batch gradient estimates; and, (iii) *Push-sum consensus* [24], which

cancels the imbalance caused by the asymmetric nature of the underlying (directional) communication.

**Push-SAGA** requires a gradient table at each node $i$, where $m_i$ component gradients $\{\nabla f_{i,j}\}_{j=1}^{m_i}$ are stored. At each iteration $k$, each node $i$ first computes an **SGP**-type iterate $\mathbf{z}_k^i$ with the help of the push-sum correction. It is important to note that the descent direction in the $\mathbf{x}_k^i$-update (and thus in the $\mathbf{z}_k^i$-update) is $\mathbf{w}_k^i$, which is the global gradient tracker, in contrast to the locally sampled gradient as in **DSGD** (1). Subsequently, node $i$ generates a random index $s_i^k$ and computes the **SAGA**-based gradient estimator $\mathbf{g}_i^k$, with the help of the current iterate $\mathbf{z}_i^k$ and the elements from the gradient table. The gradient table is updated next only at the $s_i^k$-th element, while the other elements remain unchanged. Finally, these estimators $\mathbf{g}_i^k$'s are fused over the network, with the help of dynamic average consensus, to obtain $\mathbf{w}_i^k$'s that track the global gradient.

We remark that the computation and communication advantages of **Push-SAGA** are realized at an additional storage requirement. In particular, each node requires $\mathcal{O}(pm_i)$ storage that can be reduced to $\mathcal{O}(m_i)$ for certain problems [12]. The main results and the convergence analysis of **Push-SAGA** are provided next.

## III. Main Results and Contributions

The main results of **Push-SAGA** are described next.

**Theorem 1.** *Consider Problem* $\mathbf{P}$ *and let* $M := \max_i m_i$, $m := \min_i m_i$, *and each* $f_{i,j}$ *be* $L$*-smooth and each* $f_i$ *to be* $\mu$*-strongly convex. For the stepsize* $\alpha \in (0, \overline{\alpha})$, *for some* $\overline{\alpha} > 0$, ***Push-SAGA*** *linearly converges, at each node, to the global minimum* $\mathbf{z}^*$ *of* $F$. *In particular, for* $\alpha = \overline{\alpha}$, ***Push-SAGA*** *achieves an* $\epsilon$*-optimal solution in*

$$\mathcal{O}\left(\max\left\{M, \frac{M}{m}\frac{\kappa^2\psi}{(1-\lambda)^2}\right\}\log\frac{1}{\epsilon}\right),$$

*component gradient computations (in parallel) at each node, where* $\kappa := L/\mu$, $(1-\lambda)$ *is the spectral gap of the weight matrix, and* $\psi \geq 1$ *is a directivity constant.*

We explain the implications of the above theorem as follows.

**(1) Linear convergence. Push-SAGA** is the first linearly-convergent *stochastic* method to minimize a finite sum of smooth and strongly convex cost functions over arbitrary *directed* graphs. We emphasize that the analysis of **Push-SAGA** does not extend directly from the methods over undirected graphs. This is because: (i) the underlying weight matrices do not contract in the standard Euclidean norm; and, (ii) the algorithm has a nonlinear iterative component due to the push-sum update.

**(2) Directivity constant.** We quantify the directed nature of the graph with the help of an explicit directivity constant $\psi \geq 1$, which is 1 for undirected graphs, and thus, for finite-sum problems, this work includes **DSGD**, **SGP**, **GT-DSGD**, **S-ADDOPT**, and **GT-SAGA** as special cases.

**(3) Linear speedup and Network-independence.** In a big-data regime where $M \approx m \gg \kappa^2\psi(1-\lambda)^{-2}$, **Push-SAGA** with a complexity of $\mathcal{O}(M\log\frac{1}{\epsilon})$ is $n$ times faster than the centralized **SAGA**, and this convergence rate is further independent of the network parameters.

**(4) Improved Performance.** In the aforementioned big-data regime, **Push-SAGA** improves upon the related linearly-convergent methods [14], [15] over undirected graphs in terms of the joint dependence on $\kappa$ and $m$; with the exception of **DSBA** [16] and **ADFS** [17], both of which achieve a better iteration complexity at the expense of computing the proximal mapping at each iteration.

## IV. Convergence of Push-SAGA

This section provides the formal analysis of **Push-SAGA**. We start with the following assumptions.

**Assumption 1** (Column stochastic weights)**.** *The weight matrix* $\underline{B}$ *associated with the directed graph is primitive and column stochastic, i.e.,* $\mathbf{1}_n^\top \underline{B} = \mathbf{1}_n^\top$ *and* $\underline{B}\boldsymbol{\pi} = \boldsymbol{\pi}$, *where* $\mathbf{1}_n$ *is a vector of* $n$ *ones and* $\boldsymbol{\pi}$ *is the right (positive) eigenvector of* $\underline{B}$ *for the eigenvalue* $1$ *such that* $\mathbf{1}_n^\top\boldsymbol{\pi} = 1$.

Column-stochastic weights can be locally designed at each node by choosing $b_{ri} = 1/d_i^{\text{out}}$, where $d_i^{\text{out}}$ is the out-degree at node $i$. From Perron Frobenius theorem, we have $\underline{B}^\infty := \lim_{k\to\infty}\underline{B}^k = \boldsymbol{\pi}\mathbf{1}_n^\top$. Let $\|\cdot\|_2$ and $|||\cdot|||_2$ denote the standard vector 2-norm and the matrix norm induced by it, respectively, and define a weighted inner product as $\langle \mathbf{x}, \mathbf{z}\rangle_{\boldsymbol{\pi}} := \mathbf{x}^\top \text{diag}(\boldsymbol{\pi})^{-1}\mathbf{z}$, for $\mathbf{x}, \mathbf{z} \in \mathbb{R}^p$, which leads to a weighted Euclidean norm: $\|\mathbf{x}\|_{\boldsymbol{\pi}} := \|\text{diag}(\sqrt{\boldsymbol{\pi}})^{-1}\mathbf{x}\|_2$. We denote $|||\cdot|||_{\boldsymbol{\pi}}$ as the matrix norm induced by $\|\cdot\|_{\boldsymbol{\pi}}$ such that $\forall X \in \mathbb{R}^{n\times n}$, $|||X|||_{\boldsymbol{\pi}} := \left|\left|\left| \text{diag}(\sqrt{\boldsymbol{\pi}})^{-1} X \text{diag}(\sqrt{\boldsymbol{\pi}})\right|\right|\right|_2$. Under this induced norm, $\underline{B}$ contracts in the eigenspace orthogonal to the eigenspace corresponding to the eigenvalue 1, see [30] for formal arguments, i.e.,

$$\lambda := |||\underline{B} - \underline{B}^\infty|||_{\boldsymbol{\pi}} < 1 \Rightarrow (1-\lambda) < 1, \tag{3}$$

where $(1-\lambda)$ is the spectral gap of the weight matrix $\underline{B}$. Moreover, it can be verified from the norm definitions that $\|\cdot\|_{\boldsymbol{\pi}} \leq \underline{\pi}^{-0.5}\|\cdot\|_2$ and $\|\cdot\|_2 \leq \overline{\pi}^{0.5}\|\cdot\|_{\boldsymbol{\pi}}$, where $\overline{\pi}$ and $\underline{\pi}$ are the maximum and minimum elements of $\boldsymbol{\pi}$, respectively, while $|||\underline{B}|||_{\boldsymbol{\pi}} = |||\underline{B}^\infty|||_{\boldsymbol{\pi}} = |||I_n - \underline{B}^\infty|||_{\boldsymbol{\pi}} = 1$.

**Assumption 2** (Smooth and strongly convex cost functions)**.** *Each local cost* $f_i$ *is* $\mu$*-strongly convex and each component cost* $f_{i,j}$ *is* $L$*-smooth.*

We define the class of $\mu$-strongly convex and $L$-smooth functions as $\mathcal{S}_{\mu,L}$. It can be verified that $f_i \in \mathcal{S}_{\mu,L}$, $\forall i$, and $F \in \mathcal{S}_{\mu,L}$; consequently, $F$ has a global minimum that is denoted by $\mathbf{z}^* \in \mathbb{R}^p$. For any function in $\mathcal{S}_{\mu,L}$, we define its condition number as $\kappa := L/\mu$. Note that $\kappa \geq 1$.

### A. Auxiliary Results

We now write **Push-SAGA** in a vector-matrix format. To this aim, we define global vectors $\mathbf{x}^k, \mathbf{z}^k, \mathbf{g}^k, \mathbf{w}^k$, all in $\mathbb{R}^{pn}$, and $\mathbf{y}^k \in \mathbb{R}^n$ that concatenate the local vectors $\mathbf{x}_i^k, \mathbf{z}_i^k, \mathbf{g}_i^k, \mathbf{w}_i^k$, all in $\mathbb{R}^p$ and $\mathbf{y}_i^k \in \mathbb{R}$, respectively; whereas the global matrices are defined as $B := \underline{B} \otimes I_p$ and $Y_k := \text{diag}(\mathbf{y}^k) \otimes I_p$, both in $\mathbb{R}^{pn\times pn}$. We note that $Y_k$

is invertible for all $k$ and **Push-SAGA** in Algorithm 1 can now be equivalently written as

$$\mathbf{x}^{k+1} = B\mathbf{x}^k - \alpha \cdot \mathbf{w}^k, \tag{4a}$$
$$\mathbf{y}^{k+1} = \underline{B}\mathbf{y}^k, \tag{4b}$$
$$\mathbf{z}^{k+1} = Y_k^{-1}\mathbf{x}^{k+1}, \tag{4c}$$
$$\mathbf{w}^{k+1} = B\mathbf{w}^k + \mathbf{g}^{k+1} - \mathbf{g}^k. \tag{4d}$$

For analysis, we define four errors: (i) Network agreement error: $\mathbb{E}\|\mathbf{x}^k - B^\infty\mathbf{x}^k\|^2$; (ii) Optimality gap: $\mathbb{E}\|\overline{\mathbf{x}}^k - \mathbf{z}^*\|^2$, where $\overline{\mathbf{x}}^k := \frac{1}{n}(\mathbf{1}_n^\top \otimes I_p)\mathbf{x}^k$; (iii) Mean auxiliary gap: $\mathbb{E}[\mathbf{t}^k]$, where $\mathbf{t}^k := \sum_{i=1}^n(\frac{1}{m_i}\sum_{j=1}^{m_i}\|\mathbf{v}_{i,j}^k - \mathbf{z}^*\|_2^2)$; (iv) Gradient tracking error: $\mathbb{E}\|\mathbf{w}^k - B^\infty\mathbf{w}^k\|^2$. Lemma 2 next provides a relationship between these errors with the help of Lemma 1.

**Lemma 1.** *Consider Assumption 1 with $Y^\infty := \lim_{k\to\infty} Y_k$, then $\|\!|Y_k - Y^\infty|\!\|_2 \leq T\lambda^k, \forall k$, where $T := \sqrt{h}\|\mathbf{1}_n - n\boldsymbol{\pi}\|_2$ and $h := \overline{\pi}/\underline{\pi} > 1$.*

*Proof:* We note that $\forall k \geq 0,\ \mathbf{y}^\infty := \underline{B}^\infty\mathbf{y}^k$. Thus,

$$\begin{aligned}|\!|\!|Y_k - Y^\infty|\!|\!|_2 &\leq \sqrt{\overline{\pi}}|\!|\!|\underline{B} - \underline{B}^\infty|\!|\!|_{\boldsymbol{\pi}}\|\mathbf{y}^{k-1} - \mathbf{y}^\infty\|_{\boldsymbol{\pi}} \\ &\leq \lambda^k\sqrt{h}\|\mathbf{y}^0 - \mathbf{y}^\infty\|_2;\end{aligned}$$

and the proof follows [25], [31].

**Lemma 2.** *Consider **Push-SAGA** under Assumptions 1, 2, and let $y := \sup_k |\!|\!|Y_k|\!|\!|_2$, $y_- := \sup_k |\!|\!|Y_k^{-1}|\!|\!|_2$, and for all $k \geq 0$, define $\mathbf{u}^k, \mathbf{s}^k \in \mathbb{R}^4$ and $G_\alpha, H_k \in \mathbb{R}^{4\times 4}$ as*

$$\mathbf{u}^k := \begin{bmatrix} \mathbb{E}[\|\mathbf{x}^k - B^\infty\mathbf{x}^k\|_{\boldsymbol{\pi}}^2] \\ \mathbb{E}[n\|\overline{\mathbf{x}}^k - \mathbf{z}^*\|_2^2] \\ \mathbb{E}[\mathbf{t}^k] \\ \mathbb{E}[L^{-2}\|\mathbf{w}^k - B^\infty\mathbf{w}^k\|_{\boldsymbol{\pi}}^2] \end{bmatrix},\ \mathbf{s}^k := \begin{bmatrix} \mathbb{E}[\|\mathbf{x}^k\|_2^2] \\ 0 \\ 0 \\ 0 \end{bmatrix},$$

$$G_\alpha := \begin{bmatrix} \frac{1+\lambda^2}{2} & 0 & 0 & \frac{2\alpha^2L^2}{1-\lambda^2} \\ \frac{2\alpha L^2\psi\overline{\pi}}{\mu} & 1 - \frac{\alpha\mu}{2} & \frac{2\alpha^2L^2}{n} & 0 \\ \frac{2\psi\overline{\pi}}{m} & \frac{2}{m} & 1 - \frac{1}{M} & 0 \\ \frac{188\psi}{1-\lambda^2} & \frac{169\overline{\pi}^{-1}}{1-\lambda^2} & \frac{38\overline{\pi}^{-1}}{1-\lambda^2} & \frac{3+\lambda^2}{4} \end{bmatrix},$$

$$H_k := \begin{bmatrix} 0 & 0 & 0 & 0 \\ \frac{2\alpha L^2\psi}{\mu} & 0 & 0 & 0 \\ \frac{2\psi}{m} & 0 & 0 & 0 \\ \frac{188\psi^2}{1-\lambda^2} & 0 & 0 & 0 \end{bmatrix} T\lambda^k, \tag{5}$$

*where $m := \min_i m_i$, $M := \max_i m_i$, and $\psi := yy_-^2(1+T)h$ is defined as the directivity constant. For the stepsize $0 < \alpha \leq \frac{1-\lambda^2}{28L\kappa\psi}$, we have*

$$\mathbf{u}^k \leq G_\alpha\mathbf{u}^{k-1} + H_{k-1}\mathbf{s}^{k-1}. \tag{6}$$

See [32] for a formal proof of Lemma 2, which uses standard arguments from the gradient tracking literature. We note that for doubly stochastic weights, $T = 0$ since $\boldsymbol{\pi} = \frac{1}{n}\mathbf{1}_n$, while $\psi \geq 1$ can be interpreted as a directivity constant, i.e., for undirected graphs $\psi = 1$. Moreover, the LTI system described in (5) reduces to the one in [18] for undirected graphs, where $\psi = 1$ and $T = 0$. Our strategy to establish linear convergence of **Push-SAGA** is to show that $\mathbf{u}^k$ converges linearly to zero, which leads to $\mathbf{z}_i^k \to \mathbf{z}^*$ linearly at each node $i$. We establish the linear decay of $G_\alpha$ next.

**Lemma 3.** *Consider **Push-SAGA** under Assumptions 1, 2. If $\alpha \in (0, \overline{\alpha})$ is such that $\overline{\alpha} := \min\left\{\frac{1}{5M\mu}, \frac{m}{M}\frac{(1-\lambda)^2}{400L\kappa\psi}\right\}$, then $\rho(G_\alpha) \leq |\!|\!|G_\alpha|\!|\!|_\infty^{\boldsymbol{\delta}} \leq \gamma := 1 - \min\left\{\frac{1}{20M}, \frac{m}{1600M}\frac{(1-\lambda)^2}{\kappa^2\psi}\right\}$, where $\rho(\cdot)$ is the spectral radius and $|\!|\!|\cdot|\!|\!|_\infty^{\boldsymbol{\delta}}$ is the norm induced by the weighted max-norm $\|\cdot\|_\infty^{\boldsymbol{\delta}}$, for some $\boldsymbol{\delta} > \mathbf{0}_4$.*

The above lemma uses the fact that if $G_\alpha\boldsymbol{\delta} < \gamma\boldsymbol{\delta}$ for some $\boldsymbol{\delta} > \mathbf{0}_4$ and $\gamma = \left(1 - \frac{\alpha\mu}{4}\right)$, then $\rho(G_\alpha) < \gamma$; see [32] for the proof. Moreover, we have $\lambda \leq \rho(G_\alpha)$ implying that $H_k$ decays faster than $G_\alpha$. We next show that $\|\mathbf{u}^k\|_2 \to 0$.

**Lemma 4.** *Consider **Push-SAGA** under Assumptions 1, 2. For $\alpha \in (0, \overline{\alpha})$, we have that $\|\mathbf{u}^k\|_2 \to 0$ linearly at $\mathcal{O}((\gamma + \xi)^k)$, where $\lambda < 1$ and $\xi > 0$ is arbitrarily small.*

*Proof:* Expanding (6) and taking the norm leads to

$$\begin{aligned}\|\mathbf{u}^k\|_2 &\leq \|G_\alpha^k\mathbf{u}^0\|_2 + \textstyle\sum_{r=0}^{k-1}\|G_\alpha^{k-r-1}H_r\ \mathbf{s}^r\|_2, \\ &\leq \left(\Gamma_1 + \Gamma_2\textstyle\sum_{r=0}^{k-1}\|\mathbf{s}^r\|_2\right)\gamma^k,\end{aligned}$$

for some positive constants $\Gamma_1, \Gamma_2$. It can be shown that $\|\mathbf{s}^r\|_2 \leq 6(y^2 + \overline{\pi})\|\mathbf{u}^r\|_2 + 3y^2n\|\mathbf{z}^*\|_2^2$, which leads to (after using $b := 6\Gamma_2(y^2 + \overline{\pi})$ and $c := 3\Gamma_2y^2n\|\mathbf{z}^*\|_2^2$)

$$\|\mathbf{u}^k\|_2 \leq \left(\Gamma_1 + kc + b\textstyle\sum_{r=0}^{k-1}\|\mathbf{u}^r\|_2\right)\gamma^k.$$

Let $v_k := \sum_{r=0}^{k-1}\|\mathbf{u}^r\|_2, c_k := (\Gamma_1 + kc)\gamma^k$, and $b_k := b\gamma^k$. Then, we have $\|\mathbf{u}^k\|_2 = v_{k+1} - v_k \leq (\Gamma_1 + kc + bv_k)\gamma^k$, leading to $v_{k+1} \leq (1 + b_k)v_k + c_k$. For non-negative sequences $\{v_k\}, \{b_k\}$, and $\{c_k\}$, such that $v_{k+1} \leq (1 + b_k)v_k + c_k, \forall k$, $\sum_{k=0}^\infty b_k < \infty$, and $\sum_{k=0}^\infty c_k < \infty$, we have from [33] that $\{v_k\}$ converges and is thus bounded. Hence, $\forall\theta \in (\gamma, 1)$, we can write

$$\lim_{k\to\infty}\frac{\|\mathbf{u}^k\|_2}{\theta^k} \leq \lim_{k\to\infty}\frac{(\Gamma_1 + kc + bv_k)\gamma^k}{\theta^k} = 0.$$

In other words, $\exists\phi > 0$ such that $\|\mathbf{u}^k\|_2 \leq \phi(\gamma + \xi)^k, \forall k$, where $\xi > 0$ is arbitrarily small and the lemma follows.

With the help of the above, we next prove Theorem 1.

**Proof of Theorem 1:** Recall that $\mathbf{z}_i^k$ is the estimate of $\mathbf{z}^*$ at node $i$ and iteration $k$, and $\mathbf{z}^k$ concatenates these local estimates in a vector in $\mathbb{R}^{pn}$. We have that

$$\begin{aligned}\mathbb{E}[\|\mathbf{z}^k - (\mathbf{1}_n \otimes \mathbf{z}^*)\|_2^2] &\leq 3y_-^2\mathbb{E}[\|\mathbf{x}^k - B^\infty\mathbf{x}^k\|_2^2] \\ &\quad + 3ny_-^2y^2\mathbb{E}[\|\overline{\mathbf{x}}^k - \mathbf{z}^*\|_2^2] + 3n(y_-T\lambda^k)^2\|\mathbf{z}^*\|_2^2 \\ &\leq 6y_-^2(y^2 + \overline{\pi})\|\mathbf{u}^k\|_2 + 3ny_-^2T^2\gamma^k\|\mathbf{z}^*\|_2^2 \\ &\leq 6y_-^2(y^2 + \overline{\pi})\phi(\gamma + \xi)^k + 3ny_-^2T^2(\gamma + \xi)^k\|\mathbf{z}^*\|_2^2 \\ &\leq \omega(\gamma + \xi)^k,\end{aligned}$$

where $\omega := 6y_-^2(y^2 + \overline{\pi})\phi + 3ny_-^2T^2\|\mathbf{z}^*\|_2^2$. To reach an $\epsilon$-accurate solution $\mathbb{E}[\|\mathbf{z}^k - (\mathbf{1}_n \otimes \mathbf{z}^*)\|_2^2] \leq \epsilon$, we thus need

$$\mathbb{E}[\|\mathbf{z}^k - (\mathbf{1}_n \otimes \mathbf{z}^*)\|_2^2] \leq e^{-(1-(\gamma+\xi))k}\ \omega \leq \epsilon,$$

which leads to

$$k \geq \max\left\{\frac{20M}{1-20M\xi}, \frac{1600M\kappa^2\psi}{m(1-\lambda)^2 - 1600M\kappa^2\psi\xi}\right\}\ln\ \frac{\omega}{\epsilon},$$

iterations per node, where recall that $\xi > 0$ is arbitrarily small and the theorem follows.

## V. Numerical Experiments

We now provide numerical experiments to compare the performance of **Push–SAGA** with related algorithms implemented over directed graphs, shown in Fig. 2.

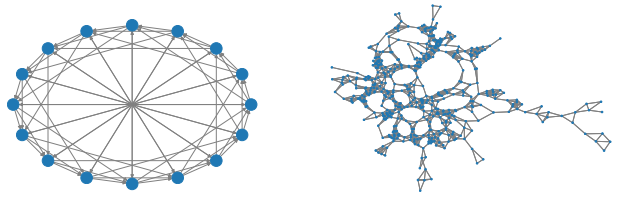

Fig. 2. (Left) Directed exponential graph with $n = 16$ nodes. (Right) geometric graph with $n = 500$ nodes.

**Logistic Regression (strongly convex)**: We first consider binary classification of $N = 12{,}000$ labelled images (taken from two classes in the MNIST and CIFAR-10 datasets) divided among $n$ nodes with the help of logistic regression with a strongly convex regularizer. We compare **SGP** (**DSGD** plus push-sum) [21], **S–ADDOPT** (**SGP** plus gradient tracking) [25], and **Push–SAGA**; along with **GP** [26] and **ADDOPT** [10], [27], which are the deterministic counterparts of **SGP** and **S–ADDOPT**, respectively. Note that **ADDOPT** (**GP** plus gradient tracking) converges linearly to the exact solution since it uses the full batch gradients at each node. We plot the optimality gap $F(\overline{\mathbf{z}}^k) - F(\mathbf{z}^*)$, where $\overline{\mathbf{z}}^k := \frac{1}{n}\sum_i \mathbf{z}_i^k$, of each algorithm versus the number of epochs, where each epoch represents $m_i$ gradient computations per node, i.e., one epoch is one iteration of **GP** and **ADDOPT** and $m_i$ iterations for the stochastic algorithms **SGP**, **S–ADDOPT**, and **Push–SAGA**. Fig. 3 (Top) compares the algorithms over an $n = 16$-node exponential graph, when the data is equally divided among the nodes, i.e., $m_i = 750, \forall i$. This scenario models a controlled training setup over a well-structured communication network as, e.g., in data centers. Similarly, Fig. 3 (Bottom) illustrates the performance over a $n = 500$-node geometric graph when the data is divided arbitrarily among the nodes modeling, e.g., ad hoc edge computing networks. It can be verified that **Push–SAGA** converges linearly for smooth and strongly convex problems and is much faster compared with its linearly-convergent non-stochastic counterpart **ADDOPT** over directed graphs.

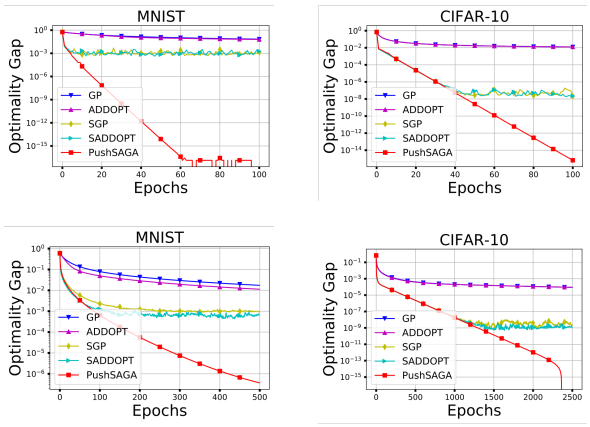


Fig. 3. Comparison over a directed exponential graph with $n = 16$ nodes (Top) and a directed geometric graph with $n = 500$ nodes (Bottom).

*Linear speed-up:* We next show the linear speed-up of **Push–SAGA** over its centralized counterpart **SAGA**. For illustration, **SGP** and **S–ADDOPT** are also compared with their centralized counterpart **SGD**. To this aim, we study binary classification based on the MNIST dataset over exponential graphs with $n = 4, 8, 16, 32$, and $64$ nodes, and plot the ratio of the number of iterations of the centralized algorithm and its decentralized counterpart to reach a certain optimality gap $\epsilon_1$: $\epsilon_1 = 10^{-15}$ for **Push–SAGA** and **SAGA**, since they linearly converge to the exact solution; while $\epsilon_1 = 10^{-3}$ for **SGP**, **S–ADDOPT**, and **SGD**, since they linearly converge to an error ball (with a constant stepsize). Recall that one iteration involves one gradient computation in centralized **SAGA** and $n$ (parallel) gradient computations in **Push–SAGA**. Fig. 4 (left) shows that **Push–SAGA**, per node, is $\mathcal{O}(n)$ times faster than **SAGA** thus acting effectively as a means for parallel computation.

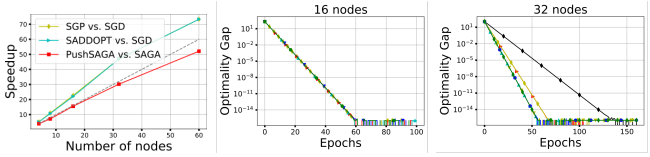


Fig. 4. (Left) Linear speedup: The plot shows **SGP** and **S–ADDOPT** vs. **SGD** to achieve an optimality gap of $10^{-3}$; **Push–SAGA** vs. **SAGA** to achieve an optimality gap of $10^{-15}$. (Middle and Right) Network-independent convergence for **Push–SAGA**: Each figure spans different graphs of varying connectivity while keeping the number of nodes $n$ fixed.

*Network independent convergence:* We now demonstrate the network-independent behavior of **Push–SAGA** in the big-data regime, i.e., when $M = m \gg \kappa^2\psi(1-\lambda)^{-2}$. For this purpose, we choose the binary classification problem based on the MNIST dataset, with $N = nm = 12{,}000$ total images equally divided over a network of $n$ nodes, and keep $\kappa \approx 1$ with an appropriate choice of the regularizer. We start with the base directed cycle and generate additional subsequent graphs by adding random directed edges until the graph is almost complete. The family of graphs in a fixed $n$-node setup thus ranges from least-connectivity (a directed cycle with $\lambda \to 1$) to improved connectivity by the addition of edges. For each family of graphs (fixed $n$), we plot the optimality gap of **Push–SAGA** in Fig. 4 (middle and

right), for $n = \{16, 32\}$ nodes that leads to $m = \{750, 375\}$ data samples per node. We observe that as long as $m$ is sufficiently larger than $\psi(1-\lambda)^{-2}$, the convergence of **Push–SAGA** is almost the same across all topologies generated by keeping a fixed number of nodes $n$. **Push–SAGA** loses network-independence for the $n = 32$-node network; this is because the big-data regime does not apply as it can be verified that $m = 375$ and $m \approx \psi(1-\lambda)^{-2}$.

**Neural Networks (non-convex)**: Finally, we compare the stochastic algorithms, i.e., **SGP**, **S–ADDOPT**, and **Push–SAGA**, for training a neural network over directed graphs. For each node, we construct a custom two-layered neural network comprising of one fully-connected, hidden layer of 64 neurons. We consider a multi-class classification problem on the MNIST and CIFAR-10 datasets with 10 classes each. Both datasets consist of 60,000 images in total and 6,000 images per class. The data samples are evenly distributed among the nodes that communicate over the 500-node geometric graph of Fig. 2 (right). We show the loss $F(\overline{\mathbf{z}}^k)$ and the test accuracy in Fig. 5. It can be observed that **Push–SAGA** shows improved performance compared to other methods particularly over the CIFAR-10 dataset.

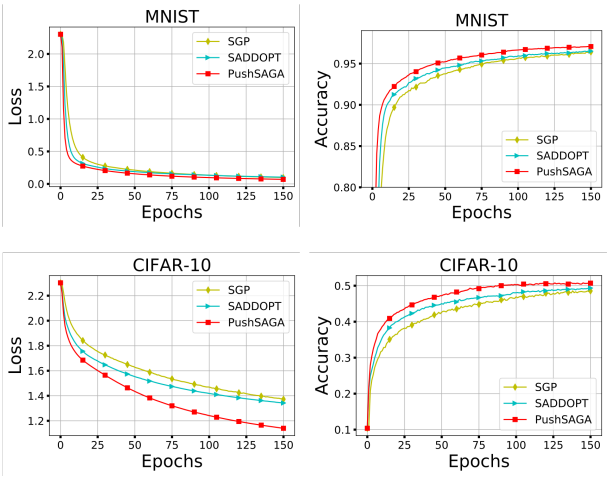


Fig. 5. Two-layer neural network over a 500-node geometric digraph.

## VI. Conclusion

This paper proposes a first-order decentralized stochastic method **Push–SAGA** for finite-sum minimization over arbitrary directed graphs. **Push–SAGA** uses variance reduction to eliminate the uncertainty caused by stochastic gradients and employs gradient tracking to address the distributed nature of data. We show that **Push–SAGA** linearly converges to the optimal solution for smooth and strongly convex problems. Moreover, we characterize the regime in which the convergence rate is network-independent and **Push–SAGA** achieves a linear speed-up compared to its centralized counterpart. Numerical experiments illustrate the performance comparison of **Push–SAGA** with corresponding methods for strongly convex and non-convex problems.

## References


[1] S. S. Ram, A. Nedić, and V. V. Veeravalli, "Distributed stochastic subgradient projection algorithms for convex optimization," *J. Optimiz. Theory App.*, vol. 147, no. 3, pp. 516–545, 2010.

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

[3] 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 *30th Advances in NeurIPS*, 2017, pp. 5330–5340.

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

[5] R. Xin, U. A. Khan, and S. Kar, "An improved convergence analysis for decentralized online stochastic non-convex optimization," *IEEE Trans. on Sig. Proc.*, Feb. 2021.

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

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

[8] P. D. Lorenzo and G. Scutari, "NEXT: in-network nonconvex optimization," *IEEE Trans. on Sig. and Inf. Proc. over Net.*, vol. 2, no. 2, pp. 120–136, 2016.

[9] G. Qu and N. Li, "Harnessing smoothness to accelerate distributed optimization," *IEEE Trans. on Cont. of Net. Sys.*, vol. 5, no. 3, pp. 1245–1260, 2017.

[10] A. Nedić, A. Olshevsky, and W. Shi, "Achieving geometric convergence for distributed optimization over time-varying graphs," *SIAM Journal on Optimization*, vol. 27, no. 4, pp. 2597–2633, 2017.

[11] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," in *Advances in NeurIPS*, 2013, pp. 315–323.

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

[13] L. M. Nguyen, J. Liu, K. Scheinberg, and M. Takáč, "SARAH: A novel method for machine learning problems using stochastic recursive gradient," in *Proc. of the 34th Int. Conf. on Machine Learning*. JMLR. org, 2017, pp. 2613–2621.

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

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

[16] Z. Shen, A. Mokhtari, T. Zhou, P. Zhao, and H. Qian, "Towards more efficient stochastic decentralized learning: Faster convergence and sparse communication," in *Proc. of the 35th Int. Conf. on Machine Learning*, Jul. 2018, vol. 80, pp. 4624–4633.

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

[18] R. Xin, U. A. Khan, and S. Kar, "Variance-reduced decentralized stochastic optimization with accelerated convergence," *IEEE Trans. on Sig. Proc.*, vol. 68, pp. 6255–6271, 2020.

[19] R. Xin, U. A. Khan, and S. Kar, "A near-optimal stochastic gradient method for decentralized non-convex finite-sum optimization," *arXiv:2008.07428*, 2020.

[20] R. Xin, U. A. Khan, and S. Kar, "A fast randomized incremental gradient method for decentralized non-convex optimization," *arXiv:2011.03853*, 2020.

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

[22] M. Assran, N. Loizou, N. Ballas, and M. G. Rabbat, "Stochastic gradient-push for distributed deep learning," in *Proc. of the 36th Int. Conf. on Machine Learning*, Jun. 2019, vol. 97, pp. 344–353.

[23] A. Spiridonoff, A. Olshevsky, and I. C. Paschalidis, "Robust asynchronous stochastic gradient-push: Asymptotically optimal and network-independent performance for strongly convex functions," *J. of Machine Learning Research*, vol. 21, no. 58, pp. 1–47, 2020.

[24] 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.
[25] M. I. Qureshi, R. Xin, S. Kar, and U. A. Khan, "S-ADDOPT: Decentralized stochastic first-order optimization over directed graphs," *IEEE Cont. Sys. Letters*, vol. 5, no. 3, pp. 953–958, 2021.
[26] K. I. Tsianos, S. Lawlor, and M. G. Rabbat, "Push-sum distributed dual averaging for convex optimization," in *51st IEEE Conf. on Dec. and Cont.*, Maui, Hawaii, Dec. 2012, pp. 5453–5458.
[27] C. Xi, R. Xin, and U. A. Khan, "ADD-OPT: Accelerated distributed directed optimization," *IEEE Trans. on Automatic Control*, vol. 63, no. 5, pp. 1329–1339, 2017.
[28] K. Yuan, S. A. Alghunaim, B. Ying, and A. H. Sayed, "On the performance of exact diffusion over adaptive networks," in *58th IEEE Conf. on Dec. and Cont.*, 2019, pp. 4898–4903.
[29] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, Cambridge, 2nd edition, 2012.
[30] R. Xin, A. K. Sahu, U. A. Khan, and S. Kar, "Distributed stochastic optimization with gradient tracking over strongly-connected networks," in *58th IEEE Conf. on Dec. and Cont.*, Dec. 2019, pp. 8353–8358.
[31] A. Nedić and A. Olshevsky, "Distributed optimization over time-varying directed graphs," *IEEE Trans. on Automatic Control*, vol. 60, no. 3, pp. 601–615, 2014.
[32] M. I. Qureshi, R. Xin, S. Kar, and U. A. Khan, "Push-SAGA: A decentralized stochastic algorithm with variance reduction over directed graphs," *arXiv:2008.06082*, 2020.
[33] B. Polyak, *Introduction to optimization*, Optimization Software, 1987.