2024 American Control Conference (ACC)
July 8-12, 2024. Toronto, Canada

Heterogeneous Distributed Subgradient

Yixuan Lin Marco Gamarra Ji Liu

***Abstract*— The paper proposes a heterogeneous push-sum based subgradient algorithm for multi-agent distributed convex optimization, in which each agent can arbitrarily switch between subgradient-push and push-subgradient at any time. It is shown that the heterogeneous algorithm converges to an optimal point at an optimal rate over time-varying directed graphs. The switching process within the heterogeneous algorithm can help prevent the leakage of agents' subgradient information.**

I. Introduction

Stemming from the pioneering work by Nedić and Ozdaglar [1], distributed optimization for multi-agent systems has attracted considerable interest and achieved great success in both theory and practice. Surveys of this area can be found in [2]–[4]. A typical distributed optimization problem is formulated as follows.

Consider a multi-agent network consisting of n agents, labeled 1 through n for the purpose of presentation. Every agent is not conscious of such a global labeling, but is capable of distinguishing between its neighbors. The neighbor relations among the n agents are characterized by a possibly time-dependent directed graph $\mathbb{G}(t) = (\mathcal{V}, \mathcal{E}(t))$ whose vertices correspond to agents and whose directed edges (or arcs) depict neighbor relations, where $\mathcal{V} = \{1, \dots, n\}$ is the vertex set and $\mathcal{E}(t) \subset \mathcal{V} \times \mathcal{V}$ is the directed edge set at time t. To be more precise, agent j is an in-neighbor of agent i at time t if $(j, i) \in \mathcal{E}(t)$, and similarly, agent k is an out-neighbor of agent i at time t if $(i, k) \in \mathcal{E}(t)$. The directions of arcs represent the directions of information flow in that each agent can send information to its out-neighbors and receive information from its in-neighbors. For convenience, we assume that each agent is always an in- and out-neighbor of itself, implying that $\mathbb{G}(t)$ has self-arcs at all vertices for any time t. We use $\mathcal{N}_i(t)$ and $\mathcal{N}_i^-(t)$ to denote the in- and out-neighbor set of agent i at time t, respectively, i.e.,

$$\mathcal{N}_i(t) = \{j \in \mathcal{V} : (j, i) \in \mathcal{E}(t)\},$$
$$\mathcal{N}_i^-(t) = \{k \in \mathcal{V} : (i, k) \in \mathcal{E}(t)\}.$$

It is easy to see that $\mathcal{N}_i(t)$ and $\mathcal{N}_i^-(t)$ are always nonempty since they both contain index i. The goal of the n agents is to cooperatively minimize the cost function

$$f(z) = \frac{1}{n} \sum_{i=1}^{n} f_i(z)$$

in which each $f_i : \mathbb{R}^d \to \mathbb{R}$ is a "private" convex (not necessarily differentiable) function only known to agent i. It is assumed that the set of optimal solutions to f, denoted by $\mathcal{Z}$, is nonempty and bounded.

To solve the distributed optimization problem just described, efforts have been made to design distributed multi-agent versions for various optimization algorithms, including the subgradient method [1], alternating direction method of multipliers (ADMM) [5], Nesterov accelerated gradient method [6], and proximal gradient descent [7], to name a few. Most existing distributed optimization algorithms require that the underlying communication graph be bi-directional or balanced[1], which allows a distributed manner to construct a doubly stochastic matrix [9], [10]. To tackle more general, unbalanced, directed graphs, the push-sum based algorithms have been proposed, with subgradient-push [11] being the first one, including notable DEXTRA [12] (a push-sum based variant of the well-known EXTRA algorithm [13]) and Push-DIGing [14]. Another approach to deal with unbalanced directed graphs is called push-pull [15], [16] while its state-of-the-art analysis assumes strongly connectedness at each time instance or relies on a carefully choosen small stepsize [17]. Push-sum is thus the most popular and probably the most powerful existing approach to design distributed (optimization) algorithms over time-varying directed graphs.

All the existing distributed optimization algorithms are homogeneous in that all the agents in a multi-agent network perform the same (order of) operations. Certain types of heterogeneity have recently been considered and incorporated in algorithm design. Examples include heterogeneous (uncoordinated) stepsize design for a gradient tracking method [18], [19], heterogeneous algorithm picking due to the coexistence of different types of agent dynamics in the network (e.g., a mix of continuous- and discrete-time dynamic agents) [20], and, particularly popular in machine learning, heterogeneous data training for distributed stochastic optimization [21]. Notwithstanding this, every agent in these algorithms has to adhere to a single protocol, without theoretical guarantee if any deviation from the protocol occurs.

With these in mind, this paper aims to design a heterogeneous distributed optimization algorithm in which each agent can change its protocol. To be more precise, the

The work of Y. Lin and J. Liu was supported in part by the National Science Foundation (NSF) under grant 2230101, by the Air Force Office of Scientific Research (AFOSR) under award number FA9550-23-1-0175, and by U.S. Air Force Task Order FA8650-23-F-2603. Y. Lin is currently with Meta and was previously affiliated with the Department of Applied Mathematics and Statistics at Stony Brook University. M. Gamarra is with the Air Force Research Laboratory (marco.gamarra@us.af.mil). J. Liu is with the Department of Electrical and Computer Engineering at Stony Brook University (ji.liu@stonybrook.edu).

[1] A weighted directed graph is called balanced if the sum of all in-weights equals the sum of all out-weights at each of its vertices [8].

iterative algorithm to be proposed will allow each agent to independently decide its order of operations in any iteration. To illustrate the idea, we focus on the subgradient-push method, and expect that the idea also works for other push-sum based first-order optimization methods.

It turns out that the proposed heterogeneous push-sum based distributed subgradient algorithm not only converges to an optimal point at an optimal convergence rate for time-varying directed graphs, but also prevents the leakage of an agent's private subgradient information through its operation switching process. Therefore, this paper proposes an operationally flexible and privacy preserving distributed optimization algorithm for convex functions.

II. Subgradient-Push and Push-Subgradient

We begin with the subgradient-push algorithm proposed in [11]. The subgradient method was first proposed in [22] for convex but not differentiable functions. For such a convex function $h : \mathbb{R}^d \to \mathbb{R}$, a vector $g \in \mathbb{R}^d$ is called a subgradient of h at point x if

$$h(y) \geq h(x) + g^\top (y - x) \ \text{ for all } \ y \in \mathbb{R}^d. \tag{1}$$

Such a vector g always exists for any x and may not be unique. In the special case when h is differentiable at x, the subgradient g is unique and equals the gradient of h at x. From (1) and the Cauchy-Schwarz inequality,

$$h(y) - h(x) \geq -G\|y - x\|,$$

where $\|\cdot\|$ denotes the 2-norm and G is an upper bound for the 2-norm of the subgradients of h at both x and y.

The subgradient-push algorithm is as follows[2]:

$$x_i(t+1) = \sum_{j \in \mathcal{N}_i(t)} w_{ij}(t)\Big[x_j(t) - \alpha(t) g_j(t)\Big], \tag{2}$$

$$y_i(t+1) = \sum_{j \in \mathcal{N}_i(t)} w_{ij}(t) y_j(t), \qquad y_i(0) = 1, \tag{3}$$

where $\alpha(t)$ is the stepsize, $g_j(t)$ is a subgradient of $f_j(z)$ at $x_j(t)/y_j(t)$, and $w_{ij}(t)$, $j \in \mathcal{N}_i(t)$, are positive weights satisfying the following assumption.

Assumption 1: There exists a constant $\beta > 0$ such that for all $i, j \in \mathcal{V}$ and t, $w_{ij}(t) \geq \beta$ whenever $j \in \mathcal{N}_i(t)$. For all $i \in \mathcal{V}$ and t, $\sum_{j \in \mathcal{N}_i^-(t)} w_{ji}(t) = 1$.

A simple choice of $w_{ij}(t)$ is $1/|\mathcal{N}_j^-(t)|$ for all $j \in \mathcal{N}_i(t)$ which can be easily computed in a distributed manner and satisfies Assumption 1 with $\beta = 1/n$. Thus, push-sum based algorithms require that each agent be aware of the number of its out-neighbors.

Let $W(t)$ be the $n \times n$ matrix whose ijth entry equals $w_{ij}(t)$ if $j \in \mathcal{N}_i(t)$ and zero otherwise; in other words, we set $w_{ij}(t) = 0$ for all $j \notin \mathcal{N}_i(t)$. Assumption 1 implies that $W(t)$ is a column stochastic matrix[3] with positive diagonal entries whose zero-nonzero pattern is compliant with the neighbor graph $\mathbb{G}(t)$ for all time t.

In implementation, at each time t, each agent j transmits two pieces of information, $w_{ij}(t)[x_j(t) - \alpha(t)g_j(t)]$ and $w_{ij}(t)y_j(t)$, to its out-neighbour i, and then each agent i updates its two variables as above. Note that if all $\alpha(t)g_j(t) = 0$, the algorithm simplifies to the push-sum algorithm [23]. Thus, at each time, each agent first performs a subgradient operation, and then follows the push-sum updates. This is why the algorithm (2)–(3) is called subgradient-push. It has been recently proved that subgradient-push converges at a rate of $O(1/\sqrt{T})$ over time-varying unbalanced directed graphs, which is the same as that of the single-agent subgradient and thus optimal [24].

Note that in the subgradient-push algorithm, all the agents in a multi-agent network perform the same order of operations, namely an optimization step (subgradient) followed by the push-sum updates. In this paper, we aim to relax this order restriction. To this end, we first introduce a variant of subgradient-push in which the order of subgradient and push-sum operations is swapped. To be more precise, each agent i updates its variables as

$$x_i(t+1) = \sum_{j \in \mathcal{N}_i(t)} w_{ij}(t) x_j(t) - \alpha(t) g_i(t), \tag{4}$$

$$y_i(t+1) = \sum_{j \in \mathcal{N}_i(t)} w_{ij}(t) y_j(t), \qquad y_i(0) = 1, \tag{5}$$

where $\alpha(t)$, $w_{ij}(t)$, and $g_i(t)$ are the same as those in subgradient-push. In the above algorithm (4)–(5) each agent i performs the push-sum updates first for both variables and then the subgradient update for x_i variable. We thus call the algorithm push-subgradient, which has never been studied, although its convergence may be analyzed similarly to that of the subgradient-push.

Push-subgradient can achieve the same performance as subgradient-push, namely, it converges to an optimal solution at a rate of $O(1/\sqrt{T})$ for general convex functions over time-varying unbalanced directed graphs. It turns out that both push-subgradient and subgradient-push are special cases of the following heterogeneous algorithm.

III. Heterogeneous Subgradient

Let $\sigma_i(t)$ be a switching signal of agent i which takes values in $\{0, 1\}$. At each time t, each agent j transmits two pieces of information, $w_{ij}(t)[x_j(t) - \alpha(t)g_j(t)\sigma_j(t)]$ and $w_{ij}(t)y_j(t)$, to its out-neighbour i, and then each agent i updates its variables as follows:

$$x_i(t+1) = \sum_{j \in \mathcal{N}_i(t)} w_{ij}(t)\Big[x_j(t) - \alpha(t) g_j(t)\sigma_j(t)\Big] - \alpha(t) g_i(t)\big(1 - \sigma_i(t)\big), \quad x_i(0) \in \mathbb{R}^d, \tag{6}$$

$$y_i(t+1) = \sum_{j \in \mathcal{N}_i(t)} w_{ij}(t) y_j(t), \qquad y_i(0) = 1, \tag{7}$$

where $\alpha(t)$ is the stepsize, $w_{ij}(t)$, $j \in \mathcal{N}_i(t)$, are positive weights satisfying Assumption 1.

[2]The algorithm is written in a different but mathematically equivalent form in [11].

[3]A square nonnegative matrix is called a column stochastic matrix if its column sums all equal one.

In the case when all $\sigma_i(t) = 1$, $i \in \mathcal{V}$, the above algorithm simplifies to the subgradient-push algorithm (2)–(3). In the case when all $\sigma_i(t) = 0$, $i \in \mathcal{V}$, the above algorithm simplifies to the push-subgradient algorithm (4)–(5). Thus, the algorithm (6)–(7) allows each agent to arbitrarily switch between subgradient-push and push-subgradient at any time, and we hence call it *heterogeneous distributed subgradient*.

To state the convergence result of the heterogeneous subgradient algorithm just proposed, we need the following typical assumption and concept.

Assumption 2: The step-size sequence $\{\alpha(t)\}$ is positive, non-increasing, and satisfies $\sum_{t=0}^{\infty}\alpha(t) = \infty$ and $\sum_{t=0}^{\infty}\alpha^2(t) < \infty$.

We say that an infinite directed graph sequence $\{\mathbb{G}(t)\}$ is uniformly strongly connected if there exists a positive integer L such that for any $t \geq 0$, the union graph $\cup_{k=t}^{t+L-1}\mathbb{G}(k)$ is strongly connected.[4] If such an integer exists, we sometimes say that $\{\mathbb{G}(t)\}$ is uniformly strongly connected by sub-sequences of length L. It is not hard to prove that the above definition is equivalent to the two popular joint connectivity definitions in consensus literature, namely "B-connected" [25] and "repeatedly jointly strongly connected" [26].

Define $z_i(t) = x_i(t)/y_i(t)$ for all $i \in \mathcal{V}$ and $\bar{z}(t) = \frac{1}{n}\sum_{i=1}^{n} z_i(t)$. It is easy to see that at the initial time $z_i(0) = x_i(0)$ for all $i \in \mathcal{V}$ and $\bar{z}(0) = \frac{1}{n}\sum_{i=1}^{n} x_i(0)$.

The following theorem shows that the heterogeneous distributed subgradient algorithm (6)–(7) still achieves the optimal rate of convergence to an optimal point.

Theorem 1: Suppose that $\{\mathbb{G}(t)\}$ is uniformly strongly connected and $\|g_i(t)\|$ is uniformly bounded for all i and t.

1) If the stepsize $\alpha(t)$ is time-varying and satisfies Assumption 2, then with $z^* \in \mathcal{Z}$,

$$\lim_{t\to\infty} f\bigg(\frac{\sum_{\tau=0}^{t}\alpha(\tau)\bar{z}(\tau)}{\sum_{\tau=0}^{t}\alpha(\tau)}\bigg) = f(z^*),$$
$$\lim_{t\to\infty} f\bigg(\frac{\sum_{\tau=0}^{t}\alpha(\tau)z_k(\tau)}{\sum_{\tau=0}^{t}\alpha(\tau)}\bigg) = f(z^*), \quad k \in \mathcal{V}.$$

2) If the stepsize is fixed and $\alpha(t) = 1/\sqrt{T}$ for $T > 0$ steps, i.e., $t \in \{0, 1, \ldots, T-1\}$, then with $z^* \in \mathcal{Z}$,

$$f\bigg(\frac{\sum_{\tau=0}^{T-1}\bar{z}(\tau)}{T}\bigg) - f(z^*) \leq O\Big(\frac{1}{\sqrt{T}}\Big),$$
$$f\bigg(\frac{\sum_{\tau=0}^{T-1}z_k(\tau)}{T}\bigg) - f(z^*) \leq O\Big(\frac{1}{\sqrt{T}}\Big), \quad k \in \mathcal{V}.$$

It is easy to show that the above theorem is a consequence of the following theorem.

Theorem 2: Suppose that $\{\mathbb{G}(t)\}$ is uniformly strongly connected by sub-sequences of length L and that $\|g_i(t)\|$ is uniformly bounded above by a positive number G for all $i \in \mathcal{V}$ and $t \geq 0$.

1) If the stepsize $\alpha(t)$ is time-varying and satisfies Assumption 2, then for all $t \geq 0$,

$$\begin{aligned} &f\bigg(\frac{\sum_{\tau=0}^{t}\alpha(\tau)\bar{z}(\tau)}{\sum_{\tau=0}^{t}\alpha(\tau)}\bigg) - f(z^*) \\ &\leq \frac{\|\bar{z}(0)-z^*\|^2 + G^2\sum_{\tau=0}^{t}\alpha^2(\tau)}{2\sum_{\tau=0}^{t}\alpha(\tau)} \\ &\quad + \frac{2G\alpha(0)\sum_{i=1}^{n}\|\bar{z}(0)-z_i(0)\|}{n\sum_{\tau=0}^{t}\alpha(\tau)} \\ &\quad + \frac{32G\sum_{i=1}^{n}\|x_i(0)\|}{\eta}\frac{\sum_{\tau=0}^{t-1}\alpha(\tau)\mu^{\tau}}{\sum_{\tau=0}^{t}\alpha(\tau)} \\ &\quad + \frac{32nG^2}{\eta\mu(1-\mu)}\frac{\sum_{\tau=0}^{t-1}\alpha(\tau)(\alpha(0)\mu^{\frac{\tau}{2}} + \alpha(\lceil\frac{\tau}{2}\rceil))}{\sum_{\tau=0}^{t}\alpha(\tau)}, \end{aligned}$$

$$\begin{aligned} &f\bigg(\frac{\sum_{\tau=0}^{t}\alpha(\tau)z_k(\tau)}{\sum_{\tau=0}^{t}\alpha(\tau)}\bigg) - f(z^*) \\ &\leq \frac{\|\bar{z}(0)-z^*\|^2 + G^2\sum_{\tau=0}^{t}\alpha^2(\tau)}{2\sum_{\tau=0}^{t}\alpha(\tau)} \\ &\quad + \frac{G\alpha(0)\sum_{i=1}^{n}(\|\bar{z}(0)-z_i(0)\| + \|z_k(0)-z_i(0)\|)}{n\sum_{\tau=0}^{t}\alpha(\tau)} \\ &\quad + \frac{32nG^2}{\eta\mu(1-\mu)}\frac{\sum_{\tau=0}^{t-1}\alpha(\tau)(\alpha(0)\mu^{\frac{\tau}{2}} + \alpha(\lceil\frac{\tau}{2}\rceil))}{\sum_{\tau=0}^{t}\alpha(\tau)} \\ &\quad + \frac{32G\sum_{i=1}^{n}\|x_i(0)\|}{\eta}\frac{\sum_{\tau=0}^{t-1}\alpha(\tau)\mu^{\tau}}{\sum_{\tau=0}^{t}\alpha(\tau)}, \ k \in \mathcal{V}. \end{aligned}$$

2) If the stepsize is fixed and $\alpha(t) = 1/\sqrt{T}$ for $T > 0$ steps, then

$$\begin{aligned} &f\bigg(\frac{\sum_{\tau=0}^{T-1}\bar{z}(\tau)}{T}\bigg) - f(z^*) \\ &\leq \frac{\|\bar{z}(0)-z^*\|^2 + G^2}{2\sqrt{T}} + \frac{2G\sum_{i=1}^{n}\|\bar{z}(0)-z_i(0)\|}{nT} \\ &\quad + \frac{32G\sum_{i=1}^{n}\|x_i(0)\|}{\eta(1-\mu)T} + \frac{32nG^2}{\eta\mu(1-\mu)\sqrt{T}}, \end{aligned}$$

$$\begin{aligned} &f\bigg(\frac{\sum_{\tau=0}^{T-1}z_k(\tau)}{T}\bigg) - f(z^*) \\ &\leq \frac{\|\bar{z}(0)-z^*\|^2 + G^2}{2\sqrt{T}} + \frac{32G\sum_{i=1}^{n}\|x_i(0)\|}{\eta(1-\mu)T} \\ &\quad + \frac{G\sum_{i=1}^{n}(\|\bar{z}(0)-z_i(0)\| + \|z_k(0)-z_i(0)\|)}{nT} \\ &\quad + \frac{32nG^2}{\eta\mu(1-\mu)\sqrt{T}}, \quad k \in \mathcal{V}. \end{aligned}$$

Here $\eta = \frac{1}{n^{nL}}$ and $\mu = (1 - \frac{1}{n^{nL}})^{1/L}$.

It is worth emphasizing that the uniform boundedness of all sugradients is a standard assumption in the literature of distributed subgradient [1], [11], [27]. Such a uniform boundedness assumption is not needed for distributed gradient descent [28].

[4] A directed graph is strongly connected if it has a directed path from any vertex to any other vertex. The union of two directed graphs, $\mathbb{G}_p$ and $\mathbb{G}_q$, with the same vertex set, written $\mathbb{G}_p \cup \mathbb{G}_q$, is meant the directed graph with the same vertex set and edge set being the union of the edge set of $\mathbb{G}_p$ and $\mathbb{G}_q$. Since this union is a commutative and associative binary operation, the definition extends unambiguously to any finite sequence of directed graphs with the same vertex set.

Theorem 2 is a generalization of Theorems 2 and 3 in [24], so its proof requires a more complicated treatment than those of Theorems 2 and 3 in [24]. It is not surprising that the bounds given in Theorems 2 and 3 in [24] are slightly better than those in Theorem 2 here as the former are tailored for a special case.

A. Analysis

We begin with a property of the $y_i(t)$ dynamics (7) which is independent of the $x_i(t)$ dynamics (6). Define a time-dependent $n \times n$ matrix $S(t)$ whose ijth entry is

$$s_{ij}(t) = \frac{w_{ij}(t)y_j(t)}{y_i(t+1)} = \frac{w_{ij}(t)y_j(t)}{\sum_{k=1}^n w_{ik}(t)y_k(t)}. \qquad (8)$$

The following lemma guarantees that each $s_{ij}(t)$, and thus $S(t)$, are well defined.

Lemma 1: If $\{\mathbb{G}(t)\}$ is uniformly strongly connected, then there exists a constant $\eta > 0$ such that $n \geq y_i(t) \geq \eta$ for all $i \in \mathcal{V}$ and $t \geq 0$.

The lemma is essentially the same as Corollary 2 (b) in [11], which further proves that if $\{\mathbb{G}(t)\}$ is uniformly strongly connected by sub-sequences of length L, $\eta \geq \frac{1}{n^{nL}}$.

It is easy to show that each $S(t)$ is a stochastic matrix[5]. An important property of $S(t)$ matrices is as follows. Let $y(t)$ be a vector in $\mathbb{R}^n$ whose ith entry is $y_i(t)$ for all $t \geq 0$.

Lemma 2: $y^\top(t) = y^\top(t+1)S(t)$ for all $t \geq 0$.

Proof of Lemma 2: From Assumption 1, $\sum_{i=1}^n w_{ij}(t) = 1$ for any $j \in \mathcal{V}$. Then, from (8),

$$[y^\top(t+1)S(t)]_j = \sum_{i=1}^n y_i(t+1)s_{ij}(t) = \sum_{i=1}^n y_i(t+1)\frac{w_{ij}(t)y_j(t)}{y_i(t+1)} = y_j(t),$$

in which $[\cdot]_j$ denotes the jth entry of a column vector. ■

The above property can be linked to the concept of "absolute probability sequence" of the sequence of stochastic matrices $\{S(t)\}$; see Proposition 2 in [24].

To proceed, define the following time-dependent quantity:

$$\langle z(t)\rangle \triangleq \frac{1}{n}\sum_{i=1}^n y_i(t)z_i(t) = \frac{1}{n}\sum_{i=1}^n x_i(t). \qquad (9)$$

Since $y_i(t) > 0$ by Lemma 1 and $\sum_{i=1}^n y_i(t) = n$, the above quantity is a time-varying convex combination of all $z_i(t)$. From update (6), for all $i \in \mathcal{V}$,

$$z_i(t+1) = \frac{x_i(t+1)}{y_i(t+1)} = \frac{\sum_{j=1}^n w_{ij}(t)x_j(t) - \alpha(t)g_i(t)}{y_i(t+1)}$$
$$= \sum_{j=1}^n \frac{w_{ij}(t)y_j(t)}{y_i(t+1)}z_j(t) - \frac{\alpha(t)g_i(t)}{y_i(t+1)}$$
$$= \sum_{j=1}^n s_{ij}(t)z_j(t) - \frac{\alpha(t)g_i(t)}{y_i(t+1)},$$

which, from Lemma 2, leads to

$$\langle z(t+1)\rangle = \sum_{i=1}^n \frac{y_i(t+1)}{n}z_i(t+1)$$
$$= \sum_{i=1}^n \frac{y_i(t+1)}{n}\sum_{j=1}^n s_{ij}(t)z_j(t) - \sum_{i=1}^n \frac{y_i(t+1)}{n}\frac{\alpha(t)g_i(t)}{y_i(t+1)}$$
$$= \sum_{j=1}^n \frac{y_j(t)}{n}z_j(t) - \sum_{i=1}^n \frac{\alpha(t)g_i(t)}{n}$$
$$= \langle z(t)\rangle - \frac{\alpha(t)}{n}\sum_{i=1}^n g_i(t). \qquad (10)$$

It is easy to show that the subgradient-push algorithm (2)–(3) and push-subgradient algorithm (4)–(5) share the same $\langle z(t)\rangle$ dynamics as given in (10). This common dynamics is the basis of the following unified analysis for heterogeneous distributed subgradient. It is also straightforward to get (10) from equation (9), update (6), and Assumption 1 as follows:

$$\langle z(t+1)\rangle = \frac{1}{n}\sum_{i=1}^n x_i(t+1)$$
$$= \frac{1}{n}\sum_{i=1}^n\sum_{j=1}^n w_{ij}(t)\Big[x_j(t) - \alpha(t)g_j(t)\sigma_j(t)\Big] - \frac{\alpha(t)}{n}\sum_{i=1}^n g_i(t)\big(1-\sigma_i(t)\big)$$
$$= \sum_{j=1}^n \frac{1}{n}\Big[x_j(t) - \alpha(t)g_j(t)\sigma_j(t)\Big] - \frac{\alpha(t)}{n}g_j(t)\big(1-\sigma_j(t)\big)$$
$$= \langle z(t)\rangle - \frac{\alpha(t)}{n}\sum_{i=1}^n g_i(t).$$

The above iterative dynamics of $\langle z\rangle$ can be treated (though not exactly the same) as a single-agent subgradient process for the convex cost function $\frac{1}{n}\sum_{i=1}^n f_i(z)$, which is a critical intermediate step.

The remaining analysis logic is as follows. Using the inequality $\|z_i(t) - z^*\|^2 \leq 2\|\langle z(t)\rangle - z^*\|^2 + 2\|\langle z(t)\rangle - z_i(t)\|^2$, the analysis is then to bound $\|\langle z(t)\rangle - z^*\|^2$ and $\|\langle z(t)\rangle - z_i(t)\|^2$ separately. For the term $\|\langle z(t)\rangle - z_i(t)\|^2$, since all z_i form a consensus process and $\langle z(t)\rangle$ is always a convex combination of all $z_i(t)$, the term can be bounded using consensus related techniques and relatively easy to deal with. Most analysis will focus on bounding the term $\|\langle z(t)\rangle - z^*\|^2$. It is worth noting that from (9), $\|\langle z(t)\rangle - z^*\|^2 = \|\frac{1}{n}\sum_{i=1}^n y_i(t)(z_i(t) - z^*)\|^2 = \|\frac{1}{n}\sum_{i=1}^n x_i(t) - z^*\|^2$, which is the actual Lyapunov function. Also note that update (10) is equivalent to $\bar{x}(t+1) = \bar{x}(t) - \frac{\alpha(t)}{n}\sum_{i=1}^n g_i(t)$ where $\bar{x}(t) = \frac{1}{n}\sum_{i=1}^n x_i(t)$, which is almost the same as the case of average consensus based subgradient [1] except that each subgradient g_i is taken at point z_i instead of x_i. But this $\bar{x}$ dynamics is elusive without Lemma 2.

To prove Theorem 2, we need the following lemmas.

Lemma 3: If $\{\mathbb{G}(t)\}$ is uniformly strongly connected, then for any fixed $\tau \geq 0$, $W(t)\cdots W(\tau+1)W(\tau)$ will converge

[5]A square nonnegative matrix is called a row stochastic matrix, or simply stochastic matrix, if its row sums all equal one.

to the set $\{v\mathbf{1}^\top : v \in \mathbb{R}^n, \mathbf{1}^\top v = 1, v > \mathbf{0}\}$ exponentially fast as $t \to \infty$.[6]

The lemma is essentially the same as Corollary 2 (a) in [11]. If $\{\mathbb{G}(t)\}$ is uniformly strongly connected by subsequences of length L, Lemma 3 implies that there exist constants $c > 0$ and $\mu \in [0, 1)$ and a sequence of stochastic vectors[7] $\{v(t)\}$ such that for all $i, j \in \mathcal{V}$ and $t \geq \tau \geq 0$,

$$\big|\big[W(t)\cdots W(\tau+1)W(\tau)\big]_{ij} - v_i(t)\big| \leq c\mu^{t-\tau}, \qquad (11)$$

where $[\cdot]_{ij}$ denotes the ijth entry of a matrix. It has been further shown in [11] that $c = 4$ and $\mu = (1 - \frac{1}{n^{nL}})^{1/L}$.

The following lemma is a generalization of Lemma 8 in [24], even though its proof follows the similar flow to that in the proof of Lemma 8 in [24].

Lemma 4: If $\{\mathbb{G}(t)\}$ is uniformly strongly connected by sub-sequences of length L and $\|g_i(t)\|$ is uniformly bounded above by a positive number G for all i and t, then for all $t \geq 0$ and $i \in \mathcal{V}$,

$$\Big\|z_i(t+1) - \frac{1}{n}\sum_{k=1}^{n} x_k(t)\Big\| \leq \frac{8}{\eta}\mu^t \sum_{k=1}^{n}\|x_k(0)\| + \frac{8nG}{\eta\mu}\sum_{s=0}^{t}\mu^{t-s}\alpha(s).$$

If, in addition, Assumption 2 holds, for all $t \geq 0$ and $i \in \mathcal{V}$,

$$\Big\|z_i(t+1) - \frac{1}{n}\sum_{k=1}^{n} x_k(t)\Big\| \leq \frac{8}{\eta}\mu^t \sum_{k=1}^{n}\|x_k(0)\| + \frac{8nG}{\eta\mu(1-\mu)}\big(\alpha(0)\mu^{t/2} + \alpha(\lceil t/2\rceil)\big).$$

Here $\eta > 0$ and $\mu \in (0, 1)$ are constants defined in Lemma 1 and (11), respectively.

The proofs of Lemma 4 and Theorem 2 are omitted due to space limitations and can be found in [29, Lemma 14 and Theorem 7].

B. A Special Case

In this subsection, we discuss a special case in which $W(t)$ is a doubly stochastic matrix[8] at all time $t \geq 0$. In this case, it is easy to see from (7) that $y_i(t) = 1$ for all $i \in \mathcal{V}$ and $t \geq 0$, and thus $z_i(t) = x_i(t)$ for all $i \in \mathcal{V}$ and $t \geq 0$. This observation holds for all push-sum based distributed optimization algorithms studied in this paper as they share the same $y_i(t)$ dynamics which is independent of their $x_i(t)$ dynamics. Then, the subgradient-push, push-subgradient, and heterogeneous subgradient algorithms all simplify to average consensus based subgradient algorithms. Specifically, subgradient-push (2)–(3) simplifies to

$$x_i(t+1) = \sum_{j\in\mathcal{N}_i(t)} w_{ij}(t)\Big[x_j(t) - \alpha(t)g_j(x_j(t))\Big], \qquad (12)$$

and push-subgradient (4)–(5) simplifies to

$$x_i(t+1) = \sum_{j\in\mathcal{N}_i(t)} w_{ij}(t)x_j(t) - \alpha(t)g_i(x_i(t)), \qquad (13)$$

which is the "standard" average consensus based distributed subgradient proposed in [1]. The two updates (12) and (13) are analogous to the so-called "adapt-then-combine" and "combine-then-adapt" diffusion strategies in distributed optimization and learning [30]. Thus, in the special case under consideration, the heterogeneous distributed subgradient algorithm (6)–(7) simplifies to

$$x_i(t+1) = \sum_{j\in\mathcal{N}_i(t)} w_{ij}(t)\Big[x_j(t) - \alpha(t)g_j(x_j(t))\sigma_j(t)\Big] - \alpha(t)g_i(x_i(t))\big(1-\sigma_i(t)\big),$$

which is an average consensus based heterogeneous distributed subgradient algorithm allowing each agent to arbitrarily switch between updates (12) and (13). The preceding discussion implies that the results in this paper apply to the corresponding average consensus based algorithms.

IV. Privacy Protection

In a multi-agent distributed computation process, agents have to collaborate via information transmission, while communication between neighboring agents may lead to information leakage in the presence of adversarial agents. There exist three major methods for privacy preserving: differential privacy [31], [32], partially homomorphic encryption [33], [34], and state decomposition [35], [36]. Each approach has its advantages and disadvantages. The differential privacy based approach faces a tradeoff between accuracy level and privacy guarantee probability, while encryption and decryption operations in the partially homomorphic encryption approach require expensive computation costs. The state decomposition approach overcomes these limitations but requires additional storage space.

In a distributed optimization process, each agent's cost function f_i is typically treated as its private information. To preserve the privacy of f_i, it is necessary to protect agent i's (sub)gradient g_i from leaking. To see this, a subgradient of a convex function at a point provides a valid gradient at that point or a range of gradients in the case of nonsmooth points. Thus, by collecting (sub)gradient information at various points, one can infer properties about the convex function, such as its behavior and shape. This information can then be used to approximate or reconstruct the function itself. For example, the seminal work [37] shows that transmitted gradients leak private training data in federated learning. Another example is in a distributed optimization based multi-robot rendezvous process, exchanging gradients with neighbors discloses an agent's position [31]. With these in

[6]We use $\mathbf{0}$ and $\mathbf{1}$ to denote the vectors whose entries all equal to 0 or 1, respectively, where the dimensions of the vectors are to be understood from the context. We use $v > \mathbf{0}$ to denote a positive vector, i.e., each entry of v is positive.

[7]A nonnegative vector is called a stochastic vector if its entries sum to 1.

[8]A square nonnegative matrix is called a doubly stochastic matrix if its row sums and column sums all equal one.

mind, we treat subgradients as agents' private information in a distributed subgradient algorithm. In the sequel, we will consider two common types of adversarial agents, namely honest-but-curious adversaries and eavesdroppers, and show that the proposed heterogeneous algorithm (6)–(7) is capable of preventing the leakage of subgradients, whereas the subgradient-push and push-subgradient algorithms are vulnerable in the presence of these adversarial agents.

We begin with the definitions. An *honest-but-curious adversary* is an agent within the network which knows the network topology, follows the given algorithm, and attempts to infer the private information of other agents [38][9]. An *eavesdropper* is an external adversary who knows the network topology and is able to eavesdrop on (a portion of) transmitted data [39]. Since both adversaries know the network topology, we assume they both know all weights $w_{ij}(t)$, $i \in \mathcal{V}$, $j \in \mathcal{N}_i(t)$, $t \geq 0$ because a typical choice of $w_{ij}(t)$ is $1/|\mathcal{N}_j^-(t)|$ for all $j \in \mathcal{N}_i(t)$, which are uniquely determined by the network topology. It is also assumed that both adversaries know all stepsizes $\alpha(t)$, $t \geq 0$, which are shared among all agents.

Let $\mathcal{H}$ be the set of all honest-but-curious adversaries and $\mathcal{I}_{\mathcal{H}}(t)$ be the information set accessible to the adversaries in $\mathcal{H}$ at time t. Let $\mathcal{E}_{\mathcal{D}}$ be the set of directed edges that an eavesdropper $\mathcal{D}$ can eavesdrop and $\mathcal{I}_{\mathcal{D}}(t)$ be the information set accessible to the eavesdropper $\mathcal{D}$ at time t. We next specify $\mathcal{I}_{\mathcal{H}}(t)$ and $\mathcal{I}_{\mathcal{D}}(t)$ for the subgradient-push, push-subgradient, and heterogeneous subgradient algorithms. From on the update of subgradient-push (2)-(3), $\mathcal{I}_{\mathcal{H}}(t) = \{x_i(t), y_i(t), g_i(t), w_{ij}(t)[x_j(t) - \alpha(t)g_j(t)], w_{ij}(t)y_j(t), i \in \mathcal{H}, j \in \mathcal{N}_i(t)\}$ and $\mathcal{I}_{\mathcal{D}}(t) = \{w_{ij}(t)[x_j(t) - \alpha(t)g_j(t)], w_{ij}(t)y_j(t), (j,i) \in \mathcal{E}_{\mathcal{D}}\}$. Since both adversaries know all $w_{ij}(t)$ weights, it follows that

$$\begin{aligned}\mathcal{I}_{\mathcal{H}}(t) &= \big\{x_i(t), y_i(t), g_i(t), x_j(t) - \alpha(t)g_j(t), y_j(t),\\ &\qquad i \in \mathcal{H}, j \in \mathcal{N}_i(t)\big\},\\ \mathcal{I}_{\mathcal{D}}(t) &= \big\{x_j(t) - \alpha(t)g_j(t), y_j(t), (j,i) \in \mathcal{E}_{\mathcal{D}}\big\}.\end{aligned}$$

Similarly, from the update of push-subgradient (4)-(5),

$$\begin{aligned}\mathcal{I}_{\mathcal{H}}(t) &= \big\{x_i(t), y_i(t), g_i(t), x_j(t), y_j(t), i \in \mathcal{H}, j \in \mathcal{N}_i(t)\big\},\\ \mathcal{I}_{\mathcal{D}}(t) &= \big\{x_j(t), y_j(t), (j,i) \in \mathcal{E}_{\mathcal{D}}\big\}.\end{aligned}$$

Also, from the heterogeneous subgradient algorithm (6)-(7),

$$\begin{aligned}\mathcal{I}_{\mathcal{H}}(t) &= \big\{x_i(t), y_i(t), g_i(t), x_j(t) - \alpha(t)g_j(t)\sigma_j(t), y_j(t),\\ &\qquad i \in \mathcal{H}, j \in \mathcal{N}_i(t)\big\},\\ \mathcal{I}_{\mathcal{D}}(t) &= \big\{x_j(t) - \alpha(t)g_j(t)\sigma_j(t), y_j(t), (j,i) \in \mathcal{E}_{\mathcal{D}}\big\}.\end{aligned}$$

The following results show that a set of honest-but-curious adversaries or an eavesdropper can infer a normal agent's subgradient information under appropriate assumptions in both subgradient-push and push-subgradient algorithms.

It is worth noting that any piece of subgradient information is always associated with a point in $\mathbb{R}^d$. To emphasize this, we will from now on write $g_i(t)$ as $g_i(z_i(t))$. Inferring an agent i's subgradient information involves determining both a point $z_i(t)$ and its corresponding subgradient $g_i(z_i(t))$. For the purpose of simple notation, we define $\mathcal{M}_i(t) = \mathcal{N}_i \setminus \{i\}$ and $\mathcal{M}_i^-(t) = \mathcal{N}_i^-(t) \setminus \{i\}$ for all $i \in \mathcal{V}$.

Lemma 5: For the subgradient-push algorithm (2)-(3) and at any time $t > 0$, a set of honest-but-curious adversaries $\mathcal{H}$ can infer $z_i(t)$ and $g_i(z_i(t))$ if $\mathcal{M}_i(t-1) \subset \mathcal{H}$, $\mathcal{M}_i^-(t-1) \cap \mathcal{H} \neq \emptyset$, and $\mathcal{M}_i^-(t) \cap \mathcal{H} \neq \emptyset$, and an eavesdropper $\mathcal{D}$ can infer $z_i(t)$ and $g_i(z_i(t))$ if $\{(j,i) : j \in \mathcal{M}_i(t-1)\} \subset \mathcal{E}_{\mathcal{D}}$, $\{(i,k) : k \in \mathcal{M}_i^-(t-1)\} \cap \mathcal{E}_{\mathcal{D}} \neq \emptyset$, and $\{(i,k) : k \in \mathcal{M}_i^-(t)\} \cap \mathcal{E}_{\mathcal{D}} \neq \emptyset$.

Proof of Lemma 5: From (2) and $\mathcal{N}_i(t) = \mathcal{M}_i(t) \cup \{i\}$, $x_i(t)$ can be decomposed as

$$\begin{aligned}x_i(t) &= w_{ii}(t-1)\Big[x_i(t-1) - \alpha(t-1)g_i(z_i(t-1))\Big]\\ &+ \sum_{j \in \mathcal{M}_i(t-1)} w_{ij}(t-1)\Big[x_j(t-1) - \alpha(t-1)g_j(z_j(t-1))\Big]\end{aligned}$$

Since $\mathcal{M}_i(t-1) \subset \mathcal{H}$, $\mathcal{H}$ knows the summation term in the above expression. At time $t-1$, agent i transmits $w_{ki}(t-1)[x_i(t-1) - \alpha(t-1)g_i(z_i(t-1))]$ to each out-neighbor $k \in \mathcal{M}_i^-(t-1)$. Since $\mathcal{M}_i^-(t-1) \cap \mathcal{H} \neq \emptyset$ and $\mathcal{H}$ knows all $w_{ki}(t-1)$ weights, $\mathcal{H}$ can infer the value of $x_i(t-1) - \alpha(t-1)g_i(z_i(t-1))$. Consequently, $\mathcal{H}$ can infer $x_i(t)$. Similarly, from (3), $\mathcal{H}$ can infer $y_i(t)$ and therefore $z_i(t) = x_i(t)/y_i(t)$. At time t, as $\mathcal{M}_i^-(t) \cap \mathcal{H} \neq \emptyset$, using the preceding argument, $\mathcal{H}$ can infer the value of $x_i(t) - \alpha(t)g_i(z_i(t))$. With this value, $\mathcal{H}$ can infer $g_i(z_i(t))$ using inferred $x_i(t)$ and known $\alpha(t)$.

It is easy to see that all the information used by $\mathcal{H}$ for inferring $z_i(t)$ and $g_i(z_i(t))$ is accessible to the eavesdropper $\mathcal{D}$ under the given conditions $\{(j,i) : j \in \mathcal{M}_i(t-1)\} \subset \mathcal{E}_{\mathcal{D}}$, $\{(i,k) : k \in \mathcal{M}_i^-(t-1)\} \cap \mathcal{E}_{\mathcal{D}} \neq \emptyset$, and $\{(i,k) : k \in \mathcal{M}_i^-(t)\} \cap \mathcal{E}_{\mathcal{D}} \neq \emptyset$. Therefore, $\mathcal{D}$ is also able to infer $z_i(t)$ and $g_i(z_i(t))$. ■

Lemma 6: For the push-subgradient algorithm (4)-(5) and at any time $t+1$ with $t \geq 0$, a set of honest-but-curious adversaries $\mathcal{H}$ can infer $z_i(t)$ and $g_i(z_i(t))$ if $\mathcal{M}_i(t) \subset \mathcal{H}$, $\mathcal{M}_i^-(t) \cap \mathcal{H} \neq \emptyset$, and $\mathcal{M}_i^-(t+1) \cap \mathcal{H} \neq \emptyset$, and an eavesdropper $\mathcal{D}$ can infer $z_i(t)$ and $g_i(z_i(t))$ if $\{(j,i) : j \in \mathcal{M}_i(t)\} \subset \mathcal{E}_{\mathcal{D}}$, $\{(i,k) : k \in \mathcal{M}_i^-(t)\} \cap \mathcal{E}_{\mathcal{D}} \neq \emptyset$, and $\{(i,k) : k \in \mathcal{M}_i^-(t+1)\} \cap \mathcal{E}_{\mathcal{D}} \neq \emptyset$.

Proof of Lemma 6: It is clear that under the given conditions, $\mathcal{H}$ has access to more information than $\mathcal{D}$. Thus, it is sufficient to show that $\mathcal{D}$ can infer $z_i(t)$ and $g_i(z_i(t))$. At time t, agent i transmits $w_{ki}(t)x_i(t)$ to each out-neighbor $k \in \mathcal{M}_i^-(t)$. Since $\{(i,k) : k \in \mathcal{M}_i^-(t)\} \cap \mathcal{E}_{\mathcal{D}} \neq \emptyset$ and $\mathcal{D}$ knows all $w_{ki}(t)$ weights, $\mathcal{D}$ can infer $x_i(t)$. Similarly, $\mathcal{D}$ can infer $y_i(t)$ and therefore $z_i(t) = x_i(t)/y_i(t)$. At time $t+1$, as $\{(i,k) : k \in \mathcal{M}_i^-(t+1)\} \cap \mathcal{E}_{\mathcal{D}} \neq \emptyset$, using the preceding argument, $\mathcal{D}$ can infer $x_i(t+1)$. From (4) and $\mathcal{N}_i(t) = \mathcal{M}_i(t) \cup \{i\}$, $x_i(t+1)$ can be decomposed as $x_i(t+1) = w_{ii}(t)x_i(t) + \sum_{j \in \mathcal{M}_i(t)} w_{ij}(t)x_j(t) - \alpha(t)g_i(z_i(t))$. Since $\{(j,i) : j \in \mathcal{M}_i(t)\} \subset \mathcal{E}_{\mathcal{D}}$, $\mathcal{D}$ knows the summation term in the decomposition, and therefore can infer $g_i(z_i(t))$ using inferred $x_i(t)$ and known $\alpha(t)$. ■

[9] An honest-but-curious adversary is called semi-honest in [38].

In contrast to subgradient-push and push-subgradient, the proposed heterogeneous subgradient algorithm (6)-(7) can prevent subgradients from leaking because the inferring approaches used in the proofs of Lemma 5 and Lemma 6 cannot be applied to the heterogeneous subgradient algorithm. This is due to the independent and private switching signal sequence $\{\sigma_i(t)\}$ at any normal agent i. It can be seen from (6) that $g_i(t)$ always appears multiplied by a private binary value $\sigma_i(t)$ or $1-\sigma_i(t)$, which makes it impossible for any set of honest-but-curious adversaries and eavesdropper to deterministically infer a normal agent i's subgradient information. We summarize as follows:

Lemma 7: Without knowledge of $\{\sigma_i(t)\}$, neither a set of honest-but-curious adversaries nor an eavesdropper can deterministically infer the subgradient information of any normal agent i through the heterogeneous subgradient algorithm (6)-(7).

While deterministic inference is impossible, adversaries may employ probabilistic inference strategies, which is a direction for future research.

ACKNOWLEDGEMENT

The authors wish to thank Yongqiang Wang (Clemson University) for useful discussion.

REFERENCES

[1] A. Nedić and A. Ozdaglar. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1):48–61, 2009.

[2] 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*, 47:278–305, 2019.

[3] A. Nedić and J. Liu. Distributed optimization for control. *Annual Review of Control, Robotics, and Autonomous Systems*, 1:77–103, 2018.

[4] D.K. Molzahn, F. Dörfler, H. Sandberg, S.H. Low, S. Chakrabarti, R. Baldick, and J. Lavaei. A survey of distributed optimization and control algorithms for electric power systems. *IEEE Transactions on Smart Grid*, 8(6):2941–2962, 2017.

[5] E. Wei and A. Ozdaglar. Distributed alternating direction method of multipliers. In *Proceedings of the 51st IEEE Conference on Decision and Control*, pages 5445–5450, 2012.

[6] G. Qu and N. Li. Accelerated distributed Nesterov gradient descent. *IEEE Transactions on Automatic Control*, 65(6):2566–2581, 2019.

[7] Z. Li, W. Shi, and M. Yan. A decentralized proximal-gradient method with network independent step-sizes and separated convergence rates. *IEEE Transactions on Signal Processing*, 67(17):4494–4506, 2019.

[8] B. Gharesifard and J. Cortés. Distributed continuous-time convex optimization on weight-balanced digraphs. *IEEE Transactions on Automatic Control*, 59(3):781–786, 2013.

[9] L. Xiao, S. Boyd, and S. Lall. A scheme for robust distributed sensor fusion based on average consensus. In *Proceedings of the 4th International Conference on Information Processing in Sensor Networks*, pages 63–70, 2005.

[10] B. Gharesifard and J. Cortés. Distributed strategies for generating weight-balanced and doubly stochastic digraphs. *European Journal of Control*, 18(6):539–557, 2012.

[11] A. Nedić and A. Olshevsky. Distributed optimization over time-varying directed graphs. *IEEE Transactions on Automatic Control*, 60(3):601–615, 2015.

[12] C. Xi and U.A. Khan. DEXTRA: A fast algorithm for optimization over directed graphs. *IEEE Transactions on Automatic Control*, 62(10):4980–4993, 2017.

[13] W. Shi, Q. Ling, G. Wu, and W. Yin. EXTRA: An exact first-order algorithm for decentralized consensus optimization. *SIAM Journal on Optimization*, 25(2):944–966, 2015.

[14] A. Nedić, A. Olshevsky, and W. Shi. Achieving geometric convergence for distributed optimization over time-varying graphs. *SIAM Journal on Optimization*, 27(4):2597–2633, 2017.

[15] S. Pu, W. Shi, J. Xu, and A. Nedić. Push-pull gradient methods for distributed optimization in networks. *IEEE Transactions on Automatic Control*, 66(1):1–16, 2021.

[16] R. Xin and U.A. Khan. A linear algorithm for optimization over directed graphs with geometric convergence. *IEEE Control Systems Letters*, 2(3):315–320, 2018.

[17] D.T.A. Nguyen, D.T. Nguyen, and A. Nedić. Accelerated AB/Push-Pull methods for distributed optimization over time-varying directed networks. *IEEE Transactions on Control of Network Systems*, 2023. accepted.

[18] A. Nedić, A. Olshevsky, W. Shi, and C.A. Uribe. Geometrically convergent distributed optimization with uncoordinated step-sizes. In *Proceedings of the 2017 American Control Conference*, pages 3950–3955, 2017.

[19] Y. Wang and A. Nedić. Decentralized gradient methods with time-varying uncoordinated stepsizes: Convergence analysis and privacy design. *IEEE Transactions on Automatic Control*, 2023. accepted.

[20] C. Sun, M. Ye, and G. Hu. Distributed optimization for two types of heterogeneous multiagent systems. *IEEE Transactions on Neural Networks and Learning Systems*, 32(3):1314–1324, 2021.

[21] T. Vogels, L. He, A. Koloskova, S.P. Karimireddy, T. Lin, S.U. Stich, and M. Jaggi. RelaySum for decentralized deep learning on heterogeneous data. In *Advances in Neural Information Processing Systems*, volume 34, pages 28004–28015, 2021.

[22] B. Polyak. A general method for solving extremum problems. *Doklady Akademii Nauk*, 8(3):593–597, 1967.

[23] D. Kempe, A. Dobra, and J. Gehrke. Gossip-based computation of aggregate information. In *Proceedings of the 44th IEEE Symposium on Foundations of Computer Science*, pages 482–491, 2003.

[24] Y. Lin and J. Liu. Subgradient-push is of the optimal convergence rate. In *Proceedings of the 61st IEEE Conference on Decision and Control*, pages 5849–5856, 2022.

[25] A. Nedić, A. Olshevsky, A. Ozdaglar, and J.N. Tsitsiklis. On distributed averaging algorithms and quantization effects. *IEEE Transactions on Automatic Control*, 54(11):2506–2517, 2009.

[26] M. Cao, A.S. Morse, and B.D.O. Anderson. Reaching a consensus in a dynamically changing environment: A graphical approach. *SIAM Journal on Control and Optimization*, 47(2):575–600, 2008.

[27] A. Nedić, A. Olshevsky, and M.G. Rabbat. Network topology and communication-computation tradeoffs in decentralized optimization. *Proceedings of the IEEE*, 106(5):953–976, 2018.

[28] A. Nedić and A. Olshevsky. Stochastic gradient-push for strongly convex functions on time-varying directed graphs. *IEEE Transactions on Automatic Control*, 61(12):3936–3947, 2016.

[29] Y. Lin and J. Liu. An analysis tool for push-sum based distributed optimization. *arXiv preprint*, 2023. arXiv:2304.09443 [math.OC].

[30] A.H. Sayed, S.-Y. Tu, J. Chen, X. Zhao, and Z.J. Towfic. Diffusion strategies for adaptation and learning over networks. *IEEE Signal Processing Magazine*, 30(3):155–171, 2013.

[31] Z. Huang, S. Mitra, and N. Vaidya. Differentially private distributed optimization. In *Proceedings of the 16th International Conference on Distributed Computing and Networking*, pages 1–10, 2015.

[32] S. Han, U. Topcu, and G.J. Pappas. Differentially private distributed constrained optimization. *IEEE Transactions on Automatic Control*, 62(1):50–64, 2017.

[33] N.M. Freris and P. Patrinos. Distributed computing over encrypted data. In *Proceedings of the 54th Annual Allerton Conference on Communication, Control, and Computing*, pages 1116–1122, 2016.

[34] Y. Lu and M. Zhu. Privacy preserving distributed optimization using homomorphic encryption. *Automatica*, 96:314–325, 2018.

[35] Y. Wang. Privacy-preserving average consensus via state decomposition. *IEEE Transactions on Automatic Control*, 64(11):4711–4716, 2019.

[36] X. Chen, L. Huang, K. Ding, S. Dey, and L. Shi. Privacy-preserving push-sum average consensus via state decomposition. *IEEE Transactions on Automatic Control*, 68(12):7974–7981, 2023.

[37] L. Zhu, Z. Liu, and S. Han. Deep leakage from gradients. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[38] O. Goldreich. Secure Multi-Party Computation. *Manuscript. Preliminary version*, 78(110), 1998.

[39] R.L. Rivest and A. Shamir. How to expose an eavesdropper. *Communications of the ACM*, 27(4):393–394, 1984.