A Decentralized Proximal-Gradient Method With Network Independent Step-Sizes and Separated Convergence Rates

Zhi Li, Wei Shi, and Ming Yan

***Abstract*—This paper proposes a novel proximal-gradient algorithm for a decentralized optimization problem with a composite objective containing smooth and nonsmooth terms. Specifically, the smooth and nonsmooth terms are dealt with by gradient and proximal updates, respectively. The proposed algorithm is closely related to a previous algorithm, PG-EXTRA (W. Shi, Q. Ling, G. Wu, and W. Yin, "A proximal gradient algorithm for decentralized composite optimization," *IEEE Trans. Signal Process.*, vol. 63, no. 22, pp. 6013–6023, 2015), but has a few advantages. First of all, agents use uncoordinated step-sizes, and the stable upper bounds on step-sizes are independent of network topologies. The step-sizes depend on local objective functions, and they can be as large as those of the gradient descent. Second, for the special case without nonsmooth terms, linear convergence can be achieved under the strong convexity assumption. The dependence of the convergence rate on the objective functions and the network are separated, and the convergence rate of the new algorithm is as good as one of the two convergence rates that match the typical rates for the general gradient descent and the consensus averaging. We provide numerical experiments to demonstrate the efficacy of the introduced algorithm and validate our theoretical discoveries.**

***Index Terms*—Decentralized optimization, proximal-gradient, convergence rates, network independent.**

I. Introduction

THIS paper focuses on the following decentralized optimization problem:

$$\underset{x\in\mathbb{R}^p}{\text{minimize}}\ \bar{f}(x) := \frac{1}{n}\sum_{i=1}^{n}(s_i(x)+r_i(x)), \tag{1}$$

where $s_i : \mathbb{R}^p \to \mathbb{R}$ and $r_i : \mathbb{R}^p \to \mathbb{R}\cup\{+\infty\}$ are two lower semi-continuous proper convex functions held privately by agent i to encode the agent's objective. We assume that one function (e.g., without loss of generality, s_i) is differentiable and has a Lipschitz continuous gradient with parameter $L > 0$, and the other function r_i is proximable, i.e., its proximal mapping

$$\mathbf{prox}_{\lambda r_i}(y) = \underset{x\in\mathbb{R}^p}{\arg\min}\ \lambda r_i(x) + \frac{1}{2}\|x-y\|^2,$$

has a closed-form solution or can be computed easily. Examples of s_i include linear functions, quadratic functions, and logistic functions, while r_i could be the ℓ_1 norm, 1D total variation, or indicator functions of simple convex sets. In addition, we assume that the agents are connected through a fixed bi-directional communication network. Every agent in the network wants to obtain an optimal solution of (1) while it can only receive/send nonsensitive messages[1] from/to its immediate neighbors.

Specific problems of form (1) that require a decentralized computing architecture have appeared in various areas including networked multi-vehicle coordination, distributed information processing and decision making in sensor networks, as well as distributed estimation and learning. Some examples include distributed average consensus [2]–[4], distributed spectrum sensing [5], information control [6], [7], power systems control [8], [9], statistical inference and learning [10]–[12]. In general, decentralized optimization fits the scenarios where the data is collected and/or stored in a distributed network, a fusion center is either inapplicable or unaffordable, and/or computing is required to be performed in a distributed but collaborative manner by multiple agents or by network designers.

A. Literature Review

The study on distributed algorithms dates back to the early 1980s [13], [14]. Since then, due to the emergence of large-scale networks, decentralized (optimization) algorithms, as a special type of distributed algorithms for solving problem (1), have received attention. Many efforts have been made on star networks with one master agent and multiple slave agents [15], [16]. This scheme is "centralized" due to the use of a "master" agent. It may suffer a single point of failure and may violate the privacy requirement in certain applications. In this paper, we focus on solving (1) in a decentralized fashion, where no "master" agent is used.

Manuscript received September 24, 2018; revised March 5, 2019 and June 18, 2019; accepted June 19, 2019. Date of publication July 1, 2019; date of current version August 8, 2019. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Mats Bengtsson. This work was supported in part by the National Science Foundation under Grant DMS-1621798. *(Corresponding author: Ming Yan.)*

Z. Li is with the Department of Computational Mathematics, Science and Engineering, Michigan State University, East Lansing, MI 48824 USA (e-mail: zhili@msu.edu).

W. Shi is with the Department of Electrical Engineering, Princeton University, Princeton, NJ 08544 USA (e-mail: wshi36@asu.edu).

M. Yan is with the Department of Computational Mathematics, Science and Engineering Department of Mathematics, Michigan State University, East Lansing, MI 48824 USA (e-mail: myan@msu.edu).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the author.

Digital Object Identifier 10.1109/TSP.2019.2926022

[1] We believe that agent i's instantaneous estimation on the optimal solution is not a piece of sensitive information but s_i and r_i are.

Incremental algorithms [17]–[22] can solve (1) without the need of a "master" agent and it is based on a directed ring network. To handle general (possibly time-varying) networks, the distributed sub-gradient algorithm was proposed in [23]. This algorithm and its variants [24], [25] are intuitive and simple but usually slow due to the diminishing step-size that is needed to obtain a consensual and optimal solution, even if the objective functions are differentiable and strongly convex. With a fixed step-size, these distributed methods can be fast, but they only converge to a neighborhood of the solution set which depends on the step-size. This phenomenon creates an exactness-speed dilemma [26].

A class of distributed approaches that bypass this dilemma is based on introducing the Lagrangian dual. The resulting algorithms include distributed dual decomposition [27] and decentralized alternating direction method of multipliers (ADMM) [28]. The decentralized ADMM and its proximal-gradient variant can employ a fixed step-size to achieve $O(1/k)$ rate under general convexity assumptions [29]–[31]. Under the strong convexity assumption, the decentralized ADMM has linear convergence for time-invariant undirected graphs [32]. There exist some other distributed methods that do not (explicitly) use dual variables but can still converge to an optimal consensual solution with fixed step-sizes. In particular, works in [33], [34] employ multi-consensus inner loops, Nesterov's acceleration, and/or the adapt-then-combine (ATC) strategy. Under the assumption that the objectives have bounded and Lipschitz continuous gradients,[2] the algorithm proposed in [34] has $O\left(\ln(k)/k^2\right)$ rate. References [1], [36] use a difference structure to cancel the steady state error in decentralized gradient descent [23], [26], thereby developing the algorithm EXTRA and its proximal-gradient variant PG-EXTRA. It converges at an $O(1/k)$ rate when the objective function in (1) is convex and has a linear convergence rate when the objective function is strongly convex and $r_i(x) = 0$ for all i.

A number of recent works employed the so-called gradient tracking [37] to conquer different issues [38]–[42]. To be specific, works [38], [42] relax the step-size rule to allow uncoordinated step-sizes across agents. Paper [39] solves non-convex optimization problems. Paper [41] aims at achieving geometric convergence over time-varying graphs. Work [40] improves the convergence rate over EXTRA, and its formulation is the same as that in [42].

Another topic of interest is decentralized optimization over directed graphs [41], [43]–[46], which is beyond the scope of this paper.

B. Proposed Algorithm and Its Advantages

To proceed, let us introduce some basic notation first. Agent i holds a local variable $x_i \in \mathbb{R}^p$, and we denote x_i^k as its value at the k-th iteration. Then, we introduce a new function that is the average of all the local functions with local variables as

$$f(\mathbf{x}) := \frac{1}{n}\sum_{i=1}^{n}(s_i(x_i) + r_i(x_i)), \tag{2}$$

where

$$\mathbf{x} := \begin{pmatrix} - & x_1^\top & - \\ - & x_2^\top & - \\ & \vdots & \\ - & x_n^\top & - \end{pmatrix} \in \mathbb{R}^{n\times p}. \tag{3}$$

If all local variables are identical, i.e., $x_1 = \cdots = x_n$, we say that $\mathbf{x}$ is consensual. In addition, we define

$$s(\mathbf{x}) := \frac{1}{n}\sum_{i=1}^{n} s_i(x_i), \quad r(\mathbf{x}) := \frac{1}{n}\sum_{i=1}^{n} r_i(x_i). \tag{4}$$

We have $f(\mathbf{x}) = s(\mathbf{x}) + r(\mathbf{x})$. The gradient of s at $\mathbf{x}$ is given in the same way as $\mathbf{x}$ in (3) by

$$\nabla s(\mathbf{x}) := \begin{pmatrix} - & (\nabla s_1(x_1))^\top & - \\ - & (\nabla s_2(x_2))^\top & - \\ & \vdots & \\ - & (\nabla s_n(x_n))^\top & - \end{pmatrix} \in \mathbb{R}^{n\times p}. \tag{5}$$

By making a simple modification over PG-EXTRA [1], our proposed algorithm brings a big improvement in the speed and the dependency of convergence over networks. To better expose this simple modification, let us compare a special case of our proposed algorithm with EXTRA for the smooth case, i.e., $r(\mathbf{x}) = 0$.

$$\begin{aligned}(\text{EXTRA}^3)\ \mathbf{x}^{k+2} = {} & \frac{\mathbf{I}+\mathbf{W}}{2}(2\mathbf{x}^{k+1} - \mathbf{x}^k) \\ & - \alpha\nabla s(\mathbf{x}^{k+1}) + \alpha\nabla s(\mathbf{x}^k),\end{aligned} \tag{6a}$$

$$\begin{aligned}(\text{Proposed NIDS})\ \mathbf{x}^{k+2} = {} & \frac{\mathbf{I}+\mathbf{W}}{2}(2\mathbf{x}^{k+1} - \mathbf{x}^k \\ & - \alpha\nabla s(\mathbf{x}^{k+1}) + \alpha\nabla s(\mathbf{x}^k)).\end{aligned} \tag{6b}$$

Here, $\mathbf{W} \in \mathbb{R}^{n\times n}$ is a matrix that represents information exchange between neighboring agents (more details about this matrix are in Assumption 1) and α is the step-size. The only difference between EXTRA and the proposed algorithm is the information exchanged between the agents. EXTRA exchanges only the estimations $2\mathbf{x}^{k+1} - \mathbf{x}^k$, while the proposed algorithm exchanges the gradient adapted estimations, i.e., $2\mathbf{x}^{k+1} - \mathbf{x}^k - \alpha\nabla s(\mathbf{x}^{k+1}) + \alpha\nabla s(\mathbf{x}^k)$. Because of this small modification, the proposed algorithm has a Network InDependent Step-size, which will be explained later. Therefore we name the proposed algorithm NIDS and will use this abbreviation throughout the paper. For the nonsmooth case, more detailed comparison between PG-EXTRA and NIDS will be given in Section III.

[2]This means that the nonsmooth terms r_i's are absent. Such assumption is much stronger than the one used for achieving the $O(1/k^2)$ rate in Nesterov's optimal gradient method [35].

[3]In the original EXTRA, two mixing matrices $\mathbf{W}$ and $\widetilde{\mathbf{W}}$ are used. For simplicity, we let $\widetilde{\mathbf{W}} = \frac{I+\mathbf{W}}{2}$ here.

TABLE I
SUMMARY OF ALGORITHMIC PARAMETERS USED IN EXTRA AND NIDS. THE STEP SIZE BOUND ON α FOR EXTRA COMES FROM REFERENCE [47] WHICH IMPROVES THAT GIVEN IN REFERENCE [36]

Method	α or α_i	c
EXTRA ($\lambda_n(\mathbf{W})$ given)	$\alpha < (5+3\lambda_n(\mathbf{W}))/(4\max_i L_i)$	–
EXTRA ($\lambda_n(\mathbf{W})$ not given)	$\alpha < 1/(2\max_i L_i)$	–
NIDS ($\lambda_n(\mathbf{W})$ given)	$\alpha_i < 2/L_i$	$c \leq 1/((1-\lambda_n(\mathbf{W}))\max_i \alpha_i)$
NIDS ($\lambda_n(\mathbf{W})$ not given)	$\alpha_i < 2/L_i$	$c \leq 1/(2\max_i \alpha_i)$

A large and network independent step-size for NIDS: All works mentioned above either employ pessimistic step-sizes or have network dependent upper bounds on step-sizes. Furthermore, the step-sizes for the strongly convex case are more conservative. For example, the step-size used to achieve linear convergence rates for EXTRA in [36], [48] is in the order of $O(\mu/L^2)$, where μ and L are the strong convexity constant of $s(\mathbf{x})$ and Lipschitz constant of $\nabla s(\mathbf{x})$, respectively. As a contrast, the centralized gradient descent can choose a step-size in the order of $O(1/L)$. The upper bound of step-size for EXTRA was recently improved to $(5+3\lambda_n(\mathbf{W}))/(4L)$ in [47]. We can choose $1/(2L)$ for any $\mathbf{W}$ satisfying Assumption 1. Another example of employing a constant step-size in distributed optimization is DIGing [41]. Although its ATC variant in [42] was shown to converge faster than DIGing, the step-size is still very conservative compared to $O(1/L)$. We will show that the step-size of NIDS can have the same upper bound $2/L$ as that of the centralized gradient descent. The achievable step-sizes of NIDS for $o(1/k)$ rate in the general convex case and the linear convergence rate in the strongly convex case are both in the order of $O(1/L)$. Furthermore, NIDS allows each agent to have an individual step-size. Each agent i can choose a step-size α_i that is as large as $2/L_i$ on any connected network, where L_i is the Lipschitz constant of $\nabla s_i(x)$ and $L_i \leq L$ for all i ($L = \max_i L_i$). Apart from the step-sizes, to run NIDS, a common/public parameter c is needed for the construction of $\widetilde{\mathbf{W}}$ (see (8) for the algorithm and $\widetilde{\mathbf{W}}$). This parameter c can be chosen without any knowledge of the network (or the mixing matrix $\mathbf{W}$). For example, $c = 1/(2\max_i \alpha_i)$. Table I provides an overview of algorithmic configurations for EXTRA and NIDS. NIDS works as long as each agent can estimate its local functional parameter: No agent needs any other global information including the number of agents in the whole network except the largest step-size, if it is not the same for all agents.

In the line of research of optimization over heterogeneous networks, after the initial work [38] regarding uncoordinated step sizes, references [49] and [50] introduce and analyze a diffusion strategy with corrections that achieves an exact linear convergence with uncoordinated step sizes. This exact diffusion algorithm is still related to the Lagrangian method but can be considered as having incorporated a CTA structure. The CTA strategy can, similar to the ATC strategy, improve the convergence speed of certain consensus optimization algorithms (see [41, Remark 3] and [46, Section II.C]). However, the analysis in [50], though allowing step size mismatch across the network, does not take into consideration the heterogeneity of agents' functional conditions. Furthermore, their upper bound for the step-size is in the order of $O(\mu/L^2)$.

Sublinear convergence rate for the general case: Under the general convexity assumption, we show that NIDS has a convergence rate of $o(1/k)$, which is slightly better than the $O(1/k)$ rate of PG-EXTRA. Because the step-size of NIDS does not depend on the network topology and is much larger than that of PG-EXTRA, NIDS can be much faster than PG-EXTRA, as shown in the numerical experiments.

Linear convergence rate for the strongly convex case: Let us first define "scalability". When the iterate x^k of an algorithm converges to the optimal solution x^* linearly, i.e., $\|x^k - x^*\|^2 = O((1-1/S)^k)$ with some positive constant S, we say that the algorithm needs to run $O(S)\log(1/\epsilon)$ iterations to reach ϵ-accuracy. So we call $O(S)$ the scalability of the algorithm.

For the case where the non-smooth terms are absent and the functions $\{s_i\}_{i=1}^n$ are strongly convex, we show that NIDS achieves a linear convergence rate whose dependencies on the functions $\{s_i\}_{i=1}^n$ and the network topology are decoupled. To be specific, to reach ϵ-accuracy, the number of iterations needed for NIDS is

$$O\left(\max\left(\frac{L}{\mu}, \frac{1-\lambda_n(\mathbf{W})}{1-\lambda_2(\mathbf{W})}\right)\right)\log\frac{1}{\epsilon},$$

where $\lambda_i(\mathbf{W})$ is the ith largest eigenvalue of $\mathbf{W}$. Both $\frac{L}{\mu}$ and $\frac{1-\lambda_n(\mathbf{W})}{1-\lambda_2(\mathbf{W})}$ are typical in the literature of optimization and average consensus, respectively. The value $\frac{L}{\mu}$, also called the condition number of the objective function, is aligned with the scalability of the standard gradient descent [35]. The value $\frac{1-\lambda_n(\mathbf{W})}{1-\lambda_2(\mathbf{W})}$ is understood as the condition number[4] of the network and aligned with the scalability of the simplest linear iterations for distributed averaging [51].

Separating the condition numbers of the objective function and the network provides a way to determine the bottleneck of NIDS for a specific problem and a given network. Therefore, the system designer might be able to smartly apply preconditioning on $\{s_i\}_{i=1}^n$ or improve the connectivity of the network to cost-effectively obtain a better convergence.

Summary and comparison of state-of-the-art algorithms: We list the properties of a few relevant algorithms in Table II. We let $\sigma := \frac{1-\lambda_n(\mathbf{W})}{1-\lambda_2(\mathbf{W})}$. This quantity is directly affected by the network topology and how the matrix $\mathbf{W}$ is defined, thus is also

[4]When we choose $\mathbf{W} = \mathbf{I} - \tau\mathbf{Ł}$ where $\mathbf{Ł}$ is the Laplacian of the underlying graph and τ is a positive tunable constant, we have $\frac{1-\lambda_n(\mathbf{W})}{1-\lambda_2(\mathbf{W})} = \frac{\lambda_1(\mathbf{Ł})}{\lambda_{n-1}(\mathbf{Ł})}$ which is the finite cwondition number of $\mathbf{Ł}$. Note that $\lambda_n(\mathbf{Ł}) = 0$.

TABLE II
SUMMARY OF A FEW RELEVANT ALGORITHMS. HERE, μ (OR μ_g OR $\bar{\mu}$) IS THE STRONG CONVEXITY CONSTANT OF THE OBJECTIVE FUNCTION (OR THAT OF A MODIFIED OBJECTIVE FUNCTION); L (OR $\bar{L}$) IS THE LIPSCHITZ CONSTANT OF THE OBJECTIVE GRADIENT (OR ITS MODIFIED VERSION). ALSO $\sigma = (1-\lambda_n(\mathbf{W}))/(1-\lambda_2(\mathbf{W}))$ IS CONSIDERED AS THE CONDITION NUMBER OF THE NETWORK, WHICH SCALES AT THE ORDER OF $O(n^2)$ IN THE WORST CASE SCENARIO. WE OMIT "$O(\cdot)$" IN "ORDERS OF STEP-SIZE BOUNDS" AND "SCALABILITY" FOR BREVITY. NOTE QUANTITIES INVOLVING K ONLY HOLD FOR A FINITE K

Algorithm	Support prox. operators	Orders of step-size bounds (strongly convex; convex)	Uncoordinated step-size	Scalability (strongly convex)	Rate (convex)
EXTRA [1], [36]	Yes	$\frac{\mu_g}{L^2}$; $\frac{1}{L}$	No	$\left(\frac{L}{\mu_g}\right)^2$	$o\left(\frac{1}{k}\right)$
Aug-DGM [38]	No	small enough	Yes	–	converges
Harness [40]	No	$\frac{\mu}{L^2\sigma^2}$; $\frac{1}{L\sigma^2}$	No	$\left(\frac{L}{\mu}\sigma\right)^2$	$O\left(\frac{1}{k}\right)$
Acc-DNGD [52]	No	$\frac{(\sigma-1)^3}{\sigma^6 L}\left(\frac{\mu}{L}\right)^{3/7}$; $\frac{\min\{(1-\sigma^{-1})^2,(\sigma)^{-3}\}}{Lk^{0.6}}$	No	$\frac{\sigma^3}{(\sigma-1)^{1.5}}\left(\frac{L}{\mu}\right)^{5/7}$	$O\left(\frac{1}{k^{1.4}}\right)$
DIGing [41], [42]	No	$\min\{\frac{\bar{\mu}^{0.5}}{\sigma(\sigma-1)L^{1.5}n^{0.5}}, \frac{1}{L}\}$; –	Yes	$\max\{\frac{L}{\bar{\mu}}, \frac{(\sigma-1)^2 n^{0.5} L^{1.5}+\sigma^2\bar{\mu}^{1.5}}{\bar{\mu}^{1.5}}\}$	–
Optimal [53], [54]	No	μ; $\frac{L\sigma}{K^2}$	No	$\left(\frac{L}{\mu}\sigma\right)^{0.5}$	$O\left(\frac{1}{K^2}\right)$
NIDS	Yes	$\frac{1}{L}$; $\frac{1}{L}$	Yes	$\max\{\frac{L}{\mu}, \sigma\}$	$o(\frac{1}{k})$

related to the consensus ability of a network. When the network is fully connected (a complete graph), we can choose $\mathbf{W}$ so that $\lambda_2(\mathbf{W}) = \lambda_n(\mathbf{W}) = 0$ and thus $\sigma = 1$ (the best case); in general $\sigma \geq 1$ since $0 < 1-\lambda_2(\mathbf{W}) \leq 1-\lambda_n(\mathbf{W}) < 2$; in the worst case, we have $\sigma \leq \frac{1}{1-\lambda_2(\mathbf{W})} = O(n^2)$ [4, Section 2.3]. We keep σ in the bounds/rates of involved algorithms for a fair comparison instead of focusing on the worst case that often gives pessimistic/conservative results. We omit "$O(\cdot)$" in "Bounds of step-sizes" and "Scalabilities" for brevity and only compare the effect of functional properties (μ and L) and network properties (σ and/or n). Before talking into details, let us clarify a few points. In EXTRA, μ_g is a quantity that is associated with the strong convexity of the original function $\bar{f}(x)$, so it covers a larger class of problems; In DIGing, $\bar{\mu}$ is the mean value of the strong convexity constants of local objectives; In Acc-DNGD, the step-size for the convex case contains k, the current number of iterations. Thus it represents a diminishing step-size sequence; In Optimal [53], [54], the total number of iterations K is used to determine the step-size for the convex case. In addition, they apply to problems in which the objectives are dual friendly (see [54] for its definition). Note some types of objectives are suitable for gradient update, some are suitable for dual gradient update (dual friendly), and some are suitable for proximal update.

Finding the algorithm with the lowest per-iteration cost depends on the problem (functions). Apparently, our bounds on step-sizes and the corresponding scalability/rate are better than those given in EXTRA and Harness (see Table II). When σ is close to 1 (the graph is well connected), the step-size bound and scalability given in DIGing are the same as NIDS. However, when σ is large, their result becomes rather conservative. Acc-DNGD and Optimal have improved the scalability/rate of gradient-based distributed optimization by employing Nesterov's acceleration technique on primal and dual problems, respectively. For the convex case, our rate is worse than theirs because our algorithm does not employ Nesterov's acceleration. For the primal distributed gradient method after acceleration [52], the scalability in σ is still worse than our result. Algorithm Optimal achieves the optimal scalability/rate for distributed optimization. However, as we have mentioned above, their algorithms are dual based thus apply to a different class of problems. In addition, NIDS supports proximable non-smooth functions and uncoordinated step-sizes while these have not been considered in Acc-DNGD and Optimal. To sum up, we have reached the best possible performance of first-order algorithms for distributed optimization without acceleration. Further improving the performance by incorporating Nesterov's techniques to our algorithm will be a future direction.

Finally, we note that, references [49], [50], appearing simultaneously with this work, also proposed (6b) to enlarge the step-size and use column stochastic matrices rather than symmetric doubly stochastic matrices. However, their algorithm only works for smooth problems, and their analysis seems to be restrictive and requires twice differentiability and strong convexity of $\{s_i\}_{i=1}^n$. The stepsize is also in the order of μ/L^2 [48].

C. Future Works

The capability of our algorithm using purely locally determined parameters increases its potential to be extended to dynamic networks with a time-varying number of nodes. Given such flexibility, we may use similar schemes to solve the decentralized empirical risk minimization problems. Furthermore, it also enhances the privacy of the agents through allowing each agent to perform their own optimization procedure without negotiation on any parameter.

By using Nesterov's acceleration technique, reference [4] shows that the scalability of a new average consensus protocol can be improved to $O(n)$; when the nonsmooth terms r_i's are absent, reference [53] shows that the scalability of a new dual based accelerated distributed gradient method can be improved

to $O(\sqrt{\sigma L/\mu})$. One of our future work is exploring the convergence rates/scalability of the Nesterov's accelerated version of our algorithm.

D. Paper Organization

The rest of this paper is organized as follows. To facilitate the description of the technical ideas, the algorithms, and the analysis, we introduce additional notation in Subsection I-E. The intuition for the network-independent step-size is provided in Section II. In Section III, we introduce our algorithm NIDS and discuss its relation to some other existing algorithms. In Section IV, we first show that NIDS can be understood as an iterative algorithm for seeking a fixed point. Following this, we establish that NIDS converges at an $o(1/k)$ rate for the general convex case and a linear rate for the strongly convex case. Then, numerical simulations are given in Section V to corroborate our theoretical claims. Final remarks are given in Section VI.

E. Notation

We use bold upper-case letters such as $\mathbf{W}$ to define matrices in $\mathbb{R}^{n\times n}$ and bold lower-case letters such as $\mathbf{x}$ and $\mathbf{z}$ to define matrices in $\mathbb{R}^{n\times p}$ (when $p=1$, they are vectors). Let $\mathbf{1}$ and $\mathbf{0}$ be matrices with all ones and zeros, respectively, and their dimensions are provided when necessary. For matrices $\mathbf{x},\ \mathbf{y}\in\mathbb{R}^{n\times p}$, we define their inner product as $\langle\mathbf{x},\mathbf{y}\rangle=\mathrm{tr}(\mathbf{x}^\top\mathbf{y})$ and the norm as $\|\mathbf{x}\|=\sqrt{\langle\mathbf{x},\mathbf{x}\rangle}$. Additionally, by an abuse of notation, we define $\langle\mathbf{x},\mathbf{y}\rangle_{\mathbf{Q}}=\mathrm{tr}(\mathbf{x}^\top\mathbf{Q}\mathbf{y})$ and $\|\mathbf{x}\|_{\mathbf{Q}}^2=\langle\mathbf{x},\mathbf{x}\rangle_{\mathbf{Q}}$ for any given symmetric matrix $\mathbf{Q}\in\mathbb{R}^{n\times n}$. Note that $\langle\cdot,\cdot\rangle_{\mathbf{Q}}$ is an inner product defined in $\mathbb{R}^{n\times p}$ if and only if $\mathbf{Q}$ is positive definite. However, when $\mathbf{Q}$ is not positive definite, $\langle\mathbf{x},\mathbf{y}\rangle_{\mathbf{Q}}$ can still be an inner product defined in a subspace of $\mathbb{R}^{n\times p}$, see Lemma 3 for more details. We define the range of $\mathbf{A}\in\mathbb{R}^{n\times n}$ by $\mathbf{range}(\mathbf{A}):=\{\mathbf{x}\in\mathbb{R}^{n\times p}:\mathbf{x}=\mathbf{A}\mathbf{y},\ \mathbf{y}\in\mathbb{R}^{n\times p}\}$. The largest eigenvalue of a symmetric matrix $\mathbf{A}$ is also denoted as $\lambda_{\max}(\mathbf{A})$. For two symmetric matrices $\mathbf{A},\mathbf{B}\in\mathbb{R}^{n\times n}$, $\mathbf{A}\succ\mathbf{B}$ (or $\mathbf{A}\succcurlyeq\mathbf{B}$) means that $\mathbf{A}-\mathbf{B}$ is positive definite (or positive semidefinite). Moreover, we use $\mathcal{N}_i$ to represent the set of agents that can directly send messages to agent i.

II. Intuition for Network-Independent Step-Size

In this section, we provide an intuition for the network-independent step-size for NIDS with only the differentiable function s. The decentralized optimization problem is equivalent to

$$\underset{\mathbf{x}}{\text{minimize}}\ s(\mathbf{x}),\quad \text{s.t.}\quad (\mathbf{I}-\mathbf{W})^{1/2}\mathbf{x}=\mathbf{0},$$

where $(\mathbf{I}-\mathbf{W})^{1/2}$ is the square root of $\mathbf{I}-\mathbf{W}$, and the constraint is the same as the consensual condition with the mixing matrix $\mathbf{W}$ given in Assumption 1. Denote $\mathbf{Ł}=(\mathbf{I}-\mathbf{W})^{1/2}$. The corresponding optimality condition with the introduction of the dual variable $\mathbf{p}$ is

$$\begin{bmatrix}0 & \mathbf{Ł}\\ -\mathbf{Ł} & 0\end{bmatrix}\begin{bmatrix}\mathbf{x}^*\\ \mathbf{p}^*\end{bmatrix}=-\begin{bmatrix}\nabla s(\mathbf{x}^*)\\ 0\end{bmatrix}.$$

EXTRA is equivalent to the Condat-Vu primal-dual algorithm [47], [55], and it can be further explained as a forward-backward splitting applied to the equation, i.e.,

$$\left[\begin{bmatrix}\frac{1}{\alpha}\mathbf{I} & -\mathbf{Ł}\\ -\mathbf{Ł} & 2\alpha\mathbf{I}\end{bmatrix}+\begin{bmatrix}0 & \mathbf{Ł}\\ -\mathbf{Ł} & 0\end{bmatrix}\right]\begin{bmatrix}\mathbf{x}^{k+1}\\ \mathbf{p}^{k+1}\end{bmatrix}$$
$$=\begin{bmatrix}\frac{1}{\alpha}\mathbf{I} & -\mathbf{Ł}\\ -\mathbf{Ł} & 2\alpha\mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{x}^{k}\\ \mathbf{p}^{k}\end{bmatrix}-\begin{bmatrix}\nabla s(\mathbf{x}^k)\\ 0\end{bmatrix}.$$

The update is

$$\mathbf{x}^{k+1}=\mathbf{x}^k-\alpha\mathbf{Ł}\mathbf{p}^k-\alpha\nabla s(\mathbf{x}^k),$$
$$\alpha\mathbf{p}^{k+1}=\alpha\mathbf{p}^k-\frac{1}{2}\mathbf{Ł}\mathbf{x}^k+\mathbf{Ł}\mathbf{x}^{k+1}.$$

It is equivalent to EXTRA after $\mathbf{p}$ is eliminated. In this case, the new metric is a full matrix, and therefore, the upper bound of the step-size α depends on the matrix $\mathbf{Ł}$. To be more specific,

$$\begin{bmatrix}\frac{1}{\alpha}\mathbf{I} & -\mathbf{Ł}\\ -\mathbf{Ł} & 2\alpha\mathbf{I}\end{bmatrix}\succcurlyeq\begin{bmatrix}\frac{L}{2}\mathbf{I} & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix},$$

which gives $\alpha\le 2(1+\lambda_{\min}(\mathbf{W}))/L$. A larger and optimal upper bound for the step-size of EXTRA is shown in [47] (See Table I), and it still depends on $\mathbf{W}$. However, we choose a block diagonal metric and have

$$\left[\begin{bmatrix}\frac{1}{\alpha}\mathbf{I} & 0\\ 0 & \alpha(\mathbf{I}+\mathbf{W})\end{bmatrix}+\begin{bmatrix}0 & \mathbf{Ł}\\ -\mathbf{Ł} & 0\end{bmatrix}\right]\begin{bmatrix}\mathbf{x}^{k+1}\\ \mathbf{p}^{k+1}\end{bmatrix}$$
$$=\begin{bmatrix}\frac{1}{\alpha}\mathbf{I} & 0\\ 0 & \alpha(\mathbf{I}+\mathbf{W})\end{bmatrix}\begin{bmatrix}\mathbf{x}^{k}\\ \mathbf{p}^{k}\end{bmatrix}-\begin{bmatrix}\nabla s(\mathbf{x}^k)\\ 0\end{bmatrix}.$$

The update becomes

$$2\alpha\mathbf{p}^{k+1}=\alpha(\mathbf{I}+\mathbf{W})\mathbf{p}^k+\mathbf{Ł}\mathbf{x}^k-\alpha\mathbf{Ł}\nabla s(\mathbf{x}^k),$$
$$\mathbf{x}^{k+1}=\mathbf{x}^k-\alpha\nabla s(\mathbf{x}^k)-\alpha\mathbf{Ł}\mathbf{p}^{k+1},$$

which is equivalent to NIDS after $\mathbf{p}$ is eliminated. Because the new metric is block diagonal, and the nonexpansiveness of the forward step depends on the function only, i.e., $\alpha\le 2/L$.

III. Proposed Algorithm NIDS

In this section, we describe our proposed NIDS in Algorithm 1 for solving (1) in more details and explain the connections to other related methods.

The mixing matrix satisfies the following assumption, which comes from [1], [36].

Assumption 1 (Mixing matrix): The connected network $\mathcal{G}=\{\mathcal{V},\mathcal{E}\}$ consists of a set of agents $\mathcal{V}=\{1,2,\dots,n\}$ and a set of undirected edges $\mathcal{E}$. An undirected edge $(i,j)\in\mathcal{E}$ means that there is a connection between agents i and j and both agents can exchange data. The mixing matrix $\mathbf{W}=[w_{ij}]\in\mathbb{R}^{n\times n}$ satisfies:

1) (Decentralized property). If $i\neq j$ and $(i,j)\notin\mathcal{E}$, then $w_{ij}=0$;
2) (Symmetry). $\mathbf{W}=\mathbf{W}^T$;
3) (Null space property). $\mathbf{Null}(\mathbf{I}-\mathbf{W})=\mathbf{span}(\mathbf{1}_{n\times 1})$;
4) (Spectral property). $2\mathbf{I}\succcurlyeq\mathbf{W}+\mathbf{I}\succ\mathbf{0}_{n\times n}$.

Remark 1: Assumption 1 implies that the eigenvalues of $\mathbf{W}$ lie in $(-1,1]$ and the multiplicity of eigenvalue 1 is one, i.e.,

Algorithm 1: NIDS.

Each agent i obtains its mixing values $w_{ij}, \forall j \in \mathcal{N}_i$;
Each agent i chooses its own step-size $\alpha_i > 0$ and the same parameter c (e.g., $c = 0.5/\max_i \alpha_i$);
Each agent i sets the mixing values $\widetilde{w}_{ij} := c\alpha_i w_{ij}$, $\forall j \in \mathcal{N}_i$ and $\widetilde{w}_{ii} := 1 - c\alpha_i + c\alpha_i w_{ii}$;
Each agent i picks arbitrary initial $x_i^0 \in \mathbb{R}^p$ and performs

$$z_i^1 = x_i^0 - \alpha_i \nabla s_i\left(x_i^0\right),$$

$$x_i^1 = \arg\min_{x\in\mathbb{R}^p} \alpha_i r_i(x) + \frac{1}{2}\left\|x - z_i^1\right\|^2.$$

for $k = 1, 2, 3 \ldots$ **do**
Each agents i performs

$$z_i^{k+1} = z_i^k - x_i^k + \sum_{j=\mathcal{N}_i\cup\{i\}} \widetilde{w}_{ij}\left(2x_j^k - x_j^{k-1} - \alpha_j\nabla s_j\left(x_j^k\right) + \alpha_j \nabla s_j\left(x_j^{k-1}\right)\right),$$

$$x_i^{k+1} = \arg\min_{x\in\mathbb{R}^p} \alpha_i r_i(x) + \frac{1}{2}\left\|x - z_i^{k+1}\right\|^2.$$

end for

$1 = \lambda_1(\mathbf{W}) > \lambda_2(\mathbf{W}) \geq \cdots \geq \lambda_n(\mathbf{W}) > -1$. Item 3 of Assumption 1 shows that $(\mathbf{I} - \mathbf{W})\mathbf{1}_{n\times 1} = \mathbf{0}$ and the orthogonal complement of $\mathbf{span}(\mathbf{1}_{n\times 1})$ is the row space of $\mathbf{I} - \mathbf{W}$, which is also the column space of $\mathbf{I} - \mathbf{W}$ because of the symmetry of $\mathbf{W}$.

The functions $\{s_i\}_{i=1}^n$ and $\{r_i\}_{i=1}^n$ satisfy the following assumption.

Assumption 2: Functions $\{s_i(x)\}_{i=1}^n$ and $\{r_i(x)\}_{i=1}^n$ are lower semi-continuous proper convex, and $\{s_i(x)\}_{i=1}^n$ have Lipschitz continuous gradients with constants $\{L_i\}_{i=1}^n$, respectively. Thus, we have

$$\langle \mathbf{x} - \mathbf{y}, \nabla s(\mathbf{x}) - \nabla s(\mathbf{y})\rangle \geq \|\nabla s\left(\mathbf{x}\right) - \nabla s\left(\mathbf{y}\right)\|^2_{\mathbf{L}^{-1}}, \quad (7)$$

where $\mathbf{L} = \mathrm{Diag}(L_1, \ldots, L_n)$ is the diagonal matrix with the Lipschitz constants [35].

Instead of using the same step-size for all the agents, we allow agent i to choose its own step-size α_i and let $\Lambda = \mathrm{Diag}(\alpha_1, \ldots, \alpha_n) \in \mathbb{R}^{n\times n}$. Then NIDS can be expressed as

$$\mathbf{z}^{k+1} = \mathbf{z}^k - \mathbf{x}^k + \widetilde{\mathbf{W}}(2\mathbf{x}^k - \mathbf{x}^{k-1} - \Lambda\nabla s(\mathbf{x}^k) + \Lambda\nabla s(\mathbf{x}^{k-1})), \quad (8a)$$

$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}\in\mathbb{R}^{n\times p}} r(\mathbf{x}) + \frac{1}{2}\|\mathbf{x} - \mathbf{z}^{k+1}\|^2_{\Lambda^{-1}}, \quad (8b)$$

where $\widetilde{\mathbf{W}} = \mathbf{I} - c\Lambda(\mathbf{I} - \mathbf{W})$ and c is chosen such that $\Lambda^{-1/2}\widetilde{\mathbf{W}}\Lambda^{1/2} = \mathbf{I} - c\Lambda^{1/2}(\mathbf{I} - \mathbf{W})\Lambda^{1/2} \succcurlyeq \mathbf{0}$. This condition shows that the upper bound of the parameter c depends on $\mathbf{W}$ and Λ. When the information about $\mathbf{W}$ is not given, we can just let $c = 1/(2\max_i \alpha_i)$ because $\lambda_n(\mathbf{W}) > -1$. To set such a parameter, a preprocessing step is needed to obtain the maximum. However, since the maximum can be easily computed in a connected network in no more than $n - 1$ rounds of communication wherein each node repeatedly takes maximum of the values from neighbors, the cost of this preprocessing is essentially negligible compared to the worst-case running time of our optimization protocol.

If all agents choose the same step-size, i.e., $\Lambda = \alpha\mathbf{I}$, and we let $c = 1/(2\alpha)$, (8) becomes

$$\mathbf{z}^{k+1} = \mathbf{z}^k - \mathbf{x}^k + \frac{\mathbf{I} + \mathbf{W}}{2}(2\mathbf{x}^k - \mathbf{x}^{k-1} - \alpha\nabla s(\mathbf{x}^k) + \alpha\nabla s(\mathbf{x}^{k-1})), \quad (9a)$$

$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}\in\mathbb{R}^{n\times p}} r(\mathbf{x}) + \frac{1}{2\alpha}\|\mathbf{x} - \mathbf{z}^{k+1}\|^2. \quad (9b)$$

Remark 2: The update of PG-EXTRA is

$$\mathbf{z}^{k+1} = \mathbf{z}^k - \mathbf{x}^k + \frac{\mathbf{I} + \mathbf{W}}{2}(2\mathbf{x}^k - \mathbf{x}^{k-1}) - \alpha\nabla s(\mathbf{x}^k) + \alpha\nabla s(\mathbf{x}^{k-1}), \quad (10a)$$

$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}\in\mathbb{R}^{n\times p}} r(\mathbf{x}) + \frac{1}{2\alpha}\|\mathbf{x} - \mathbf{z}^{k+1}\|^2. \quad (10b)$$

The only difference between NIDS and PG-EXTRA is that the mixing operation is further applied to the successive difference of the gradients $-\alpha\nabla s(\mathbf{x}^k) + \alpha\nabla s(\mathbf{x}^{k-1})$ in NIDS.

When there is no function $r(\mathbf{x})$, (8) becomes

$$\mathbf{x}^{k+1} = \widetilde{\mathbf{W}}(2\mathbf{x}^k - \mathbf{x}^{k-1} - \Lambda\nabla s(\mathbf{x}^k) + \Lambda\nabla s(\mathbf{x}^{k-1})),$$

and it further reduces to (6b) when $\Lambda = \alpha\mathbf{I}$ and $c = 1/(2\alpha)$. Note that, though (6b) appears in [49], [50], its convergence still needs a small step-size that also depends on the network topology and the strong convexity constant. In Theorem 1 of [50], the upper bound for the step-size is also $O(\mu/L^2)$, which is the same as that of PG-EXTRA.

IV. Convergence Analysis of NIDS

In order to show the convergence of NIDS, we also need the following assumption.

Assumption 3 (Solution existence): Problem (1) has at least one solution.

To simplify the analysis, we introduce a new sequence $\{\mathbf{d}^k\}_{k\geq 0}$ which is defined as

$$\mathbf{d}^k := \Lambda^{-1}(\mathbf{x}^{k-1} - \mathbf{z}^k) - \nabla s\left(\mathbf{x}^{k-1}\right). \quad (11)$$

Using the sequence $\{\mathbf{x}^k\}_{k\geq 0}$, we obtain a recursive (update) relation for $\{\mathbf{d}^k\}_{k\geq 0}$:

$$\begin{aligned}
\mathbf{d}^{k+1} &= \Lambda^{-1}(\mathbf{x}^k - \mathbf{z}^{k+1}) - \nabla s(\mathbf{x}^k)\\
&= \Lambda^{-1}(\mathbf{x}^k - \mathbf{z}^k + \mathbf{x}^k) - \nabla s(\mathbf{x}^k)\\
&\quad - \Lambda^{-1}\widetilde{\mathbf{W}}(2\mathbf{x}^k - \mathbf{x}^{k-1} - \Lambda\nabla s(\mathbf{x}^k) + \Lambda\nabla s(\mathbf{x}^{k-1}))\\
&= \Lambda^{-1}(\mathbf{x}^k - \mathbf{z}^k + \mathbf{x}^k - 2\mathbf{x}^k + \mathbf{x}^{k-1})\\
&\quad - \nabla s(\mathbf{x}^k) + \nabla s(\mathbf{x}^k) - \nabla s(\mathbf{x}^{k-1})\\
&\quad + c(\mathbf{I} - \mathbf{W})(2\mathbf{x}^k - \mathbf{x}^{k-1} - \Lambda\nabla s(\mathbf{x}^k) + \Lambda\nabla s(\mathbf{x}^{k-1}))\\
&= \mathbf{d}^k + c(\mathbf{I} - \mathbf{W})(2\mathbf{x}^k - \mathbf{z}^k - \Lambda\nabla s(\mathbf{x}^k) - \Lambda\mathbf{d}^k),
\end{aligned}$$

where the second equality comes from the update of $\mathbf{z}^{k+1}$ in (8a) and the last one holds because of the definition of $\mathbf{d}^k$ in (11). Therefore, the iteration (8) is equivalent to, with the update order $(\mathbf{x}, \mathbf{d}, \mathbf{z})$,

$$\mathbf{x}^k = \underset{\mathbf{x}\in\mathbb{R}^{n\times p}}{\arg\min}\ r(\mathbf{x}) + \frac{1}{2}\|\mathbf{x} - \mathbf{z}^k\|^2_{\Lambda^{-1}}, \tag{12a}$$

$$\begin{aligned}\mathbf{d}^{k+1} = \mathbf{d}^k + c(\mathbf{I}-\mathbf{W})\left(2\mathbf{x}^k - \mathbf{z}^k\right.\\ \left. -\Lambda\nabla s\left(\mathbf{x}^k\right) - \Lambda\mathbf{d}^k\right),\end{aligned} \tag{12b}$$

$$\mathbf{z}^{k+1} = \mathbf{x}^k - \Lambda\nabla s\left(\mathbf{x}^k\right) - \Lambda\mathbf{d}^{k+1}, \tag{12c}$$

in the sense that both (8) and (12) generate the same $\{\mathbf{x}^k, \mathbf{z}^k\}_{k>0}$ sequence.

Because $\mathbf{x}^k$ is determined by $\mathbf{z}^k$ only and can be eliminated from the iteration, iteration (12) is essentially an operator for $(\mathbf{d}, \mathbf{z})$. Note that we have $\mathbf{d}^1 = \Lambda^{-1}(\mathbf{x}^0 - \mathbf{z}^1) - \nabla s\left(\mathbf{x}^0\right) = \mathbf{0}$ from Algorithm 1. Therefore, from the update of $\mathbf{d}^{k+1}$ in (12b), $\mathbf{d}^k \in \mathbf{range}(\mathbf{I}-\mathbf{W})$ for all k. In fact, any $\mathbf{z}^1$ such that $\mathbf{d}^1 \in \mathbf{range}(\mathbf{I}-\mathbf{W})$ works for NIDS. The following two lemmas show the relation between fixed points of (12) and optimal solutions of (1). The proofs for all lemmas and propositions are included in the supplemental material.

Lemma 1 (Fixed point of (12)): $(\mathbf{d}^*, \mathbf{z}^*)$ is a fixed point of (12) if and only if there exists a subgradient $\mathbf{q}^* \in \partial r(\mathbf{x}^*)$ such that $\mathbf{z}^* = \mathbf{x}^* + \Lambda\mathbf{q}^*$ and

$$\mathbf{d}^* + \nabla s(\mathbf{x}^*) + \mathbf{q}^* = \mathbf{0}, \tag{13a}$$

$$(\mathbf{I}-\mathbf{W})\mathbf{x}^* = \mathbf{0}. \tag{13b}$$

Lemma 2 (Optimality condition): $\mathbf{x}^*$ is consensual with $x_1^* = x_2^* = \cdots = x_n^* = x^*$ being an optimal solution of problem (1) if and only if there exists $\mathbf{p}^*$ and a subgradient $\mathbf{q}^* \in \partial r(\mathbf{x}^*)$ such that:

$$(\mathbf{I}-\mathbf{W})\mathbf{p}^* + \nabla s(\mathbf{x}^*) + \mathbf{q}^* = \mathbf{0}, \tag{14a}$$

$$(\mathbf{I}-\mathbf{W})\mathbf{x}^* = \mathbf{0}. \tag{14b}$$

In addition, $(\mathbf{d}^* = (\mathbf{I}-\mathbf{W})\mathbf{p}^*, \mathbf{z}^* = \mathbf{x}^* + \Lambda\mathbf{q}^*)$ is a fixed point of iteration (12).

Lemma 2 shows that we can find a fixed point of iteration (12) to obtain an optimal solution of problem (1). It also tells us that we need $\mathbf{d}^* \in \mathbf{range}(\mathbf{I}-\mathbf{W})$ to get an optimal solution of problem (1). Therefore, we need $\mathbf{d}^1 \in \mathbf{range}(\mathbf{I}-\mathbf{W})$.

Lemma 3 (Norm over range space): For any symmetric positive semidefinite matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ with rank $r \le n$, let $\lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_r > 0$ be its r eigenvalues. Then $\mathbf{range}(\mathbf{A})$ defined in Section I-E is a rp-dimensional subspace in $\mathbb{R}^{n\times p}$ and has a norm defined by $\|\mathbf{x}\|^2_{\mathbf{A}^\dagger} := \langle \mathbf{x}, \mathbf{A}^\dagger\mathbf{x}\rangle$, where $\mathbf{A}^\dagger$ is the pseudo inverse of $\mathbf{A}$. In addition, $\lambda_1^{-1}\|\mathbf{x}\|^2 \le \|\mathbf{x}\|^2_{\mathbf{A}^\dagger} \le \lambda_r^{-1}\|\mathbf{x}\|^2$ for all $\mathbf{x} \in \mathbf{range}(\mathbf{A})$.

Proposition 1: Let $\mathbf{M} = c^{-1}(\mathbf{I}-\mathbf{W})^\dagger - \Lambda$ with $\mathbf{I} \succcurlyeq c\Lambda^{1/2}(\mathbf{I}-\mathbf{W})\Lambda^{1/2} \succcurlyeq \mathbf{0}$. Then $\|\cdot\|_{\mathbf{M}}$ is a norm defined for $\mathbf{range}(\mathbf{I}-\mathbf{W})$.

The following lemma compares the distance to a fixed point of (12) for two consecutive iterates.

Lemma 4 (Fundamental inequality): Let $(\mathbf{d}^*, \mathbf{z}^*)$ be a fixed point of iteration (12) with $\mathbf{d}^* \in \mathbf{range}(\mathbf{I}-\mathbf{W})$. The update $(\mathbf{d}^k, \mathbf{z}^k) \Rightarrow (\mathbf{d}^{k+1}, \mathbf{z}^{k+1})$ in (12) satisfies

$$\begin{aligned}&\|\mathbf{z}^{k+1} - \mathbf{z}^*\|^2_{\Lambda^{-1}} + \|\mathbf{d}^{k+1} - \mathbf{d}^*\|^2_{\mathbf{M}}\\ &\le \|\mathbf{z}^k - \mathbf{z}^*\|^2_{\Lambda^{-1}} + \|\mathbf{d}^k - \mathbf{d}^*\|^2_{\mathbf{M}}\\ &\quad - \|\mathbf{z}^k - \mathbf{z}^{k+1}\|^2_{\Lambda^{-1}} - \|\mathbf{d}^k - \mathbf{d}^{k+1}\|^2_{\mathbf{M}}\\ &\quad + 2\langle\nabla s\left(\mathbf{x}^k\right) - \nabla s\left(\mathbf{x}^*\right), \mathbf{z}^k - \mathbf{z}^{k+1}\rangle\\ &\quad - 2\langle\mathbf{x}^k - \mathbf{x}^*, \nabla s(\mathbf{x}^k) - \nabla s(\mathbf{x}^*)\rangle.\end{aligned} \tag{15}$$

Proof: From the update of $\mathbf{z}^{k+1}$ in (12c), we have

$$\begin{aligned}&\langle\mathbf{d}^{k+1} - \mathbf{d}^*, \mathbf{z}^{k+1} - \mathbf{z}^k + \mathbf{x}^k - \mathbf{x}^*\rangle\\ &= \langle\mathbf{d}^{k+1} - \mathbf{d}^*, 2\mathbf{x}^k - \mathbf{z}^k - \Lambda\nabla s(\mathbf{x}^k) - \Lambda\mathbf{d}^{k+1} - \mathbf{x}^*\rangle\\ &= \langle\mathbf{d}^{k+1} - \mathbf{d}^*, c^{-1}(\mathbf{I}-\mathbf{W})^\dagger(\mathbf{d}^{k+1} - \mathbf{d}^k)\\ &\quad + \Lambda\mathbf{d}^k - \Lambda\mathbf{d}^{k+1}\rangle\\ &= \langle\mathbf{d}^{k+1} - \mathbf{d}^*, \mathbf{d}^{k+1} - \mathbf{d}^k\rangle_{\mathbf{M}},\end{aligned} \tag{16}$$

where the second equality comes from (12b), (14b), and $\mathbf{d}^{k+1} - \mathbf{d}^* \in \mathbf{range}(\mathbf{I}-\mathbf{W})$. From (12a), we have that

$$\langle\mathbf{x}^k - \mathbf{x}^*, \mathbf{z}^k - \mathbf{x}^k - \mathbf{z}^* + \mathbf{x}^*\rangle_{\Lambda^{-1}} \ge 0. \tag{17}$$

Therefore, we have

$$\begin{aligned}&\langle\mathbf{x}^k - \mathbf{x}^*, \nabla s(\mathbf{x}^k) - \nabla s(\mathbf{x}^*)\rangle\\ &\le \langle\mathbf{x}^k - \mathbf{x}^*, \Lambda^{-1}(\mathbf{z}^k - \mathbf{x}^k - \mathbf{z}^* + \mathbf{x}^*)\\ &\quad + \nabla s(\mathbf{x}^k) - \nabla s(\mathbf{x}^*)\rangle\\ &= \langle\mathbf{x}^k - \mathbf{x}^*, \Lambda^{-1}(\mathbf{z}^k - \mathbf{z}^{k+1}) - \mathbf{d}^{k+1} + \mathbf{d}^*\rangle\\ &= \langle\mathbf{x}^k - \mathbf{x}^*, \mathbf{z}^k - \mathbf{z}^{k+1}\rangle_{\Lambda^{-1}} + \langle\mathbf{d}^{k+1}\\ &\quad - \mathbf{d}^*, \mathbf{z}^{k+1} - \mathbf{z}^k\rangle - \langle\mathbf{d}^{k+1} - \mathbf{d}^*, \mathbf{d}^{k+1} - \mathbf{d}^k\rangle_{\mathbf{M}}\\ &= \langle\Lambda^{-1}(\mathbf{x}^k - \mathbf{x}^*) - \mathbf{d}^{k+1} + \mathbf{d}^*, \mathbf{z}^k - \mathbf{z}^{k+1}\rangle\\ &\quad - \langle\mathbf{d}^{k+1} - \mathbf{d}^*, \mathbf{d}^{k+1} - \mathbf{d}^k\rangle_{\mathbf{M}}\\ &= \langle\Lambda^{-1}(\mathbf{z}^{k+1} - \mathbf{z}^*) + \nabla s(\mathbf{x}^k) - \nabla s(\mathbf{x}^*), \mathbf{z}^k - \mathbf{z}^{k+1}\rangle\\ &\quad - \langle\mathbf{d}^{k+1} - \mathbf{d}^*, \mathbf{d}^{k+1} - \mathbf{d}^k\rangle_{\mathbf{M}}\\ &= \langle\mathbf{z}^{k+1} - \mathbf{z}^*, \mathbf{z}^k - \mathbf{z}^{k+1}\rangle_{\Lambda^{-1}}\\ &\quad + \langle\nabla s(\mathbf{x}^k) - \nabla s(\mathbf{x}^*), \mathbf{z}^k - \mathbf{z}^{k+1}\rangle\\ &\quad + \langle\mathbf{d}^{k+1} - \mathbf{d}^*, \mathbf{d}^k - \mathbf{d}^{k+1}\rangle_{\mathbf{M}}.\end{aligned}$$

The inequality and the second equality comes from (17) and (16), respectively. The first and fourth equalities hold because of the update of $\mathbf{z}^{k+1}$ in (12c). Using $2\langle\mathbf{a}, \mathbf{b}\rangle = \|\mathbf{a}+\mathbf{b}\|^2 - \|\mathbf{a}\|^2 - \|\mathbf{b}\|^2$ and rearranging the previous inequality give us that

$$\begin{aligned}&2\langle\mathbf{x}^k - \mathbf{x}^*, \nabla s(\mathbf{x}^k) - \nabla s(\mathbf{x}^*)\rangle\\ &\quad - 2\langle\nabla s(\mathbf{x}^k) - \nabla s(\mathbf{x}^*), \mathbf{z}^k - \mathbf{z}^{k+1}\rangle\\ &\le 2\langle\mathbf{z}^{k+1} - \mathbf{z}^*, \mathbf{z}^k - \mathbf{z}^{k+1}\rangle_{\Lambda^{-1}}\\ &\quad + 2\langle\mathbf{d}^{k+1} - \mathbf{d}^*, \mathbf{d}^k - \mathbf{d}^{k+1}\rangle_{\mathbf{M}}\\ &= \|\mathbf{z}^k - \mathbf{z}^*\|^2_{\Lambda^{-1}} - \|\mathbf{z}^{k+1} - \mathbf{z}^*\|^2_{\Lambda^{-1}} - \|\mathbf{z}^k - \mathbf{z}^{k+1}\|^2_{\Lambda^{-1}}\\ &\quad + \|\mathbf{d}^k - \mathbf{d}^*\|^2_{\mathbf{M}} - \|\mathbf{d}^{k+1} - \mathbf{d}^*\|^2_{\mathbf{M}} - \|\mathbf{d}^k - \mathbf{d}^{k+1}\|^2_{\mathbf{M}}.\end{aligned}$$

Therefore, (15) is obtained.

A. Sublinear Convergence of NIDS

As explained in Section II, NIDS is equivalent to the primal-dual algorithm [56] applied to problem

$$\underset{\mathbf{x}}{\text{minimize}}\ s(\mathbf{x})+r(\mathbf{x})+\iota((\mathbf{I}-\mathbf{W})^{1/2}\mathbf{x}), \tag{18}$$

where $\iota(\cdot)$ is the indicator function, which return 0 for $\mathbf{0}$ and $+\infty$ otherwise, with the metric matrix being

$$\begin{bmatrix}\Lambda^{-1} & \mathbf{0}\\ \mathbf{0} & c^{-1}\mathbf{I}-(\mathbf{I}-\mathbf{W})^{1/2}\Lambda(\mathbf{I}-\mathbf{W})^{1/2}\end{bmatrix}.$$

We apply [56, Theorem 1] and obtain the following sublinear convergence result.

Theorem 1 (Sublinear rate): Let $(\mathbf{d}^k,\mathbf{z}^k)$ be the sequence generated from NIDS in (12) with $\alpha_i<2/L_i$ for all i and $\mathbf{I}\succcurlyeq c\Lambda^{1/2}(\mathbf{I}-\mathbf{W})\Lambda^{1/2}$. We have

$$\begin{aligned}&\|\mathbf{z}^k-\mathbf{z}^{k+1}\|^2_{\Lambda^{-1}}+\|\mathbf{d}^k-\mathbf{d}^{k+1}\|^2_{\mathbf{M}}\\&\le\frac{\|\mathbf{z}^1-\mathbf{z}^*\|^2_{\Lambda^{-1}}+\|\mathbf{d}^1-\mathbf{d}^*\|^2_{\mathbf{M}}}{k(1-\max_i\frac{\alpha_iL_i}{2})},\end{aligned}$$

$$\|\mathbf{z}^k-\mathbf{z}^{k+1}\|^2_{\Lambda^{-1}}+\|\mathbf{d}^k-\mathbf{d}^{k+1}\|^2_{\mathbf{M}}=o\left(\frac{1}{k+1}\right). \tag{19}$$

Furthermore, $(\mathbf{d}^k,\mathbf{z}^k)$ converges to a fixed point $(\bar{\mathbf{d}},\bar{\mathbf{z}})$ of iteration (12) and $\bar{\mathbf{d}}\in\mathbf{range}(\mathbf{I}-\mathbf{W})$, if $\mathbf{I}\succ c\Lambda^{1/2}(\mathbf{I}-\mathbf{W})\Lambda^{1/2}$.

Remark 3: Note the convergence in Theorem 1 is shown in $\mathbf{z}$ and $\mathbf{d}$. We will show the convergence in terms of (14). Recall that

$$\begin{aligned}\mathbf{z}^{k+1}-\mathbf{z}^k&=\mathbf{x}^k-\Lambda\nabla s(\mathbf{x}^k)-\Lambda\mathbf{d}^{k+1}-\mathbf{z}^k\\&=-\Lambda(\mathbf{d}^{k+1}+\nabla s(\mathbf{x}^k)+\mathbf{q}^k),\end{aligned}$$

where $\mathbf{q}^k\in\partial r(\mathbf{x}^k)$. Therefore, $\|\mathbf{z}^{k+1}-\mathbf{z}^k\|^2_{\Lambda^{-1}}\to0$ implies the convergence in terms of (14a).

Combining (12b) and (12c), we have

$$\begin{aligned}\mathbf{d}^{k+1}=\mathbf{d}^k&+c(\mathbf{I}-\mathbf{W})(\mathbf{x}^k-\mathbf{z}^k+\mathbf{z}^{k+1})\\&+c(\mathbf{I}-\mathbf{W})\Lambda(\mathbf{d}^{k+1}-\mathbf{d}^k).\end{aligned}$$

Rearranging it gives

$$\begin{aligned}&(\mathbf{I}-c(\mathbf{I}-\mathbf{W})\Lambda)\left(\mathbf{d}^{k+1}-\mathbf{d}^k\right)\\&=c(\mathbf{I}-\mathbf{W})\left(\mathbf{x}^k-\mathbf{z}^k+\mathbf{z}^{k+1}\right).\end{aligned}$$

Then we have

$$\begin{aligned}&\|c(\mathbf{I}-\mathbf{W})\left(\mathbf{x}^k-\mathbf{z}^k+\mathbf{z}^{k+1}\right)\|^2\\&=\|c\left(\mathbf{I}-\mathbf{W}\right)\mathbf{M}^{1/2}\mathbf{M}^{1/2}\left(\mathbf{d}^{k+1}-\mathbf{d}^k\right)\|^2\\&\le\|c(\mathbf{I}-\mathbf{W})\mathbf{M}^{1/2}\|^2\left\|\mathbf{d}^{k+1}-\mathbf{d}^k\right\|^2_{\mathbf{M}},\end{aligned}$$

where the second equality comes from $\mathbf{d}^{k+1}-\mathbf{d}^k\in\mathbf{range}(\mathbf{I}-\mathbf{W})$. Thus $\|\mathbf{z}^{k+1}-\mathbf{z}^k\|^2_{\Lambda^{-1}}+\|\mathbf{d}^{k+1}-\mathbf{d}^k\|^2_{\mathbf{M}}\to0$ implies the convergence in terms of (14b).

B. Linear Convergence for Special Cases

In this subsection, we provide the linear convergence rate for the case when $r(\mathbf{x})=0$, i.e., $\mathbf{z}^k=\mathbf{x}^k$ in NIDS.

Theorem 2: If $\{s_i(x)\}_{i=1}^n$ are strongly convex with parameters $\{\mu_i\}_{i=1}^n$, then

$$\langle\mathbf{x}-\mathbf{y},\nabla s(\mathbf{x})-\nabla s(\mathbf{y})\rangle\ge\|\mathbf{x}-\mathbf{y}\|^2_{\mathbf{S}}, \tag{20}$$

where $\mathbf{S}=\mathrm{Diag}(\mu_1,\dots,\mu_n)\in\mathbb{R}^{n\times n}$. Let $(\mathbf{d}^k,\mathbf{x}^k)$ be the sequence generated from NIDS with $\alpha_i<2/L_i$ for all i and $\mathbf{I}\succcurlyeq c\Lambda^{1/2}(\mathbf{I}-\mathbf{W})\Lambda^{1/2}$. We define

$$\begin{aligned}\rho=\max\Bigg(&1-(2-\max_i(\alpha_iL_i))\min_i(\mu_i\alpha_i),\\&1-\frac{c}{\lambda_{\max}(\Lambda^{-1/2}(\mathbf{I}-\mathbf{W})^\dagger\Lambda^{-1/2})}\Bigg),\end{aligned} \tag{21}$$

and have

$$\begin{aligned}&\|\mathbf{x}^{k+1}-\mathbf{x}^*\|^2_{\Lambda^{-1}}+\|\mathbf{d}^{k+1}-\mathbf{d}^*\|^2_{\mathbf{M}+\Lambda}\\&\le\rho\left(\|\mathbf{x}^k-\mathbf{x}^*\|^2_{\Lambda^{-1}}+\|\mathbf{d}^k-\mathbf{d}^*\|^2_{\mathbf{M}+\Lambda}\right).\end{aligned} \tag{22}$$

Proof: From (15), we have

$$\begin{aligned}&\|\mathbf{x}^{k+1}-\mathbf{x}^*\|^2_{\Lambda^{-1}}+\|\mathbf{d}^{k+1}-\mathbf{d}^*\|^2_{\mathbf{M}}\\&\le\|\mathbf{x}^k-\mathbf{x}^*\|^2_{\Lambda^{-1}}+\|\mathbf{d}^k-\mathbf{d}^*\|^2_{\mathbf{M}}\\&\quad-\|\mathbf{x}^k-\mathbf{x}^{k+1}\|^2_{\Lambda^{-1}}-\|\mathbf{d}^k-\mathbf{d}^{k+1}\|^2_{\mathbf{M}}\\&\quad+2\langle\nabla s\left(\mathbf{x}^k\right)-\nabla s\left(\mathbf{x}^*\right),\mathbf{x}^k-\mathbf{x}^{k+1}\rangle\\&\quad-2\langle\mathbf{x}^k-\mathbf{x}^*,\nabla s(\mathbf{x}^k)-\nabla s(\mathbf{x}^*)\rangle.\end{aligned} \tag{23}$$

For the two inner product terms, we have

$$\begin{aligned}&2\langle\nabla s\left(\mathbf{x}^k\right)-\nabla s\left(\mathbf{x}^*\right),\mathbf{x}^k-\mathbf{x}^{k+1}\rangle\\&\quad-2\langle\mathbf{x}^k-\mathbf{x}^*,\nabla s(\mathbf{x}^k)-\nabla s(\mathbf{x}^*)\rangle\\&=-\|\mathbf{x}^k-\mathbf{x}^{k+1}-\Lambda\nabla s\left(\mathbf{x}^k\right)+\Lambda\nabla s\left(\mathbf{x}^*\right)\|^2_{\Lambda^{-1}}\\&\quad+\|\mathbf{x}^k-\mathbf{x}^{k+1}\|^2_{\Lambda^{-1}}+\|\nabla s\left(\mathbf{x}^k\right)-\nabla s\left(\mathbf{x}^*\right)\|^2_{\Lambda}\\&\quad-2\langle\mathbf{x}^k-\mathbf{x}^*,\nabla s(\mathbf{x}^k)-\nabla s(\mathbf{x}^*)\rangle\\&\le-\|\mathbf{d}^{k+1}-\mathbf{d}^*\|^2_{\Lambda}+\|\mathbf{x}^k-\mathbf{x}^{k+1}\|^2_{\Lambda^{-1}}\\&\quad+\|\nabla s\left(\mathbf{x}^k\right)-\nabla s\left(\mathbf{x}^*\right)\|^2_{\Lambda}\\&\quad-\max_i(\alpha_iL_i)\|\nabla s\left(\mathbf{x}^k\right)-\nabla s\left(\mathbf{x}^*\right)\|^2_{\mathbf{L}^{-1}}\\&\quad-(2-\max_i(\alpha_iL_i))\|\mathbf{x}^k-\mathbf{x}^*\|^2_{\mathbf{S}}\\&\le-\|\mathbf{d}^{k+1}-\mathbf{d}^*\|^2_{\Lambda}+\|\mathbf{x}^k-\mathbf{x}^{k+1}\|^2_{\Lambda^{-1}}\\&\quad-(2-\max_i(\alpha_iL_i))\min_i(\mu_i\alpha_i)\|\mathbf{x}^k-\mathbf{x}^*\|^2_{\Lambda^{-1}}.\end{aligned} \tag{24}$$

The first inequality comes from $\mathbf{x}^{k+1}=\mathbf{x}^k-\Lambda\nabla s\left(\mathbf{x}^k\right)-\mathbf{d}^{k+1}$, $\Lambda\nabla s\left(\mathbf{x}^*\right)+\mathbf{d}^*=\mathbf{0}$, (7), and (20). Combing (23)

and (24), we have

$$\begin{aligned}&\|\mathbf{x}^{k+1}-\mathbf{x}^*\|^2_{\Lambda^{-1}}+\|\mathbf{d}^{k+1}-\mathbf{d}^*\|^2_{\mathbf{M}}\\&\quad\le\|\mathbf{x}^{k}-\mathbf{x}^*\|^2_{\Lambda^{-1}}+\|\mathbf{d}^{k}-\mathbf{d}^*\|^2_{\mathbf{M}}-\|\mathbf{d}^{k+1}-\mathbf{d}^*\|^2_{\Lambda}\\&\qquad-(2-\max_i(\alpha_iL_i))\min_i(\mu_i\alpha_i)\|\mathbf{x}^{k}-\mathbf{x}^*\|^2_{\Lambda^{-1}}.\end{aligned}$$

Therefore,

$$\begin{aligned}&\|\mathbf{x}^{k+1}-\mathbf{x}^*\|^2_{\Lambda^{-1}}+\|\mathbf{d}^{k+1}-\mathbf{d}^*\|^2_{\mathbf{M}+\Lambda}\\&\quad\le(1-(2-\max_i(\alpha_iL_i))\min_i(\mu_i\alpha_i))\|\mathbf{x}^{k}-\mathbf{x}^*\|^2_{\Lambda^{-1}}\\&\qquad+\frac{\lambda_{\max}(\Lambda^{-1/2}\mathbf{M}\Lambda^{-1/2})}{\lambda_{\max}(\Lambda^{-1/2}\mathbf{M}\Lambda^{-1/2})+1}\|\mathbf{d}^{k}-\mathbf{d}^*\|^2_{\mathbf{M}+\Lambda}.\end{aligned}$$

Since

$$\begin{aligned}&\frac{\lambda_{\max}(\Lambda^{-1/2}\mathbf{M}\Lambda^{-1/2})}{\lambda_{\max}(\Lambda^{-1/2}\mathbf{M}\Lambda^{-1/2})+1}\\&\quad=1-\frac{c}{\lambda_{\max}(\Lambda^{-1/2}(\mathbf{I}-\mathbf{W})^{\dagger}\Lambda^{-1/2})}.\end{aligned}$$

Let ρ defined as (21), then we have (22).

Remark 4: The condition $\mathbf{I}\succcurlyeq c\Lambda^{1/2}(\mathbf{I}-\mathbf{W})\Lambda^{1/2}$ implies that $c\le\lambda_{n-1}(\Lambda^{-1/2}(\mathbf{I}-\mathbf{W})^{\dagger}\Lambda^{-1/2})$.

- If the agents across the whole network use an identical stepsize α, that is, $\Lambda=\alpha\mathbf{I}$, then

$$\rho=\max\left(1-(2-\alpha\max_i L_i)\alpha\min_i\mu_i,\ 1-\frac{c\alpha}{\lambda_{\max}((\mathbf{I}-\mathbf{W})^{\dagger})}\right).$$

  A concise but informative expression of the rate $\rho=\max\left(1-\frac{\min_i\mu_i}{\max_i L_i},\frac{\lambda_2(\mathbf{W})-\lambda_n(\mathbf{W})}{1-\lambda_n(\mathbf{W})}\right)$ can be obtained when we specifically choose $\alpha=\frac{1}{\max_i L_i}$ and $c=\frac{1}{(1-\lambda_n(\mathbf{W}))\alpha}$. When $\lambda_n(\mathbf{W})$ is not given, we choose $c=1/(2\alpha)$ and obtain the scalability $\max\left(\frac{\max_i L_i}{\min_i\mu_i},\frac{2}{1-\lambda_2(\mathbf{W})}\right)$. In this case, the network impact and the functional impact are decoupled.
- If we let $\Lambda=\mathbf{L}^{-1}$ and $c=\lambda_{n-1}(\mathbf{L}^{-1/2}(\mathbf{I}-\mathbf{W})^{\dagger}\mathbf{L}^{-1/2})$, then the rate becomes

$$\rho=\max\left(1-\min_i\frac{\mu_i}{L_i},\ 1-\frac{\lambda_{n-1}(\mathbf{L}^{1/2}(\mathbf{I}-\mathbf{W})^{\dagger}\mathbf{L}^{1/2})}{\lambda_{\max}(\mathbf{L}^{1/2}(\mathbf{I}-\mathbf{W})^{\dagger}\mathbf{L}^{1/2})}\right).$$

  When $\lambda_n(\mathbf{W})$ is not given, we choose $c=1/(2\max_i\alpha_i)=\min_i L_i/2$ and obtain the scalability $\max\left(\max_i\frac{L_i}{\mu_i},\frac{\max_i L_i}{\min_i L_i}\cdot\frac{2}{1-\lambda_2(\mathbf{W})}\right)$. In this case, the networking impact is coupled with the function factors, i.e., the smoothness heterogeneity $\frac{\max_i L_i}{\min_i L_i}$ is multiplied on the networking impact. While the other number depends on the functional condition numbers $\frac{L_i}{\mu_i}$'s only.

Remark 5: Theorem 2 separates the dependence of the linear convergence rate on the functions and the network structure. In our current scheme, all the agents perform information exchange and the proximal-gradient step once in each iteration. If the proximal-gradient step is expensive, this explicit rate formula can help us to decide whether the so-called multi-step consensus can help reducing the computational time.

For the sake of simplicity, let us assume for this moment that all the agents have the same strong convexity constant μ and gradient Lipschitz continuity constant L. Suppose that the "t-step consensus" technique is employed, i.e., the mixing matrix $\mathbf{W}$ in our algorithm is replaced by $\mathbf{W}^t$, where t is a positive integer. Then to reach ϵ-accuracy, the number of iterations needed is

$$O\left(\max\left(\frac{L}{\mu},\frac{1-\lambda_n(\mathbf{W}^t)}{1-\lambda_2(\mathbf{W}^t)}\right)\right)\log\frac{1}{\epsilon}.$$

When $L/\mu=1$ and step sizes are chosen as $\Lambda=\mathbf{L}^{-1}$, it says that we should let $t\to+\infty$ if the graph is not a complete graph. Such theoretical result is correct in intuition since in this case, the centralized gradient descent only needs one step to reach optimal and the bottleneck in decentralized optimization is the network.

Suppose $t_{\max}$ is a reasonable upper bound on t, which is set by the system designer. It is difficult to explicitly find an optimal t. But with the above analysis as an evidence, we suggest that one choose $t=\min\left([\log_{\lambda_2(\mathbf{W})}(1-\frac{\mu}{L})],t_{\max}\right)$ if $1-\frac{\mu}{L}>\lambda_2(\mathbf{W})$; otherwise $t=1$. Here $[\cdot]$ gives the nearest integer.

If the bottleneck is on the functions, we can introduce a mapping $x=By$ and change the unknown variable from x to y. E.g., if the function $s_i(x)$ is a composition of a convex function with a linear mapping, replacing x using y changes the linear mapping and the condition number L/μ of the function. When B is diagonal, it is similar to the column normalization in machine learning applications. There are other possible ways for reducing the condition number of the functions. It is out of the scope of this work, and we leave this as future work.

V. Numerical Experiments

In this section, we compare the performance of NIDS with several state-of-the-art algorithms for decentralized optimization. These methods are

- The EXTRA/PG-EXTRA (see (10));
- The DIGing-ATC [42]. For reference, the DIGing-ATC updates are provided as follows:

$$\begin{aligned}\mathbf{x}^{k+1}&=\mathbf{W}(\mathbf{x}^k-\alpha\mathbf{y}^k),\\\mathbf{y}^{k+1}&=\mathbf{W}(\mathbf{y}^k+\nabla f(\mathbf{x}^{k+1})-\nabla f(\mathbf{x}^k)).\end{aligned}$$

- The accelerated distributed Nesterov gradient descent (Acc-DNGD-SC in [52]);
- The (dual friendly) optimal algorithm (OA) for distributed optimization (equation (7) in [54]).

Note there are two rounds of communication in each iteration of DIGing-ATC and Acc-DNGD-SC while there is only one round in that of EXTRA/NIDS/OA. For all the experiments, we first compute the exact solution $\mathbf{x}^*$ for (1) using the centralized (proximal) gradient descent. All networks are randomly generated with connectivity ratio τ, where τ is defined as the number of actual edges divided by the total number of possible edges

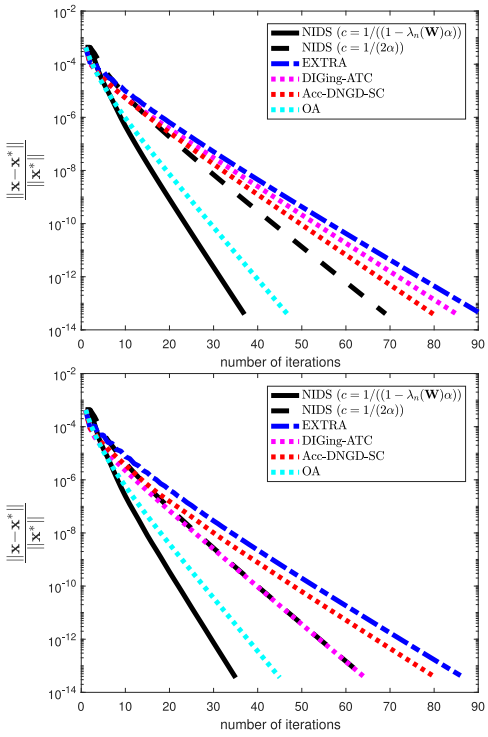


Fig. 1. Relative error $\frac{\|\mathbf{x}-\mathbf{x}^*\|}{\|\mathbf{x}^*\|}$ against the number of iterations for two different networks (top: $\tau = 0.35$; bottom: $\tau = 0.45$). NIDS, EXTRA, and DIGing-ATC use the same step-size $\alpha = 1/L$, where $L = \max_i L_i$. The step-size for Acc-DNGD-SC is hand-optimized. We use the default step-sizes for OA as suggested by the authors.

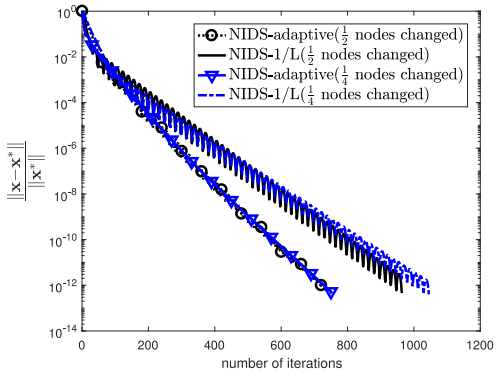


Fig. 2. The relative error $\frac{\|\mathbf{x}-\mathbf{x}^*\|}{\|\mathbf{x}^*\|}$ against the number of iterations. NIDS-1/L uses the same step-size $1/L$, where $L = \max_i L_i$, and NIDS-adaptive uses the step-size $1/L_i$ for each node. We assume that no graph information is available, thus $c = 1/(2\max_i \alpha_i)$. The connectivity ratio of the network is set as $\tau = 0.1$.

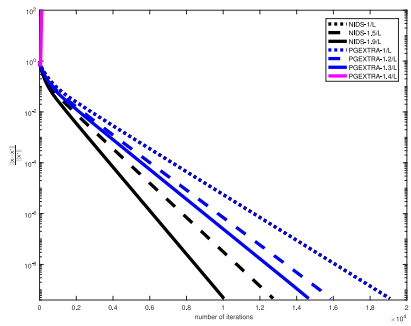


Fig. 3. The relative error $\frac{\|\mathbf{x}-\mathbf{x}^*\|}{\|\mathbf{x}^*\|}$ against the number of iterations. Different step-sizes for PG-EXTRA and NIDS are considered. For instance, "NIDS-$1/L$" is NIDS using the same step-size $1/L$ across the network of agents, where $L = \max_i L_i$. The connectivity ratio of the network is $\tau = 0.4$.

$\frac{n(n-1)}{2}$. We will report the specific τ used in each test. The mixing matrix $\mathbf{W}$ is always chosen with the Metropolis rule (see [57] and [36, Section 2.4]).

The experiments are carried in Matlab R2016b running on a laptop with Intel i7 CPU @ 2.60 HZ, 16.0 GB of RAM, and Windows 10 operating system. The source codes for reproducing the numerical results can be accessed at https://github.com/mingyan08/NIDS.

A. *The Strongly Convex Case With* $r(\mathbf{x}) = 0$

Consider the decentralized problem that solves for an unknown signal $x \in \mathbb{R}^p$. Each agent $i \in \{1, \ldots, n\}$ takes its own measurement via $y_i = \mathbf{M}_i x + e_i$, where $y_i \in \mathbb{R}^{m_i}$ is the measurement vector, $\mathbf{M}_i \in \mathbb{R}^{m_i \times p}$ is the sensing matrix, and $e_i \in \mathbb{R}^{m_i}$ is the independent and identically distributed noise. To estimate x collaboratively, we apply the decentralized algorithms to solve

$$\underset{x}{\text{minimize}} \ \frac{1}{n}\sum_{i=1}^{n} \frac{1}{2}\|\mathbf{M}_i x - y_i\|^2$$

In order to ensure that each function $\frac{1}{2}\|\mathbf{M}_i x - y_i\|^2$ is strongly convex, we choose $m_i = 60$ and $p = 50$ and set the number of nodes $n = 40$. For the first experiment, we choose $\mathbf{M}_i$ such that the Lipshchitz constant of ∇s_i satisfies $L_i = 1$ and the strongly convex constant $\mu_i = 0.5$ for all i. Based on Remark 4, we choose $\alpha = 1/(\max_i L_i) = 1$ and $c = 1/(1 - \lambda_n(\mathbf{W}))$ for NIDS. In addition, we choose $c = 1/2$ such that $\widetilde{\mathbf{W}} = \frac{\mathbf{I}+\mathbf{W}}{2}$, which gives the same as that for EXTRA.

The comparison of these methods (NIDS with $c = 1/((1 - \lambda_n(\mathbf{W}))\alpha)$, NIDS with $c = 1/2$, EXTRA, DIGing-ATC, Acc-DNGD-SC, and OA) is shown in Fig. 1 for two different networks with connectivity ratios $\tau = 0.35$ (top) and $\tau = 0.45$ (bottom), respectively. It shows better performance of NIDS in both choices of c (corresponding to known $\mathbf{W}$ and unknown $\mathbf{W}$) than that of other algorithms. NIDS with $c = 1/((1 - \lambda_n(\mathbf{W}))\alpha)$ always takes less than half the number of iterations used by EXTRA to reach the same accuracy. In our experiment, DIGing-ATC appears to be sensitive to networks. Under a better connected network (see Fig. 1 bottom), DIGing-ATC can catch up with NIDS with $c = 1/(2\alpha)$. The theoretical step-size of Acc-DNGD-SC is too small due to a very small constant in the bound in [52], and the convergence of Acc-DNGD-DC under such theoretical step-size in our test is slow and uncompetitive. Thus we have carefully tuned its step-size. With the hand-optimized step-size, Acc-DNGD-SC can achieve a comparable performance as NIDS with $c = 1/(2\alpha)$. In the plots, we observe that OA is fast in terms of the number of iterations. However, in this case, the per-iteration cost of OA is relatively high since it requires solving a system of linear

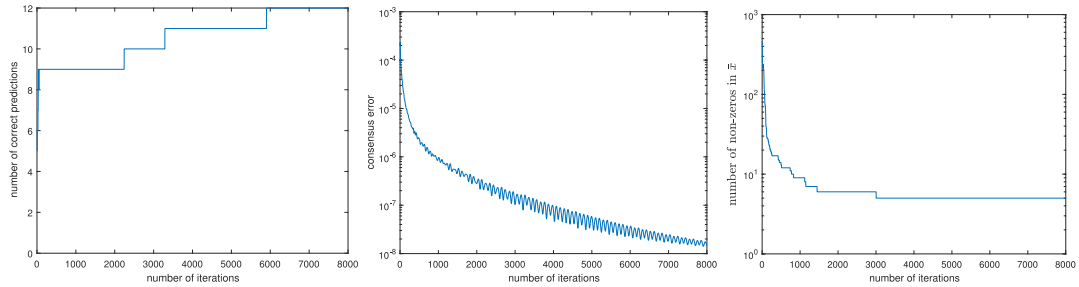


Fig. 4. Performance of NIDS for sparse logistic regression. Left: number of correct predictions vs. iteration. Middle: consensus error $\left\|\mathbf{x}^k\right\|_{I-\mathbf{W}}$ vs. iteration; Right: number of non-zero elements in $\bar{x}^k = \frac{1}{50}\sum_{i=1}^{50} x_i^k$ vs. iteration.

equations at each iteration (though factorization tricks may be used to save some computational time).

Next, we demonstrate the effort of uncoordinated/adaptive step-size. We construct the function with $\mu_i = 0.02$ and $L_i = 1$ for each node i. Then we change the L_i values by multiplying the function by a constant. We use the same mixing matrix for the following two experiments.

- We change half nodes. We randomly pick an even number node and multiply its function by 4. For remaining even number nodes, we multiply their functions by a random integer 2 or 3.
- We change a quarter nodes. We randomly pick an node not in $\mathcal{U} = \{4, 8, 16, \ldots, 40\}$ and multiply its function by 10. Then for other nodes in $\mathcal{U}$, we multiply their functions by a random integer between 2 and 9.

We compare NIDS with adaptive step-size ($1/L_i$ for node i) and NIDS with same step-size $1/\max_i L_i$ in Fig. 2. We let $c = 1/(2\max_i \alpha_i)$, so no network information is needed. As shown in Fig. 2, NIDS with adaptive step-size converges faster than same step-size.

B. The Case With Nonsmooth Function $r(\mathbf{x})$

In this subsection, we compare the performance of NIDS with PG-EXTRA [1] only since other methods in Section V-A, such as DIGing, can not be applied to this nonsmooth case. We consider a decentralized compressed sensing problem. Again, each agent $i \in \{1, \ldots, n\}$ takes its own measurement via $y_i = \mathbf{M}_i x + e_i$, where $y_i \in \mathbb{R}^{m_i}$ is the measurement vector, $\mathbf{M}_i \in \mathbb{R}^{m_i \times p}$ is the sensing matrix, and $e_i \in \mathbb{R}^{m_i}$ is the independent and identically distributed noise. Here, x is a sparse signal. The optimization problem is

$$\underset{x}{\text{minimize}}\ \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}\|\mathbf{M}_i x - y_i\|^2 + \frac{1}{n}\sum_{i=1}^{n}\lambda_i\|x\|_1,$$

where the connectivity ratio of the network $\tau = 0.1$. We normalize the problem to make sure that the Lipschitz constant satisfies $L_i = 1$ for each node, we choose $m_i = 3$ and $p = 200$ and set the number of nodes $n = 40$.

Fig. 3 shows that a larger step-size in NIDS leads to faster convergence. With step-size 1, NIDS and PG-EXTRA converge at the same speed. But if we keep increasing the step-size, PG-EXTRA will diverge with step-size 1.4 while the step-size of NIDS can be increased to 1.9 maintaining convergence at a faster speed.

C. An Application in Classification for Healthcare Data

We consider a decentralized sparse logistic regression problem to classify the colon-cancer data [58]. There are 62 samples, and each sample features 2,000 pieces of gene expressing information (numericalized and normalized [58]) and a binary outcome. The outcome can be normal/negative ($+1$) or tumor/positive (-1) and the data set contains 22 normal and 40 tumor colon tissue samples. We store the gene information in $\mathbf{M}_i \in \mathbb{R}^{1\times 2001}$ (one more dimension is augmented to take care of the linear offset/constant in the logit function model), and the outcome information is $y_i \in \{-1, 1\}$, $i \in \mathcal{S}_1 \bigcup \mathcal{S}_2$, where $\mathcal{S}_1$ serves for training while $\mathcal{S}_2$ serves for testing. Suppose we have a 50-node connected network where each node i holds 1 sample $(\mathbf{M}_i, y_i)$ (the connected network is randomly generated and its connectivity ratio is set to 0.08; the 50 in-network samples indexed by $\mathcal{S}_1$ are randomly drawn from the 62 samples). The decentralized logistic regression

$$\underset{x}{\text{minimize}}\ \frac{1}{|\mathcal{S}_1|}\sum_{i\in\mathcal{S}_1}\ln(1+\exp(-\mathbf{M}_i x_i y_i)) + \frac{1}{|\mathcal{S}_1|}\sum_{i\in\mathcal{S}_1}\hat{\lambda}_i\|x_i\|_2^2 + \frac{1}{|\mathcal{S}_1|}\sum_{i\in\mathcal{S}_1}\lambda_i\|x_i\|_1,$$

is then solved over the network to train a sparse linear classifier $\mathbf{x}^*$ for the outcome prediction of the remaining/future samples/data. In the optimization formulation, the ℓ_2 norm term imposes strong convexity to s, while ℓ_1 term promotes sparsity of the solution. Aside from the 50 samples used for training purpose, we randomly select 12 nodes from the 50 nodes to show the prediction performance of the remaining 12 samples in Fig. 4 left. The middle and right figures in Fig. 4 show how the consensus error $\left\|\mathbf{x}^k\right\|_{I-\mathbf{W}}$ and the sparsity of the average solution vector $\frac{1}{50}\sum_{i=1}^{50} x_i^k$ drops, respectively.

VI. Conclusion

We proposed a novel decentralized consensus algorithm NIDS, whose step-size does not depend on the network structure. In NIDS, the step-size depends *only* on the objective function, and it can be as large as $2/L$, where L is the Lipschitz constant of the gradient of the smooth function. We showed that NIDS

converges at the $o(1/k)$ rate for the general convex case and at a linear rate for the strongly convex case. For the strongly convex case, we separated the linear convergence rate's dependence on the objective function and the network. The separated convergence rates match the typical rates for the general gradient descent and the consensus averaging. Furthermore, every agent in the network can choose its own step-size independently by its own objective function. Numerical experiments validated the theoretical results and demonstrated better performance of NIDS over state-of-the-art algorithms. Because the step-size of NIDS does not depend on the network structure, there are many possible future extensions. One extension is to apply NIDS on dynamic networks where nodes can join and drop off.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for helpful comments and suggestions to improve the clarity of this paper.

REFERENCES

[1] W. Shi, Q. Ling, G. Wu, and W. Yin, "A proximal gradient algorithm for decentralized composite optimization," *IEEE Trans. Signal Process.*, vol. 63, no. 22, pp. 6013–6023, Nov. 2015.

[2] L. Xiao, S. Boyd, and S. Kim, "Distributed average consensus with least-mean-square deviation," *J. Parallel Distrib. Comput.*, vol. 67, no. 1, pp. 33–46, 2007.

[3] K. Cai and H. Ishii, "Average consensus on arbitrary strongly connected digraphs with time-varying topologies," *IEEE Trans. Autom. Control*, vol. 59, no. 4, pp. 1066–1071, Apr. 2014.

[4] A. Olshevsky, "Linear time average consensus and distributed optimization on fixed graphs," *SIAM J. Control Optim.*, vol. 55, no. 6, pp. 3990–4014, 2017.

[5] J. Bazerque and G. Giannakis, "Distributed spectrum sensing for cognitive radio networks by exploiting sparsity," *IEEE Trans. Signal Process.*, vol. 58, no. 3, pp. 1847–1862, Mar. 2010.

[6] W. Ren, "Consensus based formation control strategies for multi-vehicle systems," in *Proc. Amer. Control Conf.*, 2006, pp. 4237–4242.

[7] A. Olshevsky, "efficient information aggregation strategies for distributed control and signal processing," Ph.D. dissertation, Massachusetts Inst. Technol., Cambridge, MA, USA, 2010.

[8] S. Ram, V. Veeravalli, and A. Nedić, "Distributed non-autonomous power control through distributed convex optimization," in *Proc. INFOCOM*, 2009, pp. 3001–3005.

[9] L. Gan, U. Topcu, and S. Low, "Optimal decentralized protocol for electric vehicle charging," *IEEE Trans. Power Syst.*, vol. 28, no. 2, pp. 940–951, May 2013.

[10] M. Rabbat and R. Nowak, "Distributed optimization in sensor networks," in *Proc. 3rd Int. Symp. Inf. Process. Sensor Netw.*, 2004, pp. 20–27.

[11] P. Forero, A. Cano, and G. Giannakis, "Consensus-based distributed support vector machines," *J. Mach. Learn. Res.*, vol. 59, pp. 1663–1707, 2010.

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

[13] D. Bertsekas, "Distributed asynchronous computation of fixed points," *Math. Program.*, vol. 27, no. 1, pp. 107–120, 1983.

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

[15] 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, 2011.

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

[17] A. Nedić and D. Bertsekas, "Convergence rate of incremental subgradient algorithms," in *Stochastic Optimization: Algorithms and Applications*. New York, NY, USA: Springer, 2001, pp. 223–264.

[18] A. Nedić and D. Bertsekas, "Incremental subgradient methods for nondifferentiable optimization," *SIAM J. Optim.*, vol. 12, no. 1, pp. 109–138, 2001.

[19] A. Nedich, D. P. Bertsekas, and V. S. Borkar, "Distributed asynchronous incremental subgradient methods," *Studies Comput. Math.*, vol. 8, pp. 381–407, 2001.

[20] S. Ram, A. Nedić, and V. Veeravalli, "Incremental stochastic subgradient algorithms for convex optimization," *SIAM J. Optim.*, vol. 20, no. 2, pp. 691–717, 2009.

[21] D. P. Bertsekas, "Incremental proximal methods for large scale convex optimization," *Math. Program.*, vol. 129, pp. 163–195, 2011.

[22] M. Wang and D. P. Bertsekas, "Incremental constraint projection-proximal methods for nonsmooth convex optimization," Lab. Information and Decision Systems Rep. LIDS-P-2907, MIT, Cambridge, MA, USA, Jul. 2013.

[23] 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.

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

[25] A. Nedić, "Asynchronous broadcast-based convex optimization over a network," *IEEE Trans. Autom. Control*, vol. 56, no. 6, pp. 1337–1351, Jun. 2011.

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

[27] H. Terelius, U. Topcu, and R. Murray, "Decentralized multi-agent optimization via dual decomposition," *IFAC Proc. Vol.*, vol. 44, no. 1, pp. 11 245–11 251, 2011.

[28] D. P. Bertsekas and J. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*, 2nd ed. Nashua, NH, USA: Athena Scientific, 1997.

[29] E. Wei and A. Ozdaglar, "On the $O(1/k)$ convergence of asynchronous distributed alternating direction method of multipliers," in *Proc. IEEE Global Conf. Signal Inf. Process.*, 2013, pp. 551–554.

[30] T.-H. Chang, M. Hong, and X. Wang, "Multi-agent distributed optimization via inexact consensus ADMM," *IEEE Trans. Signal Process.*, vol. 63, no. 2, pp. 482–497, Jan. 2015.

[31] M. Hong and T.-H. Chang, "Stochastic proximal gradient consensus over random networks," *IEEE Trans. Signal Process.*, vol. 65, no. 11, pp. 2933–2948, Jun. 2017.

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

[33] A. Chen and A. Ozdaglar, "A fast distributed proximal-gradient method," in *Proc. 50th Annu. Allerton Conf. Commun., Control, Comput.*, 2012, pp. 601–608.

[34] D. Jakovetic, J. Xavier, and J. Moura, "Fast distributed gradient methods," *IEEE Trans. Autom. Control*, vol. 59, no. 5, pp. 1131–1146, May 2014.

[35] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*. Berlin, Germany: Springer Science & Business Media, 2013, vol. 87.

[36] 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.

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

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

[39] P. Di Lorenzo and G. Scutari, "NEXT: In-network nonconvex optimization," *IEEE Trans. Signal Inf. Process. Over Netw.*, vol. 2, no. 2, pp. 120–136, Jun. 2016.

[40] G. Qu and N. Li, "Harnessing smoothness to accelerate distributed optimization," in *Proc. IEEE 55th Conf. Decis. Control*, 2016, pp. 159–166.

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

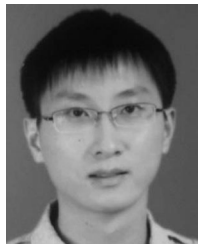
[42] A. Nedić, A. Olshevsky, W. Shi, and C. A. Uribe, "Geometrically convergent distributed optimization with uncoordinated step-sizes," in *Proc. Amer. Control Conf.*, 2017, pp. 3950–3955.

[43] A. Nedić and A. Olshevsky, "Distributed optimization over time-varying directed graphs," in *Proc. 52nd IEEE Annu. Conf. Decis. Control*, 2013, pp. 6855–6860.

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

[45] J. Zeng and W. Yin, "ExtraPush for convex smooth decentralized optimization over directed networks," *J. Comput. Math.*, vol. 35, no. 4, pp. 381–394, 2017.
[46] Y. Sun, G. Scutari, and D. Palomar, "Distributed nonconvex multiagent optimization over time-varying networks," in *Proc. 50th Asilomar Conf. Signals, Syst. Comput.*, 2016, pp. 788–794.
[47] Z. Li and M. Yan, "A primal-dual algorithm with optimal stepsizes and its application in decentralized consensus optimization," 2017, arXiv:1711.06785.
[48] S. A. Alghunaim and A. H. Sayed, "Linear convergence of primal-dual gradient methods and their performance in distributed optimization," Apr. 2019, arXiv:1904.01196.
[49] K. Yuan, B. Ying, X. Zhao, and A. H. Sayed, "Exact diffusion for distributed optimization and learning Part I: Algorithm development," *IEEE Trans. Signal Process.*, vol. 67, no. 3, pp. 708–723, Feb. 2019.
[50] K. Yuan, B. Ying, X. Zhao, and A. H. Sayed, "Exact diffusion for distributed optimization and learning Part II: Convergence analysis," *IEEE Trans. Signal Process.*, vol. 67, no. 3, pp. 724–739, Feb. 2019.
[51] A. Nedic, A. Olshevsky, A. Ozdaglar, and J. N. Tsitsiklis, "On distributed averaging algorithms and quantization effects," *IEEE Trans. Autom. Control*, vol. 54, no. 11, pp. 2506–2517, Nov. 2009.
[52] G. Qu and N. Li, "Accelerated distributed Nesterov gradient descent for convex and smooth functions," in *Proc. IEEE 56th Annu. Conf. Decis. Control*, 2017, pp. 2260–2267.
[53] K. Scaman, F. Bach, S. Bubeck, Y. T. Lee, and L. Massoulié, "Optimal algorithms for smooth and strongly convex distributed optimization in networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 3027–3036.
[54] C. A. Uribe, S. Lee, A. Gasnikov, and A. Nedić, "Optimal algorithms for distributed optimization," 2017, arXiv:1712.00232.
[55] T. Wu, K. Yuan, Q. Ling, W. Yin, and A. H. Sayed, "Decentralized consensus optimization with asynchrony and delays," *IEEE Trans. Signal Inf. Process. Over Netw.*, vol. 4, no. 2, pp. 293–307, Jun. 2018.
[56] M. Yan, "A new primal-dual algorithm for minimizing the sum of three functions with a linear operator," *J. Sci. Comput.*, vol. 76, pp. 1698–1717, 2018.
[57] S. Boyd, P. Diaconis, and L. Xiao, "Fastest mixing Markov chain on a graph," *SIAM Rev.*, vol. 46, no. 4, pp. 667–689, 2004.
[58] J. Liu, J. Chen, and J. Ye, "Large-scale sparse logistic regression," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2009, pp. 547–556.

Zhi Li received the B.S. and M.S. degrees in applied mathematics from China University of Petroleum, Shandong, China, in 2007 and 2010, respectively. He received the M.S. degree in applied science from Saint Marys University, Halifax, NS, Canada, in 2012. After being awarded the Hong Kong Ph.D. Fellowship, he went to Hong Kong Baptist University, Hong Kong, where he received the Ph.D. degree in applied mathematics in 2016. Since 2016, he has been a Postdoctoral Researcher with the Department of Computational Mathematics, Science and Engineering, Michigan State University, East Lansing, MI, USA.



Wei Shi received the B.E. degree in automation and the Ph.D. degree in control science and engineering from the University of Science and Technology of China, Hefei, China, in 2010 and 2015, respectively. He was a Postdoctoral with the University of Illinois at Urbana-Champaign, Arizona State University, and Princeton University since 2015. His research interests spanned in optimization, cyberphysical systems, and big data analytics. He received the 2017 Young Author Best Paper Award from the IEEE Signal Processing Society.

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