Distributed Linearized ADMM for Network Cost Minimization

Xuanyu Cao and K. J. Ray Liu, *Fellow, IEEE*

***Abstract*—In this paper, we study a generic network cost minimization problem, in which every node has a local decision vector to determine. Each node incurs a cost depending on its decision vector and each link also incurs a cost depending on the decision vectors of its two end nodes. All nodes cooperate to minimize the overall network cost. The formulated network cost minimization problem has broad applications in distributed signal processing and control over multiagent systems. To obtain a decentralized algorithm for the formulated problem, we resort to the distributed alternating direction method of multipliers (DADMM). However, each iteration of the DADMM involves solving a local optimization problem at each node, leading to intractable computational burden in many circumstances. As such, inspired by recent works on approximated ADMM for consensus optimization problem, we propose a distributed linearized ADMM (DLADMM) algorithm for network cost minimization. In the DLADMM, each iteration only involves closed-form computations and avoids local optimization problems, which greatly reduces the computational complexity compared to the DADMM. We prove that the DLADMM converges to an optimal point when the local cost functions are convex and have Lipschitz continuous gradients. Linear convergence rate of the DLADMM is also established if the local cost functions are further strongly convex. Numerical experiments are conducted to corroborate the effectiveness of the DLADMM and we observe that the DLADMM has similar convergence performance as DADMM does while the former enjoys much lower computational overhead. The impact of network topology, connectivity, and algorithm parameters are also investigated through simulations.**

***Index Terms*—Decentralized optimization, network optimization, alternating direction method of multipliers.**

I. Introduction

THE last decade has witnessed the advances of decentralized signal processing and control over networked multi-agent systems, which result in great research interest in distributed optimizations over networks. Such distributed optimization problems arise in fields such as adaptive signal processing over networks [1], distributed estimation over sensor networks [2], [3], decentralized power system state estimation and management [4], [5] as well as signal processing for communication networks [6], [7]. In these applications, data are distributed over individual nodes across the network. Centralized data processing and optimization suffer from high or even prohibitive communication overload and are vulnerable to link failures and network congestions. As such, optimizing and processing data in a decentralized manner, where only local information exchange among neighbors is allowed, are more favorable.

In the literature, distributed optimization has been extensively studied recently. Two important categories of distributed optimization problems are distributed network utility maximization (NUM) and consensus optimization. In distributed NUM, each agent has a local decision variable, based on which it obtains some utility. Agents cooperatively maximize the total utilities of the network subject to some coupling resource constraints such as the link capacity constraint in communication networks. For NUM, Wei *et al.* propose and analyze a distributed Newton method in [8], [9], while the effect of noisy information exchange is studied in [10]. Moreover, Niu and Li present an asynchronous decentralized algorithm with elegant pricing interpretations for NUM [11]. On the other hand, in consensus optimization, all agents share the same decision variable but have different local cost functions and the goal is to cooperatively minimize the total cost of the network. Nedic and Ozdaglar propose a decentralized subgradient method for consensus optimization in [12] while a dual averaging method is presented in [13]. Specific forms of consensus problems such as adaptive signal processing over networks [1] and average consensus (where agents cooperate to compute the average of individuals' data) [14] have been studied by using the alternating direction method of multipliers (ADMM). More recently, the general form of consensus problem is investigated by using the distributed Nesterov gradient algorithm in [15] and the distributed ADMM (DADMM) in [16]. Later, several variants of DADMM are proposed for the consensus problems, including linearized ADMM [1], quadratically approximated ADMM [18], inexact ADMM [19], weighted ADMM [20], asynchronous ADMM [21], [22], and dynamic ADMM [23]. Recently, the convergence properties of ADMM for non-convex optimization are studied by Hong *et al.* in [24]. In addition, second order methods, i.e., Newton's method and its variants, are examined for consensus optimization in [25], [26].

In all the aforementioned works, only costs or utilities at individual nodes are taken into consideration while the costs or gains of links are ignored. For example, for consensus optimization, the network cost is only composed of local cost at each node and

Manuscript received August 11, 2017; revised November 29, 2017 and January 22, 2018; accepted February 12, 2018. Date of publication February 15, 2018; date of current version August 7, 2018. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Amir Leshem. *(Corresponding author: Xuanyu Cao.)*

X. Cao is with the Department of Electrical Engineering, Princeton University, Princeton, NJ 08544 USA (e-mail: x.cao@princeton.edu).

K. J. Ray Liu is with the Department of Electrical and Computer Engineering, University of Maryland, College Park, MD 20742 USA (e-mail: kjrliu@umd.edu).

Digital Object Identifier 10.1109/TSIPN.2018.2806841

the effect of the link is not incorporated. In fact, for consensus problems, though the decentralized algorithms may depend on the network topology (the links connecting nodes), the problem formulation itself is independent of the network structure. This is not suitable for many applications in distributed signal processing and control, where the notion of link cost or link utility naturally arises. For example, in multitask adaptive learning [27], each node i aims at estimating its weight vector $\mathbf{w}_i$, which, in contrast to the consensus problems, is different from other nodes' weight vectors. In most networks, neighbor nodes tend to have similar weight vectors. To incorporate this prior knowledge into the estimator, the objective function to be minimized should include terms promoting similarity between neighbors such as $\|\mathbf{w}_i - \mathbf{w}_j\|_2^2$, where i, j are neighbors. This term is tantamount to a link cost of the link (i, j).

In this paper, we study the network cost minimization problem, where the network cost encompasses both node costs and link costs. The consensus optimization problems studied in [17], [18], [14], [23] are special cases of the network cost minimization problem examined in this paper. In fact, by using the link costs to enforce proximity between neighbors and letting the weight of link costs go to infinity, we can recover the consensus constraints of the network. To obtain a distributed algorithm for the network cost minimization problem, we resort to the distributed alternating direction method of multipliers (DADMM) [28], a primal-dual optimization algorithm which generally converges faster than primal domain alternatives such as the distributed subgradient method [12]. However, each iteration of the DADMM algorithm involves solving a local optimization problem at each node, which is a major computational burden. To avoid this, inspired by the recent works [17], [18] on approximated ADMM for concensus optimization, we propose a distributed linearized ADMM (DLADMM) algorithm for network cost minimization. The DLADMM algorithm replaces the local optimization problem with closed form computations through linearizations and thus greatly reduce the computational complexity compared to DADMM. We further theoretically demonstrate that the DLADMM algorithm has appealing convergence properties. We note that an analogous DLADMM algorithm has been proposed in [17] for consensus optimization problem. However, the decentralized algorithm and the convergence analysis depend on the specific structure of consensus optimization problem. In this work, we develop and analyze a DLADMM algorithm suitable for the generic network cost minimization problem, which encompasses consensus optimization as a special case. Our contributions can be summarized as follows.

- We formulate a generic form of network cost minimization problem incorporating both node costs and link costs. The formulated problem has broad applications in distributed signal processing and control in networked systems.
- A distributed linearized ADMM algorithm for the network cost minimization problem is presented. The DLADMM algorithm operates in a decentralized manner and each iteration only consists of simple closed form computations, which endows the DLADMM with much lower computational overhead than the DADMM algorithm.
- We prove that the DLADMM algorithm converges to an optimal point if the local cost functions are convex and have Lipschitz continuous gradients. Linear convergence rate of the DLADMM algorithm is also established provided that the local cost functions are further strongly convex.
- Numerical experiments are conducted to validate the performance of the DLADMM algorithm. We empirically observe that the DLADMM algorithm has similar convergence speed as the DADMM algorithm does while the former enjoys much lower computational complexity. The impact of network topology, connectivity and algorithm parameters is also investigated.

The organization of the rest of this paper is as follows. In Section II, the network cost minimization problem is formally formulated and the DLADMM, DADMM algorithms are developed. In Section III, the convergence properties of the DLADMM algorithm are analyzed. In Section IV, numerical simulations are conducted. In Section V, we conclude this work.

II. Problem Statement and Algorithm Development

In this section, we first motivate and formulate the network cost minimization problem. Then, we present a brief review of the basics of the ADMM, following which a distributed ADMM (DADMM) algorithm for the network cost minimization problem is shown. Finally, to reduce the computational burden of the DADMM, we propose a distributed linearized ADMM (DLADMM) algorithm for the network cost minimization problem.

A. *The Statement of the Problem*

Consider a network of n nodes and some links between these nodes. We assume that the network is a simple graph, i.e., the network is undirected with no self-loop and there is at most one edge between any pair of nodes. Denote the number of links as m, in which (i, j) and (j, i) are counted as two links for ease of later exposition. Denote the set of neighbors of node i (those who are linked with node i) as Ω_i. The network can be either connected or disconnected (there does not necessarily exist a path connecting every pair of nodes). Suppose each node i has a p-dimensional local decision variable $\mathbf{x}_i \in \mathbb{R}^p$. Given $\mathbf{x}_i$, the cost at node i is $f_i(\mathbf{x}_i)$, where f_i is called the node cost function at node i. Moreover, given two connected nodes i and j and their decision variables $\mathbf{x}_i$ and $\mathbf{x}_j$, there is a cost of $g_{ij}(\mathbf{x}_i, \mathbf{x}_j)$ associated with the link (i, j), where g_{ij} is called the link cost function of the link (i, j). The goal of the network is to solve the following network cost minimization problem in a decentralized manner:

$$\text{Minimize} \sum_{i=1}^{n} f_i(\mathbf{x}_i) + \sum_{i=1}^{n} \sum_{j \in \Omega_i} g_{ij}(\mathbf{x}_i, \mathbf{x}_j). \tag{1}$$

We remark that the consensus optimization problems in [14], [17], [18], [23] are special cases of the network cost minimization problem (1) here. Actually, by setting the link costs $g_{ij}(\mathbf{x}_i, \mathbf{x}_j)$ to be the weighted distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ and letting the weights of link costs go to infinity, we recover the consensus constraints provided that the network is connected. The problem formulation (1) has broad applications, among which we name three in the following.

- In distributed estimation over (sensor) networks, each node i has a local unknown vector $\mathbf{x}_i$ to be estimated. The cost at node i, i.e., $f_i(\mathbf{x}_i)$ may be some squared error or the negative log-likelihood (the former can be regarded as a special case of the latter when the noise is Gaussian) with respect to the local data observed by node i. The link cost $g_{ij}(\mathbf{x}_i, \mathbf{x}_j)$ for a link (i, j) can be used to enforce similarity between neighbor nodes, e.g., $\|\mathbf{x}_i - \mathbf{x}_j\|_2^2$ in multitask adaptive networks in [27], [29].
- For resource allocation over networks, $\mathbf{x}_i$ corresponds to some resources at node i and the node cost $f_i(\mathbf{x}_i)$ is the negative of node i's utility. The link cost $g_{ij}(\mathbf{x}_i, \mathbf{x}_j)$ for a link (i, j) may represent the negative effect of the consumption of the resources $\mathbf{x}_i$ and $\mathbf{x}_j$. For instance, in wireless networks, $\mathbf{x}_i$ may be the transmission power of node i and two nodes are linked if they are within the wireless interference range. In such a case, the link cost $g_{ij}(\mathbf{x}_i, \mathbf{x}_j)$ for a link (i, j) can be used to quantify the cost incurred by mutual interference in wireless communications.
- For an image, each x_i is the value of the i-th pixel and two pixels (or nodes) are linked if they are adjacent. In the image denoising problem, one wants to minimize the total variations of the pixels (as noises are often irregular values making the pixels abnormally different from their neighbor pixels) while remaining faithful to the given noisy image. The node cost $f_i(x_i)$ can be used to quantify the deviation of x_i from the given noisy pixel $\widetilde{x}_i$ and the link cost $g_{ij}(x_i, x_j)$ can represent the difference between the two neighbor pixels i and j.

For ease of reference, we define the following assumptions, some of which will be adopted in later theorems.

Assumption 1: All the node cost functions f_i's and the link cost functions g_{ij}'s are convex.

Assumption 2: All the node cost functions f_i's and the link cost functions g_{ij}'s have Lipschitz continuous gradients with constant $L > 0$, i.e., (a) $\forall i, \mathbf{x}_i, \mathbf{x}_i' \in \mathbb{R}^p$:

$$\|\nabla f_i(\mathbf{x}_i) - \nabla f_i(\mathbf{x}_i')\|_2 \leq L\|\mathbf{x}_i - \mathbf{x}_i'\|_2; \tag{2}$$

(b) $\forall i, j \in \Omega_i, \mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_i', \mathbf{x}_j' \in \mathbb{R}^p$:

$$\|\nabla g_{ij}(\mathbf{x}_i, \mathbf{x}_j) - \nabla g_{ij}(\mathbf{x}_i', \mathbf{x}_j')\|_2 \leq L \left\| \begin{bmatrix} \mathbf{x}_i \\ \mathbf{x}_j \end{bmatrix} - \begin{bmatrix} \mathbf{x}_i' \\ \mathbf{x}_j' \end{bmatrix} \right\|_2. \tag{3}$$

Assumption 3: All the node cost functions f_i's and the link cost functions g_{ij}'s are strongly convex with constant $\tau > 0$, i.e., (a) For any $i = 1, ..., n$:

$$(\nabla f_i(\mathbf{x}_i) - \nabla f_i(\mathbf{x}_i'))^\mathsf{T}(\mathbf{x}_i - \mathbf{x}_i') \geq \tau\|\mathbf{x}_i - \mathbf{x}_i'\|_2^2, \forall \mathbf{x}_i, \mathbf{x}_i' \in \mathbb{R}^p; \tag{4}$$

(b) For any $i, j \in \Omega_i$:

$$\begin{aligned} &\left(\begin{bmatrix} \nabla_{\mathbf{x}_i} g_{ij}(\mathbf{x}_i, \mathbf{x}_j) \\ \nabla_{\mathbf{x}_j} g_{ij}(\mathbf{x}_i, \mathbf{x}_j) \end{bmatrix} - \begin{bmatrix} \nabla_{\mathbf{x}_i'} g_{ij}(\mathbf{x}_i', \mathbf{x}_j') \\ \nabla_{\mathbf{x}_j'} g_{ij}(\mathbf{x}_i', \mathbf{x}_j') \end{bmatrix} \right)^\mathsf{T} \\ &\cdot \left(\begin{bmatrix} \mathbf{x}_i \\ \mathbf{x}_j \end{bmatrix} - \begin{bmatrix} \mathbf{x}_i' \\ \mathbf{x}_j' \end{bmatrix} \right) \\ &\geq \tau \left\| \begin{bmatrix} \mathbf{x}_i \\ \mathbf{x}_j \end{bmatrix} - \begin{bmatrix} \mathbf{x}_i' \\ \mathbf{x}_j' \end{bmatrix} \right\|_2^2, \ \forall \mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_i', \mathbf{x}_j' \in \mathbb{R}^p. \end{aligned} \tag{5}$$

Remark 1: We note the following facts. When f_i is twice differentiable, the condition (4) of Assumption 3 is equivalent to $\nabla^2 f_i(\mathbf{x}_i) \succeq \tau \mathbf{I}_p, \forall \mathbf{x}_i$. Similarly, when g_{ij} is twice differentiable, the condition (5) of Assumption 3 is equivalent to $\nabla^2 g_{ij}(\mathbf{x}_i, \mathbf{x}_j) \succeq \tau \mathbf{I}_{2p}, \forall \mathbf{x}_i, \mathbf{x}_j$. This second order definition of strong convexity is more intuitively acceptable and has been used in the analysis of convex optimization algorithms in the literature [30]. But it requires twice differentiability and is not directly useful in the analysis in this work.

Remark 2: All three assumptions are standard in the literature of numerical optimization when analyzing the performance of optimization algorithms [16], [30], [31].

B. Preliminaries of ADMM

ADMM is an optimization framework widely applied to various signal processing applications, including wireless communications [6], power systems [32] and multi-agent coordination [33]. It enjoys fast convergence speed under mild technical conditions [31] and is especially suitable for the development of distributed algorithms [28], [34]. ADMM solves problems of the following form:

$$\text{Minimize}_{\mathbf{x},\mathbf{z}} f(\mathbf{x}) + g(\mathbf{z}) \text{ s.t. } \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} = \mathbf{c}, \tag{6}$$

where $\mathbf{A} \in \mathbb{R}^{p \times n}, \mathbf{B} \in \mathbb{R}^{p \times m}, \mathbf{c} \in \mathbb{R}^p$ are constants and $\mathbf{x} \in \mathbb{R}^n, \mathbf{z} \in \mathbb{R}^m$ are optimization variables. $f : \mathbb{R}^n \mapsto \mathbb{R}$ and $g : \mathbb{R}^m \mapsto \mathbb{R}$ are two convex functions. The augmented Lagrangian can be formed as:

$$\begin{aligned} \mathfrak{L}_\rho(\mathbf{x}, \mathbf{z}, \mathbf{y}) = f(\mathbf{x}) + g(\mathbf{z}) + \mathbf{y}^\mathsf{T}(\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}) \\ + \frac{\rho}{2}\|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}\|_2^2, \end{aligned} \tag{7}$$

where $\mathbf{y} \in \mathbb{R}^p$ is the Lagrange multiplier and $\rho > 0$ is some constant. The ADMM then iterates over the following three steps for $k \geq 0$ (the iteration index):

$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \mathfrak{L}_\rho\left(\mathbf{x}, \mathbf{z}^k, \mathbf{y}^k\right), \tag{8}$$

$$\mathbf{z}^{k+1} = \arg\min_{\mathbf{z}} \mathfrak{L}_\rho\left(\mathbf{x}^{k+1}, \mathbf{z}, \mathbf{y}^k\right), \tag{9}$$

$$\mathbf{y}^{k+1} = \mathbf{y}^k + \rho\left(\mathbf{A}\mathbf{x}^{k+1} + \mathbf{B}\mathbf{z}^{k+1} - \mathbf{c}\right). \tag{10}$$

The ADMM is guaranteed to converge to the optimal point of (6) as long as f and g are convex [28], [34]. It is recently shown that global linear convergence can be ensured provided additional assumptions on problem (6) holds [31].

C. Development of the Distributed ADMM (DADMM) for Network Cost Minimization

To develop an ADMM algorithm for (1), we introduce auxiliary variables $\mathbf{y}_i$ and $\mathbf{z}_{ij}$ $\forall i, j \in \Omega_i$ and reformulate (1) equivalently as:

$$\text{Minimize} \quad \sum_{i=1}^{n} f_i(\mathbf{x}_i) + \sum_{i=1}^{n}\sum_{j\in\Omega_i} g_{ij}(\mathbf{y}_i, \mathbf{z}_{ij}). \tag{11}$$

$$\text{s.t.} \quad \mathbf{x}_i = \mathbf{y}_i, \ i = 1, ..., n, \tag{12}$$

$$\mathbf{x}_j = \mathbf{z}_{ij}, \ i = 1, ..., n, j \in \Omega_i. \tag{13}$$

Further introducing Lagrangian multipliers $\boldsymbol{\lambda}_i, \boldsymbol{\mu}_{ij} \in \mathbb{R}^p, \forall i = 1, ..., n, j \in \Omega_i$, we form the augmented Lagrangian of the above optimization problem as:

$$\begin{aligned}\mathfrak{L}_\rho(\mathbf{x}, \mathbf{y}, \mathbf{z}, \boldsymbol{\lambda}, \boldsymbol{\mu}) &= \sum_{i=1}^{n} f_i(\mathbf{x}_i) + \sum_{i=1}^{n}\sum_{j\in\Omega_i} g_{ij}(\mathbf{y}_i, \mathbf{z}_{ij}) \\ &+ \sum_{i=1}^{n} \boldsymbol{\lambda}_i^{\mathsf{T}}(\mathbf{x}_i - \mathbf{y}_i) + \sum_{i=1}^{n}\sum_{j\in\Omega_i} \boldsymbol{\mu}_{ij}^{\mathsf{T}}(\mathbf{x}_j - \mathbf{z}_{ij}) \\ &+ \frac{\rho}{2}\sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{y}_i\|_2^2 + \frac{\rho}{2}\sum_{i=1}^{n}\sum_{j\in\Omega_i} \|\mathbf{x}_j - \mathbf{z}_{ij}\|_2^2, \end{aligned} \tag{14}$$

where $\mathbf{x} \in \mathbb{R}^{np}$ is the concatenation of all $\mathbf{x}_i$'s into a column vector, i.e., $\mathbf{x} = \left[\mathbf{x}_1^{\mathsf{T}}, ..., \mathbf{x}_n^{\mathsf{T}}\right]^{\mathsf{T}}$; $\mathbf{y}, \boldsymbol{\lambda} \in \mathbb{R}^{np}$ are analogously defined; $\mathbf{z} \in \mathbb{R}^{mp}$ is the concatenation of all $\mathbf{z}_{ij}$'s in an arbitrary order of links; $\boldsymbol{\mu} \in \mathbb{R}^{mp}$ is analogously defined with the same link order as $\mathbf{z}$; $\rho > 0$ is some positive constant. The ADMM algorithm can be derived as follows.

1) Updating $\mathbf{x}$*:* The update of $\mathbf{x}$ in the ADMM is:

$$\begin{aligned}\mathbf{x}^{k+1} &= \arg\min_{\mathbf{x}} \sum_{i=1}^{n} f_i(\mathbf{x}_i) + \sum_{i=1}^{n} \boldsymbol{\lambda}_i^{k\mathsf{T}}\mathbf{x}_i + \sum_{i=1}^{n}\sum_{j\in\Omega_i} \boldsymbol{\mu}_{ij}^{k\mathsf{T}}\mathbf{x}_j \\ &+ \frac{\rho}{2}\sum_{i=1}^{n} \left\|\mathbf{x}_i - \mathbf{y}_i^k\right\|_2^2 + \frac{\rho}{2}\sum_{i=1}^{n}\sum_{j\in\Omega_i} \left\|\mathbf{x}_j - \mathbf{z}_{ij}^k\right\|_2^2, \end{aligned} \tag{15}$$

which can be decomposed across nodes: $\forall i$,

$$\begin{aligned}\mathbf{x}_i^{k+1} &= \arg\min_{\mathbf{x}_i} f_i(\mathbf{x}_i) + \boldsymbol{\lambda}_i^{k\mathsf{T}}\mathbf{x}_i + \sum_{l\in\Omega_i} \boldsymbol{\mu}_{li}^{k\mathsf{T}}\mathbf{x}_i \\ &+ \frac{\rho}{2}\left\|\mathbf{x}_i - \mathbf{y}_i^k\right\|_2^2 + \frac{\rho}{2}\sum_{l\in\Omega_i} \left\|\mathbf{x}_i - \mathbf{z}_{li}^k\right\|_2^2. \end{aligned} \tag{16}$$

2) Updating $\mathbf{y}, \mathbf{z}$*:* The update of $\mathbf{y}, \mathbf{z}$ in the ADMM is:

$$\begin{aligned}&\left\{\mathbf{y}^{k+1}, \mathbf{z}^{k+1}\right\} \\ &= \arg\min_{\mathbf{y},\mathbf{z}} \sum_{i=1}^{n}\sum_{j\in\Omega_i} g_{ij}(\mathbf{y}_i, \mathbf{z}_{ij}) - \sum_{i=1}^{n} \boldsymbol{\lambda}_i^{k\mathsf{T}}\mathbf{y}_i \\ &- \sum_{i=1}^{n}\sum_{j\in\Omega_i} \boldsymbol{\mu}_{ij}^{k\mathsf{T}}\mathbf{z}_{ij} + \frac{\rho}{2}\sum_{i=1}^{n} \left\|\mathbf{y}_i - \mathbf{x}_i^{k+1}\right\|_2^2 \\ &+ \frac{\rho}{2}\sum_{i=1}^{n}\sum_{j\in\Omega_i} \left\|\mathbf{z}_{ij} - \mathbf{x}_j^{k+1}\right\|_2^2, \end{aligned} \tag{17}$$

Algorithm 1: The DADMM algorithm run at node i.

```
1: Initialize $\mathbf{x}_i^0 = \mathbf{y}_i^0 = \boldsymbol{\lambda}_i^0 = \mathbf{0}$ and $\mathbf{z}_{ij}^0 = \boldsymbol{\mu}_{ij}^0 = \mathbf{0}, \forall j \in \Omega_i$. $k = 0$.
2: Repeat:
3:    Compute $\mathbf{x}_i^{k+1}$ by solving the local optimization problem (16) and then broadcast $\mathbf{x}_i^{k+1}$ to the neighbors $\Omega_i$.
4:    Compute $\mathbf{y}_i^{k+1}$ and $\mathbf{z}_{ij}^{k+1}, j \in \Omega_i$ by solving the local optimization problem (18) and then transmit $\mathbf{z}_{ij}^{k+1}$ to the neighbor node $j$ for each $j \in \Omega_i$.
5:    Compute $\boldsymbol{\lambda}_i^{k+1}$ and $\boldsymbol{\mu}_{ij}^{k+1}, j \in \Omega_i$ according to (19) and (20), respectively. Transmit $\boldsymbol{\mu}_{ij}^{k+1}$ to the neighbor node $j$ for each $j \in \Omega_i$.
6:    $k \leftarrow k + 1$.
```

which can be decomposed across nodes: $\forall i$,

$$\begin{aligned}&\left\{\mathbf{y}_i^{k+1}, \left\{\mathbf{z}_{ij}^{k+1}\right\}_{j\in\Omega_i}\right\} \\ &= \arg\min_{\mathbf{y}_i, \{\mathbf{z}_{ij}\}_{j\in\Omega_i}} \sum_{j\in\Omega_i} g_{ij}(\mathbf{y}_i, \mathbf{z}_{ij}) - \boldsymbol{\lambda}_i^{k\mathsf{T}}\mathbf{y}_i - \sum_{j\in\Omega_i} \boldsymbol{\mu}_{ij}^{k\mathsf{T}}\mathbf{z}_{ij} \\ &+ \frac{\rho}{2}\left\|\mathbf{y}_i - \mathbf{x}_i^{k+1}\right\|_2^2 + \frac{\rho}{2}\sum_{j\in\Omega_i} \left\|\mathbf{z}_{ij} - \mathbf{x}_j^{k+1}\right\|_2^2. \end{aligned} \tag{18}$$

3) Updating $\boldsymbol{\lambda}, \boldsymbol{\mu}$*:* The update of $\boldsymbol{\lambda}, \boldsymbol{\mu}$ is also decomposed across nodes: $\forall i, j \in \Omega_i$

$$\boldsymbol{\lambda}_i^{k+1} = \boldsymbol{\lambda}_i^k + \rho\left(\mathbf{x}_i^{k+1} - \mathbf{y}_i^{k+1}\right), \tag{19}$$

$$\boldsymbol{\mu}_{ij}^{k+1} = \boldsymbol{\mu}_{ij}^k + \rho\left(\mathbf{x}_j^{k+1} - \mathbf{z}_{ij}^{k+1}\right). \tag{20}$$

Equations (16), (18), (19) and (20) together lead to a distributed ADMM (DADMM) algorithm for problem (1), which is summarized from the perspective of an arbitrary node i in Algorithm 1. We note that only the values of $\mathbf{x}, \mathbf{z}, \boldsymbol{\mu}$ at the neighbors are needed for the ADMM updates. Therefore, in terms of information exchange, each node i only needs to (i) broadcast $\mathbf{x}_i$ to the neighbors in Ω_i; (ii) transmit $\mathbf{z}_{ij}$ to the neighbor j for each $j \in \Omega_i$; (iii) transmit $\boldsymbol{\mu}_{ij}$ to the neighbor j for each $j \in \Omega_i$.

D. Development of the Distributed Linearized ADMM (DLADMM) for Network Cost Minimization

In the DADMM, i.e., Algorithm 1, the updates for $\mathbf{x}, \mathbf{y}, \mathbf{z}$ involve solving local optimization problems (16) and (18), which generally do not admit close-form solutions and have to be solved iteratively. This can be a major computational burden for Algorithm 1 especially when individual node has only limited computational capability, e.g., the cheap sensors vastly deployed in sensor networks usually can only carry out simple calculations. This motivates us to propose an algorithm which can approximately solve the local optimization problems efficiently and most preferably with closed form solutions. To this end, we first define $f(\mathbf{x}) = \sum_{i=1}^{n} f_i(\mathbf{x}_i)$ and $g(\mathbf{y}, \mathbf{z}) = \sum_{i=1}^{n}\sum_{j\in\Omega_i} g_{ij}(\mathbf{y}_i, \mathbf{z}_{ij})$. We further define a block matrix $\mathbf{A} \in \mathbb{R}^{mp\times np}$ consisting of $m \times n$ blocks of matrices $\mathbf{A}_{kj} \in \mathbb{R}^{p\times p}$, where $\mathbf{A}_{kj}$ is equal to $\mathbb{I}_{p\times p}$ if the k-th

p-dimensional block of $\mathbf{z}$ is $\mathbf{z}_{ij}$ for some $i = 1, ..., n$, otherwise $\mathbf{A}_{kj}$ is equal to $\mathbf{0}_{p\times p}$. Then, we may rewrite problem (11) compactly as:

$$\text{Minimize } f(\mathbf{x}) + g(\mathbf{y}, \mathbf{z}) \tag{21}$$

$$\text{s.t. } \mathbf{x} = \mathbf{y}, \tag{22}$$

$$\mathbf{A}\mathbf{x} = \mathbf{z}. \tag{23}$$

Further define $\mathbf{w} = \left[\mathbf{y}^\mathsf{T}, \mathbf{z}^\mathsf{T}\right]^\mathsf{T}$ and $\mathbf{B} = \left[\mathbf{I}, \mathbf{A}^\mathsf{T}\right]^\mathsf{T}$. Thus, (21) can be rewritten as:

$$\text{Minimize } f(\mathbf{x}) + g(\mathbf{w}) \tag{24}$$

$$\text{s.t. } \mathbf{B}\mathbf{x} - \mathbf{w} = \mathbf{0}. \tag{25}$$

The augmented Lagrangian can be written as:

$$\begin{aligned}&\mathfrak{L}_\rho(\mathbf{x}, \mathbf{w}, \boldsymbol{\alpha})\\ &= f(\mathbf{x}) + g(\mathbf{w}) + \alpha^\mathsf{T}(\mathbf{B}\mathbf{x} - \mathbf{w}) + \frac{\rho}{2}\|\mathbf{B}\mathbf{x} - \mathbf{w}\|_2^2,\end{aligned} \tag{26}$$

where $\boldsymbol{\alpha} = [\boldsymbol{\lambda}^\mathsf{T}, \boldsymbol{\mu}^\mathsf{T}]^\mathsf{T}$ is the Lagrangian multiplier. The original DADMM algorithm necessitates solving local optimization problems involving f and g. To avoid this burden, inspired by the variants of ADMM proposed in [17], [18] for consensus optimization, we approximate f, g with their first order approximations and propose a distributed linearized ADMM (DLADMM) algorithm for network cost minimization in the following.

1) Updating $\mathbf{x}$*:* The update of $\mathbf{x}$ in DLADMM is:

$$\begin{aligned}\mathbf{x}^{k+1} &= \arg\min_{\mathbf{x}} \nabla f\left(\mathbf{x}^k\right)^\mathsf{T}\left(\mathbf{x} - \mathbf{x}^k\right) + \frac{c}{2}\left\|\mathbf{x} - \mathbf{x}^k\right\|_2^2\\ &\quad + \boldsymbol{\alpha}^{k\mathsf{T}}\mathbf{B}\mathbf{x} + \frac{\rho}{2}\left\|\mathbf{B}\mathbf{x} - \mathbf{w}^k\right\|_2^2,\end{aligned} \tag{27}$$

where $c > 0$ is some positive constant and the term $\frac{c}{2}\left\|\mathbf{x} - \mathbf{x}^k\right\|_2^2$ is to refrain $\mathbf{x}^{k+1}$ from being too far away from $\mathbf{x}^k$ as the first order approximation of f around the point $\mathbf{x}^k$ is only accurate when $\mathbf{x}$ is close to $\mathbf{x}^k$. Note that this small step size or small variation between iterations is common in the literature of numerical optimization [30] and adaptive signal processing such as least mean squares (LMS) [35]. Another meaning of the term $\frac{c}{2}\|\mathbf{x} - \mathbf{x}^k\|_2^2$ is that it provides a quadratic approximation of f at $\mathbf{x}^k$, namely,

$$f(\mathbf{x}) \approx f(\mathbf{x}^k) + \nabla f(\mathbf{x}^k)^\mathsf{T}(\mathbf{x} - \mathbf{x}^k) + \frac{c}{2}\|\mathbf{x} - \mathbf{x}^k\|_2^2.$$

This approximation is common in the literature of numerical optimization and is indeed one of the most fundamental concepts of gradient-based optimization. In fact, problem (27) can be viewed as an iteration of solving $\min_{\mathbf{x}} \mathfrak{L}_\rho(\mathbf{x}, \mathbf{w}^k, \boldsymbol{\alpha}^k)$ approximately, by performing one proximal gradient step [36].

Since the objective function in (27) is a convex quadratic function of $\mathbf{x}$, the problem of (27) can be solved in closed form through the first order condition:

$$\begin{aligned}&\nabla f\left(\mathbf{x}^k\right) + c\left(\mathbf{x}^{k+1} - \mathbf{x}^k\right) + \mathbf{B}^\mathsf{T}\boldsymbol{\alpha}^k\\ &\quad + \rho\left(\mathbf{B}^\mathsf{T}\mathbf{B}\mathbf{x}^{k+1} - \mathbf{B}^\mathsf{T}\mathbf{w}^k\right) = 0.\end{aligned} \tag{28}$$

We note that the optimization problem (27) can be decomposed across nodes:

$$\begin{aligned}&\mathbf{x}_i^{k+1}\\ &= \arg\min_{\mathbf{x}_i} \nabla f_i\left(\mathbf{x}_i^k\right)^\mathsf{T}\left(\mathbf{x}_i - \mathbf{x}_i^k\right) + \frac{c}{2}\left\|\mathbf{x}_i - \mathbf{x}_i^k\right\|_2^2 + \boldsymbol{\lambda}_i^{k\mathsf{T}}\mathbf{x}_i\\ &+ \sum_{l\in\Omega_i}\boldsymbol{\mu}_{li}^{k\mathsf{T}}\mathbf{x}_i + \frac{\rho}{2}\left\|\mathbf{x}_i - \mathbf{y}_i^k\right\|_2^2 + \frac{\rho}{2}\sum_{l\in\Omega_i}\left\|\mathbf{x}_i - \mathbf{z}_{li}^k\right\|_2^2,\end{aligned}$$

which can be solved in closed form:

$$\begin{aligned}\mathbf{x}_i^{k+1} = \frac{1}{c + \rho + \rho|\Omega_i|}\Bigg[&-\nabla f_i\left(\mathbf{x}_i^k\right) + c\mathbf{x}_i^k - \lambda_i^k - \sum_{l\in\Omega_i}\boldsymbol{\mu}_{li}^k\\ &+ \rho\mathbf{y}_i^k + \rho\sum_{l\in\Omega_i}\mathbf{z}_{li}^k\Bigg].\end{aligned} \tag{29}$$

2) Updating $\mathbf{w}$*, i.e.,* $\mathbf{y}$ *and* $\mathbf{z}$*:* The update of $\mathbf{w}$ in the DLADMM algorithm is:

$$\begin{aligned}\mathbf{w}^{k+1} = &\arg\min_{\mathbf{w}} \nabla g\left(\mathbf{w}^k\right)^\mathsf{T}\left(\mathbf{w} - \mathbf{w}^k\right) + \frac{c}{2}\left\|\mathbf{w} - \mathbf{w}^k\right\|_2^2\\ &- \boldsymbol{\alpha}^{k\mathsf{T}}\mathbf{w} + \frac{\rho}{2}\left\|\mathbf{w} - \mathbf{B}\mathbf{x}^{k+1}\right\|_2^2,\end{aligned} \tag{30}$$

which is equivalent to:

$$\nabla g(\mathbf{w}^k) + c\left(\mathbf{w}^{k+1} - \mathbf{w}^k\right) - \boldsymbol{\alpha}^k + \rho\left(\mathbf{w}^{k+1} - \mathbf{B}\mathbf{x}^{k+1}\right) = 0. \tag{31}$$

Notice that the problem (30) can also be decomposed across nodes:

$$\begin{aligned}&\left\{\mathbf{y}_i^{k+1}, \{\mathbf{z}_{ij}^{k+1}\}_{j\in\Omega_i}\right\}\\ &= \arg\min_{\mathbf{y}_i, \{\mathbf{z}_{ij}\}_{j\in\Omega_i}} \sum_{j\in\Omega_i}\begin{bmatrix}\nabla_{\mathbf{y}_i} g_{ij}\left(\mathbf{y}_i^k, \mathbf{z}_{ij}^k\right)\\ \nabla_{\mathbf{z}_{ij}} g_{ij}\left(\mathbf{y}_i^k, \mathbf{z}_{ij}^k\right)\end{bmatrix}^\mathsf{T}\begin{bmatrix}\mathbf{y}_i - \mathbf{y}_i^k\\ \mathbf{z}_{ij} - \mathbf{z}_{ij}^k\end{bmatrix}\\ &+ \frac{c}{2}\left\|\mathbf{y}_i - \mathbf{y}_i^k\right\|_2^2 + \frac{c}{2}\sum_{j\in\Omega_i}\left\|\mathbf{z}_{ij} - \mathbf{z}_{ij}^k\right\|_2^2 - \boldsymbol{\lambda}_i^{k\mathsf{T}}\mathbf{y}_i\\ &- \sum_{j\in\Omega_i}\boldsymbol{\mu}_{ij}^{k\mathsf{T}}\mathbf{z}_{ij} + \frac{\rho}{2}\left\|\mathbf{y}_i - \mathbf{x}_i^{k+1}\right\|_2^2 + \frac{\rho}{2}\sum_{j\in\Omega_i}\left\|\mathbf{z}_{ij} - \mathbf{x}_j^{k+1}\right\|_2^2,\end{aligned} \tag{32}$$

which can be solved as:

$$\begin{aligned}&\mathbf{y}_i^{k+1}\\ &= \frac{1}{c + \rho}\left[-\sum_{j\in\Omega_i}\nabla_{\mathbf{y}_i} g_{ij}\left(\mathbf{y}_i^k, \mathbf{z}_{ij}^k\right) + c\mathbf{y}_i^k + \boldsymbol{\lambda}_i^k + \rho\mathbf{x}_i^{k+1}\right],\end{aligned} \tag{33}$$

$$\mathbf{z}_{ij}^{k+1} = \frac{1}{c + \rho}\left[-\nabla_{\mathbf{z}_{ij}} g_{ij}\left(\mathbf{y}_i^k, \mathbf{z}_{ij}^k\right) + c\mathbf{z}_{ij}^k + \boldsymbol{\mu}_{ij}^k + \rho\mathbf{x}_j^{k+1}\right]. \tag{34}$$

3) Updating $\boldsymbol{\alpha}$*, i.e.,* $\boldsymbol{\lambda}$ *and* $\boldsymbol{\mu}$*:* The update of $\boldsymbol{\alpha}$ is:

$$\boldsymbol{\alpha}^{k+1} = \boldsymbol{\alpha}^k + \rho\left(\mathbf{B}\mathbf{x}^{k+1} - \mathbf{w}^{k+1}\right), \tag{35}$$

which can be implemented in a decentralized manner as in (19) and (20). In other words, the update of the dual variables in DLADMM is the same as that of DADMM. Combining (29),

Algorithm 2: The DLADMM algorithm run at node i.

```
1: Initialize $\mathbf{x}_i^0 = \mathbf{y}_i^0 = \boldsymbol{\lambda}_i^0 = \mathbf{0}$ and $\mathbf{z}_{ij}^0 = \boldsymbol{\mu}_{ij}^0 = \mathbf{0}, \forall j \in \Omega_i$. $k = 0$.
2: Repeat:
3:   Compute $\mathbf{x}_i^{k+1}$ according to (29) and then broadcast $\mathbf{x}_i^{k+1}$ to the neighbors $\Omega_i$.
4:   Compute $\mathbf{y}_i^{k+1}$ and $\mathbf{z}_{ij}^{k+1}, j \in \Omega_i$ according to (33) and (34), respectively. Then transmit $\mathbf{z}_{ij}^{k+1}$ to the neighbor node $j$ for each $j \in \Omega_i$.
5:   Compute $\boldsymbol{\lambda}_i^{k+1}$ and $\boldsymbol{\mu}_{ij}^{k+1}, j \in \Omega_i$ according to (19) and (20), respectively. Transmit $\boldsymbol{\mu}_{ij}^{k+1}$ to the neighbor node $j$ for each $j \in \Omega_i$.
6:   $k \leftarrow k + 1$.
```

(33), (34), (19) and (20) yields the proposed DLADMM algorithm, which is summarized in Algorithm 2. We remark that, as opposed to Algorithm 1, each iteration of Algorithm 2 only involves direct closed form computations without solving any local optimization problems iteratively. This enables DLADMM to enjoy significantly lower computational complexity compared to DADMM.

We remark that, in both the DADMM and the DLADMM, each node i needs to transmit $(2|\Omega_i| + 1)$ p-dimensional vectors, namely $\mathbf{x}_i$, $\mathbf{z}_{ij}$, $\boldsymbol{\mu}_{ij}$, $j \in \Omega_i$, to its neighbors in each iteration. Therefore, the per-node communication complexity of both the DADMM and the DLADMM is $(2|\Omega_i| + 1)p$, which is not a heavy burden as long as the network is not too densely connected.

We note that, in [37], an algorithm named Stochastic Gradient Augmented Lagrangian Method (SGALM) has been proposed. The SGALM is an variant of the ADMM algorithm that replaces the primal updates (8) and (9) with their respective one step gradient descent. Though SGALM bears some similarity with the proposed DLADMM, there are several subtle differences, which render the latter superior to the former, especially in the particular problem of network cost minimization considered in this paper. In the following, for the network cost minimization problem (24), we compare the $\mathbf{x}$-update of DLADMM with that of the SGALM. The $\mathbf{w}$-updates can be analogously compared. Specifically, the $\mathbf{x}$-update of SGALM for problem (24) is:

$$\begin{aligned}\mathbf{x}^{k+1} &= \mathbf{x}^k - \eta \nabla_{\mathbf{x}} \mathfrak{L}_\rho(\mathbf{x}^k, \mathbf{w}^k, \boldsymbol{\alpha}^k) \qquad (36)\\ &= \mathbf{x}^k - \eta \left[\nabla f(\mathbf{x}^k) + \mathbf{B}^\mathsf{T}\boldsymbol{\alpha}^k + \rho \mathbf{B}^\mathsf{T}\left(\mathbf{B}\mathbf{x}^k - \mathbf{w}^k\right)\right], \qquad (37)\end{aligned}$$

where $\eta > 0$ is the stepsize. On the other hand, the $\mathbf{x}$-update of the proposed DLADMM given in (28) can be rewritten as:

$$\begin{aligned}\mathbf{x}^{k+1} = &\left(c\mathbf{I} + \rho\mathbf{B}^\mathsf{T}\mathbf{B}\right)^{-1}\\ &\cdot \left[-\nabla f(\mathbf{x}^k) + c\mathbf{x}^k - \mathbf{B}^\mathsf{T}\boldsymbol{\alpha}^k + \rho\mathbf{B}^\mathsf{T}\mathbf{w}^k\right]. \qquad (38)\end{aligned}$$

Thus, we can see that the $\mathbf{x}$-update in SGALM is different from that in the proposed DLADMM. In particular, the $\mathbf{x}$-update in DLADMM includes a matrix inversion $\left(c\mathbf{I} + \rho\mathbf{B}^\mathsf{T}\mathbf{B}\right)^{-1}$ while that of SGALM does not. The reason of the difference between the two algorithms is as follows. In the $\mathbf{x}$-update of DLADMM, only the term $f(\mathbf{x})$ is linearized while the term $\frac{\rho}{2}\left\|\mathbf{B}\mathbf{x} - \mathbf{w}^k\right\|_2^2$ remains unchanged in the objective (c.f. (27)). In constrast, in the $\mathbf{x}$-update of SGALM, the derivative is taken for the whole augmented Lagrangian $\mathfrak{L}_\rho(\mathbf{x}, \mathbf{w}^k, \boldsymbol{\alpha}^k)$ so that the term $\frac{\rho}{2}\left\|\mathbf{B}\mathbf{x} - \mathbf{w}^k\right\|_2^2$ is also differentiated, i.e., it is also implicitely linearized. This linearization will deteriorate the accuracy of the approximation of the original ADMM update and is indeed unnecessary from a computational perspective since, without it, the DLADMM $\mathbf{x}$-update in (38) can still be implemented in a decentralized computationally efficient manner (c.f. (29)), thanks to the special structure of the matrix $\mathbf{B}$ in the network cost minimization problem under study. Due to the aforementioned discrepancy between DLADMM and SGALM, the analysis in [37] cannot be applied to the proposed DLADMM in this paper. Furthermore, the theoretical guarantee for the convergence rate of SGALM is $O(1/k)$ in the deterministic setting under the minimal assumption that the objective function is convex. In contrast, under the more stringent assumption of strong convexity, we show that the convergence rate of DLADMM is exponential (linear convergence) both theoretically (c.f. Theorem 2) and empirically (cf. Fig. 1). This further suggests that the convergence analysis of DLADMM in this paper is substantially different from that of the SGALM in [37].

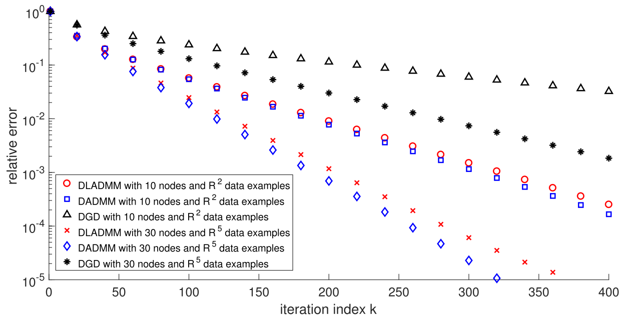


Fig. 1. Comparison between the DLADMM, the DADMM and the DGD.

Furthermore, we note that two algorithms, namely the proximal ADMM and the majorized ADMM, have been developed in [22] for non-convex non-differentiable distributed optimization, extending the recent work [24] on non-convex ADMM. These algorithms are different from the proposed DLADMM, which enjoys lower computational complexity. Specifically, in the majorized ADMM, there is no linearization in the algorithm design. In each iteration, each node needs to solve two optimization subproblems related to the node cost function and the convex surrogate function of the link cost function, respectively. Though these two subproblems are both convex, solving them may still incur considerable computational burden in many cases, e.g., the distributed logistic regression problem which involves logarithmic functions in the objective function (c.f. Section IV). On the other hand, in the proximal ADMM, though the link cost subproblem is linearized to give closed-form update equations, the node cost subproblem still needs to be solved exactly (through proximal operator), which can be computationally demanding. In contrast, in the proposed DLADMM in this paper, both the node cost subproblem (i.e., update of $\mathbf{x}$) and the link cost subproblem (i.e., update of $\mathbf{y}, \mathbf{z}$ or equivalently $\mathbf{w}$) are linearized, leading to simple closed-form updates for all the involved variables and rendering the DLADMM very computationally efficient. Due to these discrepancies in algorithm design between the methods in [22] and the proposed DLADMM, the convergence analyses of this paper is very different from those in [22]. Besides, the algorithms in [22] are only guaranteed to converge, but there is no theoretical results regarding their convergence rates. In contrast, the proposed DLADMM will be shown to converge linearly to the optimum both theoretically (c.f. Theorem 2) and empirically (cf. Fig. 1).

In addition, though linearization has been applied to consensus optimization in prior works, it has not been used in the formulated network cost minimization problem (1), which has attracted little attention in the literature. In fact, due to the existence of link cost functions g_{ij} in (1), the convergence analysis of the linearized ADMM for consensus optimization in [17] no longer holds and the impact of linearization on the performance of distributed ADMM for problem (1) remains unknown. This motivates us to study the performance of the DLADMM for problem (1) rigorously in this paper. We note that the extension of the convergence analysis of DLADMM to problem (1) is nontrivial, due to the presence of the link cost functions g_{ij} and the new coupling structure of individual variables.

As a last remark, we note that the network cost minimization problem, i.e., problem (1), cannot be readily converted into a consensus optimization problem in a computationally efficient way and should not be viewed as a special case of consensus optimization. To forcibly convert problem (1) into consensus optimization, one has to concatenate all the local variables into one high dimensional variable $\mathbf{x} = [\mathbf{x}_1^\mathsf{T}, ..., \mathbf{x}_n^\mathsf{T}]^\mathsf{T}$ and define the functions $\phi_i(\mathbf{x}) = f_i(\mathbf{x}_i) + \sum_{j\in\Omega_i} g_{ij}(\mathbf{x}_i, \mathbf{x}_j)$. Thus, problem (1) is equivalent to minimizing $\sum_{i=1}^n \phi_i(\mathbf{x})$, which is a consensus optimization problem. However, such an awkward conversion is almost useless in designing practical computationally tractable algorithms since $\mathbf{x}$ is of very high dimension, even in moderately large scale networks. It is computationally prohibitive for each node to update and exchange a local version of $\mathbf{x}$ so that virtually all existing methods on consensus optimization will fail. Therefore, practically, problem (1) is not a special case of consensus optimization and consensus ADMM cannot be applied to problem (1) directly.

III. Convergence Analysis

In this section, we analyze the convergence behaviors of the proposed DLADMM algorithm for the network cost minimization problem (1). Instead of analyzing the DLADMM algorithm outlined in Algorithm 2, we will analyze its centralized version in (28), (31) and (35), which are tantamount to their decentralized counterpart in Algorithm 2. We perform convergence analysis based on (28), (31) and (35) as they are more compact and thus more amenable to analyses and expositions.[1] Before formally analyzing the convergence of DLADMM, we first present some preliminaries. After that, we show the convergence guarantee of the DLADMM (Theorem 1) and in particular, a linear convergence rate of the DLADMM (Theorem 2).

A. Preliminaries

First, we can derive Lipschitz continuity of ∇f and ∇g from Assumption 2 as follows.

Lemma 1: If Assumption 2 holds, then ∇f is Lipschitz continuous with constant L and ∇g is Lipschitz continuous with constant $M = \sqrt{L^2K^2 + L^2K}$, where $K = \max_i|\Omega_i|$ is the maximum degree of the network.

We further note the following fact from convex analysis [38].

Lemma 2: For any differentiable convex function $h : \mathbb{R}^l \mapsto \mathbb{R}$ and positive constant $L > 0$, the following two statements are equivalent:

1) ∇h is Lipschitz continuous with constant L, i.e., $\|\nabla h(\mathbf{x}) - \nabla h(\mathbf{x}')\|_2 \le L\|\mathbf{x} - \mathbf{x}'\|_2, \forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^l$.
2) $\forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^l$:

$$\|\nabla h(\mathbf{x}) - \nabla h(\mathbf{x}')\|_2^2 \le L(\mathbf{x} - \mathbf{x}')^\mathsf{T}(\nabla h(\mathbf{x}) - \nabla h(\mathbf{x}')).$$

Utilizing the Lemma 1 and Lemma 2, we immediately have the following result.

Lemma 3: If Assumptions 1 and 2 hold, we have: $\forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^{np}$,

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{x}')\|_2^2 \le L(\mathbf{x} - \mathbf{x}')^\mathsf{T}(\nabla f(\mathbf{x}) - \nabla f(\mathbf{x}')), \tag{39}$$

and $\forall \mathbf{w}, \mathbf{w}' \in \mathbb{R}^{np+mp}$,

$$\|\nabla g(\mathbf{w}) - \nabla g(\mathbf{w}')\|_2^2 \le M(\mathbf{w} - \mathbf{w}')^\mathsf{T}(\nabla g(\mathbf{w}) - \nabla g(\mathbf{w}')), \tag{40}$$

where M is defined in Lemma 1.

B. Convergence

We define a diagonal positive definite matrix $\boldsymbol{\Lambda}$ as:

$$\boldsymbol{\Lambda} = \begin{bmatrix} \frac{c}{2}\mathbf{I}_{np} & & \\ & \frac{\rho+c}{2}\mathbf{I}_{np+mp} & \\ & & \frac{1}{2\rho}\mathbf{I}_{np+mp} \end{bmatrix}. \tag{41}$$

For ease of notation, we define $\mathbf{u} \in \mathbb{R}^{3np+2mp}$ to be the concatenation of $\mathbf{x}, \mathbf{w}, \boldsymbol{\alpha}$ into a single column vector and similarly for $\mathbf{u}^k, \mathbf{u}^*$. Since $\boldsymbol{\Lambda}$ is a positive definite matrix, we can further define a norm on $\mathbb{R}^{3np+2mp}$ as: $\|\mathbf{u}\|_{\boldsymbol{\Lambda}} = \sqrt{\mathbf{u}^\mathsf{T}\boldsymbol{\Lambda}\mathbf{u}}$. We have the following result.

Proposition 1: Suppose Assumptions 1 and 2 hold. Then, for any primal/dual optimal point of problem (24) $\mathbf{u}^* = [\mathbf{x}^{*\mathsf{T}}, \mathbf{w}^{*\mathsf{T}}, \boldsymbol{\alpha}^{*\mathsf{T}}]^\mathsf{T}$, the sequence $\mathbf{u}^k = \left[\mathbf{x}^{k\mathsf{T}}, \mathbf{w}^{k\mathsf{T}}, \boldsymbol{\alpha}^{k\mathsf{T}}\right]^\mathsf{T}$ generated by the DLADMM algorithm satisfies $\forall k \ge 0$:

$$\left\|\mathbf{u}^{k+1} - \mathbf{u}^*\right\|_{\boldsymbol{\Lambda}}^2 \le \left\|\mathbf{u}^k - \mathbf{u}^*\right\|_{\boldsymbol{\Lambda}}^2 - \left(\frac{c}{2} - \frac{L}{4}\right)\left\|\mathbf{x}^k - \mathbf{x}^{k+1}\right\|_2^2 - \left(\frac{c-\rho}{2} - \frac{M}{4}\right)\left\|\mathbf{w}^k - \mathbf{w}^{k+1}\right\|_2^2 - \frac{1}{4\rho}\left\|\boldsymbol{\alpha}^{k+1} - \boldsymbol{\alpha}^k\right\|_2^2.$$

Proof: The proof is given in Appendix A. ■

[1] Note that completely equivalent analysis based on the decentralized implementation in Algorithm 2 can be conducted, though the notations are more cluttered.

Now, we are ready to state our first main theorem of convergence.

Theorem 1: Suppose Assumptions 1,2 hold and $c > \frac{M}{2} + \rho$. Then, the sequence $\mathbf{u}^k$ generated by the DLADMM algorithm converges to some primal/dual optimal point of problem (24), i.e., there exists a primal/dual optimal point of problem (24) $\mathbf{u}^*$ such that $\lim_{k\to\infty}\mathbf{u}^k = \mathbf{u}^*$.

Proof: Given any primal/dual optimal point of problem (24) $\mathbf{u}^*$, according to Proposition 1 and $c > \frac{M}{2} + \rho$, we know that $\left\|\mathbf{u}^k - \mathbf{u}^*\right\|_{\mathbf{\Lambda}}^2$ is a decreasing sequence. Since it is clearly lower bounded by 0, we have that $\left\|\mathbf{u}^k - \mathbf{u}^*\right\|_{\mathbf{\Lambda}}^2$ is convergent. From Proposition 1, we further deduce that:

$$0 \le \left(\frac{c}{2} - \frac{L}{4}\right)\left\|\mathbf{x}^k - \mathbf{x}^{k+1}\right\|_2^2 + \left(\frac{c-\rho}{2} - \frac{M}{4}\right)\left\|\mathbf{w}^k - \mathbf{w}^{k+1}\right\|_2^2 + \frac{1}{4\rho}\left\|\boldsymbol{\alpha}^{k+1} - \boldsymbol{\alpha}^k\right\|_2^2 \tag{42}$$

$$\le \left\|\mathbf{u}^k - \mathbf{u}^*\right\|_{\mathbf{\Lambda}}^2 - \left\|\mathbf{u}^{k+1} - \mathbf{u}^*\right\|_{\mathbf{\Lambda}}^2. \tag{43}$$

Because $\left\|\mathbf{u}^k - \mathbf{u}^*\right\|_{\mathbf{\Lambda}}^2$ is convergent, we know that the quantity in (43) converges to zero as k goes to infinity. Hence, the quantity in (42) converges to zero as well. Therefore,

$$\lim_{k\to\infty}\left(\mathbf{x}^k - \mathbf{x}^{k+1}\right) = \mathbf{0}, \tag{44}$$

$$\lim_{k\to\infty}\left(\mathbf{w}^k - \mathbf{w}^{k+1}\right) = \mathbf{0}, \tag{45}$$

$$\lim_{k\to\infty}\left(\boldsymbol{\alpha}^k - \boldsymbol{\alpha}^{k+1}\right) = \mathbf{0}. \tag{46}$$

Substituting the above limits into (28), (31) and (35) yields:

$$\lim_{k\to\infty}\left[\nabla f\left(\mathbf{x}^k\right) + \mathbf{B}^{\mathsf{T}}\boldsymbol{\alpha}^k + \rho\left(\mathbf{B}^{\mathsf{T}}\mathbf{B}\mathbf{x}^{k+1} - \mathbf{B}^{\mathsf{T}}\mathbf{w}^k\right)\right] = \mathbf{0}, \tag{47}$$

$$\lim_{k\to\infty}\left[\nabla g\left(\mathbf{w}^k\right) - \boldsymbol{\alpha}^k + \rho\left(\mathbf{w}^{k+1} - \mathbf{B}\mathbf{x}^{k+1}\right)\right] = \mathbf{0}, \tag{48}$$

$$\lim_{k\to\infty}\left(\mathbf{B}\mathbf{x}^{k+1} - \mathbf{w}^{k+1}\right) = \mathbf{0}. \tag{49}$$

Equation (49) clearly implies:

$$\lim_{k\to\infty}\left(\mathbf{B}\mathbf{x}^k - \mathbf{w}^k\right) = \mathbf{0}. \tag{50}$$

Combining (48) and (49) leads to:

$$\lim_{k\to\infty}\left[\nabla g\left(\mathbf{w}^k\right) - \boldsymbol{\alpha}^k\right] = \mathbf{0}. \tag{51}$$

Moreover, from (49) and (45), we obtain:

$$\mathbf{B}\mathbf{x}^{k+1} - \mathbf{w}^k = \mathbf{B}\mathbf{x}^{k+1} - \mathbf{w}^{k+1} + \mathbf{w}^{k+1} - \mathbf{w}^k \to \mathbf{0}, \quad \text{as } k \to \infty. \tag{52}$$

Combining (52) and (47), we get:

$$\lim_{k\to\infty}\left[\nabla f\left(\mathbf{x}^k\right) + \mathbf{B}^{\mathsf{T}}\boldsymbol{\alpha}^k\right] = \mathbf{0}. \tag{53}$$

Since $\forall k : \|\mathbf{u}^k - \mathbf{u}^*\|_{\mathbf{\Lambda}} \le \|\mathbf{u}^0 - \mathbf{u}^*\|_{\mathbf{\Lambda}}$, we know that $\{\mathbf{u}^k\}_{k=0,1,\ldots}$ is a bounded sequence. So, it has convergent subsequence, which is denoted as $\left\{\mathbf{u}^{k_i}\right\}_{i=1,2,\ldots}$. Let $\widehat{\mathbf{u}}$ be the limit of this convergent subsequence, i.e., $\lim_{i\to\infty}\mathbf{u}^{k_i} = \widehat{\mathbf{u}}$. Equations (50), (51) and (53) are still satisfied along the subsequence $\left\{\mathbf{u}^{k_i}\right\}_{i=1,2,\ldots}$ and hence,

$$\lim_{i\to\infty}\left(\mathbf{B}\mathbf{x}^{k_i} - \mathbf{w}^{k_i}\right) = \mathbf{0}, \tag{54}$$

$$\lim_{i\to\infty}\left[\nabla g\left(\mathbf{w}^{k_i}\right) - \boldsymbol{\alpha}^{k_i}\right] = \mathbf{0}, \tag{55}$$

$$\lim_{i\to\infty}\left[\nabla f\left(\mathbf{x}^{k_i}\right) + \mathbf{B}^{\mathsf{T}}\boldsymbol{\alpha}^{k_i}\right] = \mathbf{0}. \tag{56}$$

Making use of the convergence of the subsequence $\{\mathbf{u}^{k_i}\}_{i=1,2,\ldots}$ to $\widehat{\mathbf{u}}$, we obtain:

$$\mathbf{B}\widehat{\mathbf{x}} - \widehat{\mathbf{w}} = \mathbf{0}, \tag{57}$$

$$\nabla g\left(\widehat{\mathbf{w}}\right) - \widehat{\boldsymbol{\alpha}} = \mathbf{0}, \tag{58}$$

$$\nabla f\left(\widehat{\mathbf{x}}\right) + \mathbf{B}^{\mathsf{T}}\widehat{\boldsymbol{\alpha}} = \mathbf{0}. \tag{59}$$

These are the KKT conditions of problem (24). So $\widehat{\mathbf{u}}$ is a primal/dual optimal point of problem (24). In the following, we endeavor to show that the sequence $\mathbf{u}^k$ converges to $\widehat{\mathbf{u}}$. Before that, we first present a lemma without proof.

Lemma 4: If the sequence $\{\mathbf{u}^k\}_{k=0,1,\ldots}$ has two subsequences $\{\mathbf{u}^{k_i}\}_{i=1,2,\ldots}$ and $\{\mathbf{u}^{k_i'}\}_{i=1,2,\ldots}$ converging to $\underline{\mathbf{u}}$ and $\overline{\mathbf{u}}$, respectively, then $\underline{\mathbf{u}} = \overline{\mathbf{u}}$.

Now, we show that $\mathbf{u}^k$ converges to $\widehat{\mathbf{u}}$ by making use of Lemma 4. Suppose, on the contrary, $\mathbf{u}^k$ does not converge to $\widehat{\mathbf{u}}$. Then, there exists some positive ϵ, such that for any positive integer N, there exists some $k \ge N$ with $\|\mathbf{u}^k - \widehat{\mathbf{u}}\|_2 \ge \epsilon$. Thus, letting $N = 1$, we get some $\tilde{k}_1 \ge 1$ with $\|\mathbf{u}^{\tilde{k}_1} - \widehat{\mathbf{u}}\|_2 \ge \epsilon$. Letting $N = \tilde{k}_1 + 1$, we get some $\tilde{k}_2 \ge \tilde{k}_1 + 1$ with $\|\mathbf{u}^{\tilde{k}_2} - \widehat{\mathbf{u}}\|_2 \ge \epsilon$. Continuing this process, we obtain a subsequence $\{\mathbf{u}^{\tilde{k}_i}\}_{i=1,2,\ldots}$ such that $\|\mathbf{u}^{\tilde{k}_i} - \widehat{\mathbf{u}}\|_2 \ge \epsilon, \forall i$. The subsequence $\{\mathbf{u}^{\tilde{k}_i}\}_{i=1,2,\ldots}$ is bounded as the original sequence $\{\mathbf{u}^k\}_{k=0,1,\ldots}$ is bounded. As such, the subsequence $\{\mathbf{u}^{\tilde{k}_i}\}_{i=1,2,\ldots}$ has a convergent sub-subsequence $\{\mathbf{u}^{\tilde{k}_{i_j}}\}_{j=1,2,\ldots}$. Denote the limit of this convergent sub-subsequence as $\widetilde{\mathbf{u}}$, i.e., $\lim_{j\to\infty}\mathbf{u}^{\tilde{k}_{i_j}} = \widetilde{\mathbf{u}}$. Obviously, $\|\widetilde{\mathbf{u}} - \widehat{\mathbf{u}}\|_2 \ge \epsilon$. But, according to Lemma 4, we should have $\widetilde{\mathbf{u}} = \widehat{\mathbf{u}}$. This is a contradiction. So, we must have $\lim_{k\to\infty}\mathbf{u}^k = \widehat{\mathbf{u}}$. Note that we have previously shown that $\widehat{\mathbf{u}}$ is a primal/dual optimal point of problem (24). We hence conclude the theorem. ■

C. Linear Rate of Convergence

With the strong convexity assumption, we can further guarantee linear convergence rate of the DLADMM algorithm. Before formally stating this result, we first present an implication of Assumption 3.

Lemma 5: If Assumption 3 holds, then f and g are both strongly convex with constant τ.

The strong convexity of f and g implies that there exists a unique primal/dual optimal point $\mathbf{u}^*$ for problem (24). Denote the spectral norm (maximum singular value) of $\mathbf{B}$ as Γ. Now, we are ready to state our second main theorem regarding linear convergence rate. The proof follows analogous idea as the development of linear convergence rate in [17], [18], [16] and is omitted here.

Theorem 2: Suppose Assumptions 2, 3 hold and $c > \max\left\{\frac{L^2}{2\tau}, \rho + \frac{M^2}{2\tau}\right\}$. Then, $\forall k$:

$$\left\|\mathbf{u}^{k+1} - \mathbf{u}^*\right\|_{\mathbf{\Lambda}}^2 \leq \frac{1}{1+\delta}\left\|\mathbf{u}^{k} - \mathbf{u}^*\right\|_{\mathbf{\Lambda}}^2. \tag{60}$$

In (60), $\delta > 0$ is a positive constant defined as:

$$\delta = \min\left\{\frac{\tau - \frac{L^2}{2c}}{\frac{c}{2} + \frac{3\rho\mu\Gamma^2}{\mu-1}}, \frac{\tau - \frac{\beta}{2}}{\frac{c+\rho}{2} + \frac{3\rho\mu}{\mu-1} + \frac{2M^2\mu}{\rho}}, \frac{1}{4}\right\}, \tag{61}$$

where $\beta \in (\frac{M^2}{c-\rho}, 2\tau)$ is the solution of the equation:

$$\frac{\tau - \frac{\beta}{2}}{\frac{c+\rho}{2} + \frac{3\rho\mu}{\mu-1} + \frac{2M^2\mu}{\rho}} = \frac{\frac{c-\rho}{2} - \frac{M^2}{2\beta}}{\frac{3c^2\mu}{\rho(\mu-1)} + \frac{2M^2\mu}{\rho}}, \tag{62}$$

and $\mu > 1$ is any constant greater than 1.

Remark 3: The constant δ determining the convergence rate of the DLADMM depends on the local cost functions (L, M, τ), the network topology (Γ) as well as the algorithm parameters (ρ, c). This sheds some light on how to tune the parameters to achieve better convergence speed in practice. Furthermore, we note that the Theorem 2 only provides a sufficient condition for linear convergence of the DLADMM. In later numerical experiments, we will see that even when the assumptions of Theorem 2 is violated (e.g., the local cost functions are not strongly convex), the DLADMM algorithm may still converge in linear rate.

IV. Numerical Experiments

In this section, numerical results are presented to corroborate the effectiveness of the proposed DLADMM algorithm. In particular, we consider the problem of distributed logistic regression. Suppose each node i has a training set of q training examples $\{\mathbf{u}_{il}, t_{il}\}_{l=1,\ldots,q}$, where $\mathbf{u}_{il} \in \mathbb{R}^p$ is the input feature vector and $t_{il} \in \{-1, 1\}$ is the corresponding output label. Logistic regression model postulates that, for node i, the probability of the output t_i given the input $\mathbf{u}_i$ is $\Pr(t_i = 1|\mathbf{u}_i) = \frac{1}{1+\exp\left\{-\mathbf{u}_i^{\mathsf{T}}\mathbf{x}_i\right\}}$, where $\mathbf{x}_i$ is the classifier for node i. Our goal is to estimate the classifiers of all nodes and thus, together with a decision threshold, we can achieve a input-output mapping at each node. Moreover, we note that neighbor nodes tend to have similar classifiers. Incorporating this prior knowledge into the maximum likelihood estimator of the logistic regression yields the following optimization problem:

$$\begin{aligned}\text{Minimize}_{\{\mathbf{x}_i\}_{i=1,\ldots,n}} \quad & \sum_{i=1}^{n}\sum_{l=1}^{q} \log\left(1 + \exp\left(-t_{il}\mathbf{u}_{il}^{\mathsf{T}}\mathbf{x}_i\right)\right) \\ & + \beta\sum_{i=1}^{n}\sum_{j\in\Omega_i} \|\mathbf{x}_i - \mathbf{x}_j\|_2^2. \end{aligned} \tag{63}$$

The problem (63) is clearly in the form of (1) with:

$$f_i(\mathbf{x}_i) = \sum_{l=1}^{q} \log\left(1 + \exp\left(-t_{il}\mathbf{u}_{il}^{\mathsf{T}}\mathbf{x}_i\right)\right), \tag{64}$$

$$g_{ij}(\mathbf{x}_i, \mathbf{x}_j) = \beta\|\mathbf{x}_i - \mathbf{x}_j\|_2^2. \tag{65}$$

We note that f_i and g_{ij} are all convex, i.e., they satisfy Assumption 1. In addition, ∇f_i is Lipschitz continuous with constant $\frac{1}{4}\sum_{l=1}^{q}\|\mathbf{u}_{il}\|_2^2$ and ∇g_{ij} is Lipschitz continuous with constant 4β. So, Assumption 2 holds with $L = \max\left\{4\beta, \frac{1}{4}\max_{i=1,\ldots,n}\sum_{l=1}^{q}\|\mathbf{u}_{il}\|_2^2\right\}$. Thus, Theorem 1 can be applied with appropriate algorithm parameters and convergence of the DLADMM algorithm is guaranteed theoretically. Moreover, though neither f_i nor g_{ij} is strongly convex (Assumption 3 does not hold), we can still empirically observe linear convergence of the DLADMM in later experiments.

A. *Comparison between the DLADMM, the DADMM and the DGD*

We first conduct an experiment to compare the performance of the DLADMM, the DADMM, and the distributed gradient descent (DGD), in which each node updates its local variable $\mathbf{x}_i$ according to the local gradient descent step:

$$\begin{aligned}\mathbf{x}_i^{k+1} = \mathbf{x}_i^k - \eta\Bigg[\nabla f_i(\mathbf{x}_i^k) + \sum_{j\in\Omega_i}\Bigg(&\nabla_{\mathbf{x}_i} g_{ij}(\mathbf{x}_i^k, \mathbf{x}_j^k) \\ &+ \nabla_{\mathbf{x}_i} g_{ji}(\mathbf{x}_j^k, \mathbf{x}_i^k)\Bigg)\Bigg],\end{aligned} \tag{66}$$

where $\eta > 0$ is the stepsize. We consider two scenarios: (i) a random network with $n = 10$ nodes; the dimension of each data sample is $p = 2$; and each node has $q = 50$ data samples; (ii) a random network with $n = 30$ nodes; the dimension of each data instance is $p = 5$; and each node has $q = 10$ data samples. The average degree of the network is 2. The ADMM algorithm parameter is set to be $\rho = 50$ and the linearization parameter is $c = 3$ in scenario (i) and $c = 5$ in scenario (ii). The stepsize of the DGD is set to be $\eta = 0.01$. In Fig. 1, we compare the relative errors $\frac{\|\mathbf{x}^k - \mathbf{x}^*\|_2}{\|\mathbf{x}^*\|_2}$ ($\mathbf{x}^*$ is the optimal point of (63) obtained by solving the centralized optimization problem with the CVX package [39], [40]) of the DLADMM, the DADMM and the DGD. We observe that the convergence curve of the DLADMM algorithm is very close to that of the DADMM algorithm in both scenarios. Both the DADMM and the DLADMM converge linearly to the optimal point. However, the computational complexity of the DLADMM is much lower than that of the DADMM. It takes several hours for the DADMM to finish 400 iterations while the DLADMM only needs about 5 seconds to finish the same number of iterations. The reason is that, for each node, each iteration of the DADMM necessitates solving a local optimization problem containing log functions, which must be approximated iteratively. Thus, each iteration of the DADMM is carried out very slowly. On the contrary, each iteration of the DLADMM only involves direct closed-form computations, which can be implemented very quickly. This endows the DLADMM with great computational advantage over the DADMM. Furthermore, we observe that both the DLADMM and the DADMM outperform the DGD in terms of convergence rate. This is unsurprising since primal domain methods, such as DGD, usually converge slower than primal dual methods such as ADMM and its variants.

B. *Impact of Network Topology*

Next, we investigate the impact of network topology on the performance of the DLADMM. We set the total number

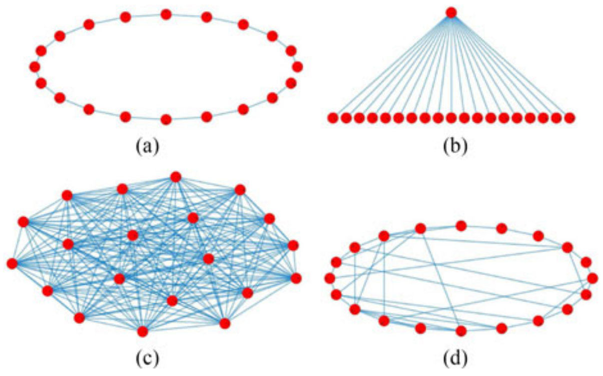


Fig. 2. Different network topologies. (a) Line network. (b) Star network. (c) Complete network. (d) Small-world network.

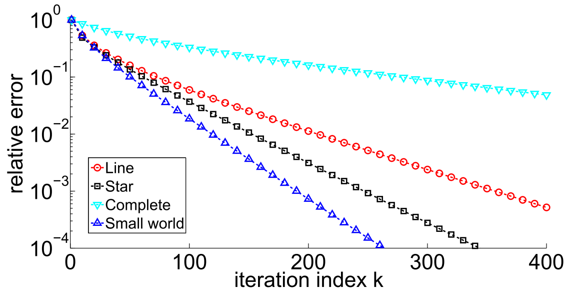


Fig. 3. Performance of the DLADMM on different network topologies.

of nodes to be $n = 20$ and the algorithm parameters to be $\rho = 100, c = 50$. We consider four network topologies: the line network, the star network, the complete network and the small-world network. The four network topologies are illustrated in Fig. 2. To obtain small-world network, we first generate a cycle network, and then add 20 random links between them. As its name suggests, in small-world networks, the distance between two nodes, i.e., the length of the shortest path connecting these two nodes, is small. Many properties of real-world networks can be obtained by the small-world networks [41]. The convergence curves of the DLADMM on different network topologies are shown in Fig. 3. We observe that the convergence of the small-world network and the star network are faster than that of the line network and complete network. The phenomenon can be explained as follows. For the complete network, the number of constraints in the ADMM formulation of the network cost problem (11), i.e., the number of nodes plus the number of links, is large. Thus, the number of dual variables at each node is also large, resulting in slow convergence. For the line network, the distance between nodes is generally large, so that information from a node cannot propagate quickly to many distant nodes. This also prohibits the DLADMM from fast convergence. In contrast, for the star network and the small-world network: (i) the distances between nodes are small so that information can be efficiently diffused; (ii) the average degree of nodes is small so that each node only has a small number of dual variables to update, which can converge quickly. Lastly, we remark that though the DLADMM converges at different speeds for different network topologies, it converges linearly to the optimal point in all circumstances.

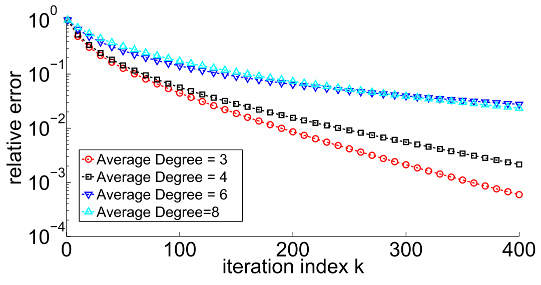


Fig. 4. Performance of the DLADMM on the small-world networks with different average degrees.

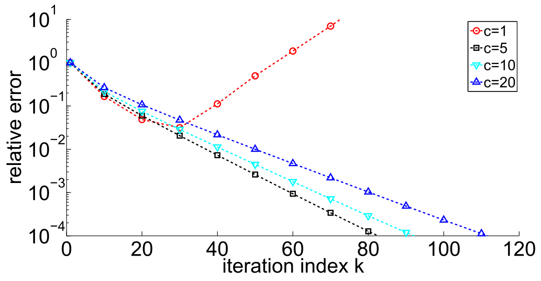


Fig. 5. Performance of the DLADMM with different values of c.

C. Impact of Network Connectivity

We further study the impact of network connectivity, measured by the average node degree, on the performance of the DLADMM over small-world networks. To this end, we first form a cycle network and then add different numbers of random links to obtain small-world networks of different average degrees. The convergence curves of the DLADMM algorithm on small-world networks with different average degrees are reported in Fig. 4. We observe that the small-world networks with smaller average degree have faster convergence speed. The reason is that for small-world networks, even when the average degree is small (e.g., 3), the distances between nodes are short so that information of one node can spread across the network quickly. Additionally, for small-world network with lower degrees, each node only needs to update a small number of dual variables and thus the convergence is faster. Note that when the average degree is high, the small-world networks become analogous to the complete network, over which the DLADMM converges slowly (Fig. 3).

D. Impact of the Linearization Parameter c

Finally, in Fig. 5, we study the impact of the linearization parameter c on the convergence of the DLADMM over small-world networks. We observe that as long as the DLADMM converges, the smaller the value of c , the faster the convergence speed. But c cannot be too small, otherwise the DLADMM may diverge, e.g., when $c = 1$. Recall that the parameter c is introduced to limit the step size between consecutive iterations and therefore plays a similar role as the step size parameter in numerical optimization [30] and adaptive signal processing [35]. A general tradeoff for such parameters is that (i) when they are too large, the convergence is slow; (ii) when they are too small, the algorithm risks divergence. We note that similar phenomenon can be observed in Fig. 5.

V. Conclusion and Future Work

In this paper, we study the generic form of network cost minimization problem, in which the network cost includes both node costs and link costs. The formulated problem has broad applications in distributed signal processing and control over multi-agent networked systems. A distributed linearized ADMM algorithm is presented for the formulated problem. The DLADMM algorithm operates in a decentralized manner and each iteration only involves simple closed-form computations, which endows the DLADMM much lower computational complexity than the distributed ADMM. Under the assumptions that the local cost functions are convex and possess Lipschitz continuous gradients, we show that the DLADMM converges to an optimal point of the network cost minimization problem. By further assuming that the local cost functions are strongly convex, we can guarantee linear convergence rate of the DLADMM. Numerical simulations are carried out to validate the performance of the DLADMM and we empirically observe that the DLADMM has similar convergence performance as DADMM does while the former has much lower computational overhead. The impacts of network topology, connectivity and algorithm parameters on the convergence behaviors of the DLADMM are also discussed.

One possible future direction is to examine the convergence rate of the DLADMM in the absence of strong convexity assumption (Assumption 3). In the literature, it has been shown that ADMM converges to the optimal solution with the rate of $O(1/k)$ even when the objective function is not strongly convex [42]. An analogous convergence rate is expected for the DLADMM in network cost minimization.

Appendix A
Proof of Proposition 1

According to Assumption 1, problem (24) is a convex optimization problem. Thus, the Karush-Kuhn-Tucker (KKT) conditions are necessary and sufficient for optimality. So, the primal/dual optimal point $\mathbf{u}^*$ satisfies the following KKT conditions:

$$\nabla f(\mathbf{x}^*) + \mathbf{B}^\mathsf{T}\boldsymbol{\alpha}^* = \mathbf{0}, \tag{67}$$

$$\nabla g(\mathbf{w}^*) - \boldsymbol{\alpha}^* = \mathbf{0}, \tag{68}$$

$$\mathbf{B}\mathbf{x}^* - \mathbf{w}^* = \mathbf{0}. \tag{69}$$

Subtracting (67) from (28) and exploiting (69) yields gives:

$$\nabla f\left(\mathbf{x}^k\right) - \nabla f(\mathbf{x}^*) + c\left(\mathbf{x}^{k+1} - \mathbf{x}^k\right) + \mathbf{B}^\mathsf{T}\left(\boldsymbol{\alpha}^k - \boldsymbol{\alpha}^*\right) + \rho\mathbf{B}^\mathsf{T}\mathbf{B}\left(\mathbf{x}^{k+1} - \mathbf{x}^*\right) + \rho\mathbf{B}^\mathsf{T}\left(\mathbf{w}^* - \mathbf{w}^k\right) = \mathbf{0}. \tag{70}$$

Similarly, subtracting (68) from (31) and exploiting (69) yields:

$$\nabla g\left(\mathbf{w}^k\right) - \nabla g\left(\mathbf{w}^*\right) + c\left(\mathbf{w}^{k+1} - \mathbf{w}^k\right) + \boldsymbol{\alpha}^* - \boldsymbol{\alpha}^k + \rho\left(\mathbf{w}^{k+1} - \mathbf{w}^*\right) + \rho\mathbf{B}\left(\mathbf{x}^* - \mathbf{x}^{k+1}\right) = \mathbf{0}. \tag{71}$$

Combining (35) and (69) leads to:

$$\boldsymbol{\alpha}^{k+1} - \boldsymbol{\alpha}^k + \rho\mathbf{B}\left(\mathbf{x}^* - \mathbf{x}^{k+1}\right) + \rho\left(\mathbf{w}^{k+1} - \mathbf{w}^*\right) = \mathbf{0}. \tag{72}$$

According to Lemma 3, we have:

$$\frac{1}{L}\left\|\nabla f\left(\mathbf{x}^k\right) - \nabla f(\mathbf{x}^*)\right\|_2^2 \tag{73}$$

$$\leq \left(\mathbf{x}^k - \mathbf{x}^*\right)^\mathsf{T}\left(\nabla f\left(\mathbf{x}^k\right) - \nabla f(\mathbf{x}^*)\right) \tag{74}$$

$$= \left(\mathbf{x}^{k+1} - \mathbf{x}^*\right)^\mathsf{T}\left(\nabla f\left(\mathbf{x}^k\right) - \nabla f(\mathbf{x}^*)\right) + \left(\mathbf{x}^k - \mathbf{x}^{k+1}\right)^\mathsf{T}\left(\nabla f\left(\mathbf{x}^k\right) - \nabla f(\mathbf{x}^*)\right). \tag{75}$$

For the first term of (75), according to (70), we have:

$$\begin{aligned}&\left(\mathbf{x}^{k+1} - \mathbf{x}^*\right)^\mathsf{T}\left(\nabla f\left(\mathbf{x}^k\right) - \nabla f(\mathbf{x}^*)\right)\\ &= \left(\mathbf{x}^{k+1} - \mathbf{x}^*\right)^\mathsf{T}\Big[c\left(\mathbf{x}^k - \mathbf{x}^{k+1}\right) + \mathbf{B}^\mathsf{T}\left(\boldsymbol{\alpha}^* - \boldsymbol{\alpha}^k\right)\\ &\quad + \rho\mathbf{B}^\mathsf{T}\mathbf{B}\left(\mathbf{x}^* - \mathbf{x}^{k+1}\right) + \rho\mathbf{B}^\mathsf{T}\left(\mathbf{w}^{k+1} - \mathbf{w}^*\right)\\ &\quad + \rho\mathbf{B}^\mathsf{T}\left(\mathbf{w}^k - \mathbf{w}^{k+1}\right)\Big].\end{aligned} \tag{76}$$

On the other hand, using Lemma 3 again, we have:

$$\frac{1}{M}\left\|\nabla g\left(\mathbf{w}^k\right) - \nabla g(\mathbf{w}^*)\right\|_2^2 \tag{77}$$

$$\leq \left(\mathbf{w}^k - \mathbf{w}^*\right)^\mathsf{T}\left(\nabla g\left(\mathbf{w}^k\right) - \nabla g(\mathbf{w}^*)\right) \tag{78}$$

$$= \left(\mathbf{w}^{k+1} - \mathbf{w}^*\right)^\mathsf{T}\left(\nabla g\left(\mathbf{w}^k\right) - \nabla g(\mathbf{w}^*)\right) + \left(\mathbf{w}^k - \mathbf{w}^{k+1}\right)^\mathsf{T}\left(\nabla g\left(\mathbf{w}^k\right) - \nabla g(\mathbf{w}^*)\right). \tag{79}$$

For the first term of (79), by using (71), we obtain:

$$\left(\mathbf{w}^{k+1} - \mathbf{w}^*\right)^\mathsf{T}\left(\nabla g\left(\mathbf{w}^k\right) - \nabla g(\mathbf{w}^*)\right) \tag{80}$$

$$= \left(\mathbf{w}^{k+1} - \mathbf{w}^*\right)^\mathsf{T}\Big[c\left(\mathbf{w}^k - \mathbf{w}^{k+1}\right) + \boldsymbol{\alpha}^k - \boldsymbol{\alpha}^* + \rho\left(\mathbf{w}^* - \mathbf{w}^{k+1}\right) + \rho\mathbf{B}\left(\mathbf{x}^{k+1} - \mathbf{x}^*\right)\Big] \tag{81}$$

We note the following fact, which shall be used frequently.

Lemma 6: For any symmetric matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$ and any vectors $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{R}^N$, we have:

$$2(\mathbf{x}-\mathbf{y})^\mathsf{T}\mathbf{A}(\mathbf{z}-\mathbf{x}) = (\mathbf{z}-\mathbf{y})^\mathsf{T}\mathbf{A}(\mathbf{z}-\mathbf{y}) - (\mathbf{x}-\mathbf{y})^\mathsf{T}\mathbf{A}(\mathbf{x}-\mathbf{y}) - (\mathbf{z}-\mathbf{x})^\mathsf{T}\mathbf{A}(\mathbf{z}-\mathbf{x}). \tag{82}$$

Making use of Lemma 6, we obtain:

$$c\left(\mathbf{x}^{k+1} - \mathbf{x}^*\right)^\mathsf{T}\left(\mathbf{x}^k - \mathbf{x}^{k+1}\right) = \frac{c}{2}\left\|\mathbf{x}^k - \mathbf{x}^*\right\|_2^2 - \frac{c}{2}\left\|\mathbf{x}^{k+1} - \mathbf{x}^*\right\|_2^2 - \frac{c}{2}\left\|\mathbf{x}^k - \mathbf{x}^{k+1}\right\|_2^2, \tag{83}$$

and

$$c\left(\mathbf{w}^{k+1} - \mathbf{w}^*\right)^\mathsf{T}\left(\mathbf{w}^k - \mathbf{w}^{k+1}\right) = \frac{c}{2}\left\|\mathbf{w}^k - \mathbf{w}^*\right\|_2^2 - \frac{c}{2}\left\|\mathbf{w}^{k+1} - \mathbf{w}^*\right\|_2^2 - \frac{c}{2}\left\|\mathbf{w}^k - \mathbf{w}^{k+1}\right\|_2^2. \tag{84}$$

Based on (72), we get:

$$\left(\mathbf{x}^{k+1}-\mathbf{x}^*\right)^\mathsf{T}\mathbf{B}^\mathsf{T}\left(\boldsymbol{\alpha}^*-\boldsymbol{\alpha}^k\right)+\left(\mathbf{w}^{k+1}-\mathbf{w}^*\right)^\mathsf{T}\left(\boldsymbol{\alpha}^k-\boldsymbol{\alpha}^*\right)$$
$$=\left[-\mathbf{B}\left(\mathbf{x}^{k+1}-\mathbf{x}^*\right)+\mathbf{w}^{k+1}-\mathbf{w}^*\right]^\mathsf{T}\left(\boldsymbol{\alpha}^k-\boldsymbol{\alpha}^*\right) \quad (85)$$
$$=\frac{1}{\rho}\left(\boldsymbol{\alpha}^k-\boldsymbol{\alpha}^{k+1}\right)^\mathsf{T}\left(\boldsymbol{\alpha}^k-\boldsymbol{\alpha}^*\right) \quad (86)$$
$$=-\frac{1}{2\rho}\left\|\boldsymbol{\alpha}^*-\boldsymbol{\alpha}^{k+1}\right\|_2^2+\frac{1}{2\rho}\left\|\boldsymbol{\alpha}^k-\boldsymbol{\alpha}^{k+1}\right\|_2^2+\frac{1}{2\rho}\left\|\boldsymbol{\alpha}^*-\boldsymbol{\alpha}^k\right\|_2^2. \quad (87)$$

Again, using (72), we obtain:

$$\left(\mathbf{x}^{k+1}-\mathbf{x}^*\right)^\mathsf{T}\left[\rho\mathbf{B}^\mathsf{T}\mathbf{B}\left(\mathbf{x}^*-\mathbf{x}^{k+1}\right)+\rho\mathbf{B}^\mathsf{T}\left(\mathbf{w}^{k+1}-\mathbf{w}^*\right)\right]+\left(\mathbf{w}^{k+1}-\mathbf{w}^*\right)^\mathsf{T}\left[\rho\left(\mathbf{w}^*-\mathbf{w}^{k+1}\right)+\rho\mathbf{B}\left(\mathbf{x}^{k+1}-\mathbf{x}^*\right)\right] \quad (88)$$
$$=-\rho\left\|\mathbf{B}\left(\mathbf{x}^{k+1}-\mathbf{x}^*\right)+\mathbf{w}^*-\mathbf{w}^{k+1}\right\|_2^2 \quad (89)$$
$$=-\frac{1}{\rho}\left\|\boldsymbol{\alpha}^{k+1}-\boldsymbol{\alpha}^k\right\|_2^2. \quad (90)$$

Once again, using (72), we have:

$$\rho\left(\mathbf{x}^{k+1}-\mathbf{x}^*\right)^\mathsf{T}\mathbf{B}^\mathsf{T}\left(\mathbf{w}^k-\mathbf{w}^{k+1}\right) \quad (91)$$
$$=\left[\boldsymbol{\alpha}^{k+1}-\boldsymbol{\alpha}^k+\rho\left(\mathbf{w}^{k+1}-\mathbf{w}^*\right)\right]^\mathsf{T}\left(\mathbf{w}^k-\mathbf{w}^{k+1}\right) \quad (92)$$
$$=\left(\boldsymbol{\alpha}^{k+1}-\boldsymbol{\alpha}^k\right)^\mathsf{T}\left(\mathbf{w}^k-\mathbf{w}^{k+1}\right)+\frac{\rho}{2}\left\|\mathbf{w}^k-\mathbf{w}^*\right\|_2^2-\frac{\rho}{2}\left\|\mathbf{w}^{k+1}-\mathbf{w}^*\right\|_2^2-\frac{\rho}{2}\left\|\mathbf{w}^k-\mathbf{w}^{k+1}\right\|_2^2 \quad (93)$$
$$\le\frac{1}{2}\left\|\frac{1}{\sqrt{2\rho}}\left(\boldsymbol{\alpha}^{k+1}-\boldsymbol{\alpha}^k\right)\right\|_2^2+\frac{1}{2}\left\|\sqrt{2\rho}\left(\mathbf{w}^k-\mathbf{w}^{k+1}\right)\right\|_2^2+\frac{\rho}{2}\left\|\mathbf{w}^k-\mathbf{w}^*\right\|_2^2-\frac{\rho}{2}\left\|\mathbf{w}^{k+1}-\mathbf{w}^*\right\|_2^2-\frac{\rho}{2}\left\|\mathbf{w}^k-\mathbf{w}^{k+1}\right\|_2^2 \quad (94)$$
$$=\frac{1}{4\rho}\left\|\boldsymbol{\alpha}^{k+1}-\boldsymbol{\alpha}^k\right\|_2^2+\frac{\rho}{2}\left\|\mathbf{w}^k-\mathbf{w}^{k+1}\right\|_2^2+\frac{\rho}{2}\left\|\mathbf{w}^k-\mathbf{w}^*\right\|_2^2-\frac{\rho}{2}\left\|\mathbf{w}^{k+1}-\mathbf{w}^*\right\|_2^2. \quad (95)$$

Adding the results in (83), (84), (87), (90) and (95) all together and noting (76) and (81), we get:

$$\left(\mathbf{x}^{k+1}-\mathbf{x}^*\right)^\mathsf{T}\left(\nabla f\left(\mathbf{x}^k\right)-\nabla f(\mathbf{x}^*)\right)+\left(\mathbf{w}^{k+1}-\mathbf{w}^*\right)^\mathsf{T}\left(\nabla g\left(\mathbf{w}^k\right)-\nabla g(\mathbf{w}^*)\right) \quad (96)$$
$$\le\frac{c}{2}\left\|\mathbf{x}^k-\mathbf{x}^*\right\|_2^2-\frac{c}{2}\left\|\mathbf{x}^{k+1}-\mathbf{x}^*\right\|_2^2-\frac{c}{2}\left\|\mathbf{x}^k-\mathbf{x}^{k+1}\right\|_2^2+\frac{c}{2}\left\|\mathbf{w}^k-\mathbf{w}^*\right\|_2^2-\frac{c}{2}\left\|\mathbf{w}^{k+1}-\mathbf{w}^*\right\|-\frac{c}{2}\left\|\mathbf{w}^k-\mathbf{w}^{k+1}\right\|_2^2$$
$$-\frac{1}{2\rho}\left\|\boldsymbol{\alpha}^*-\boldsymbol{\alpha}^{k+1}\right\|_2^2+\frac{1}{2\rho}\left\|\boldsymbol{\alpha}^*-\boldsymbol{\alpha}^k\right\|_2^2-\frac{1}{4\rho}\left\|\boldsymbol{\alpha}^{k+1}-\boldsymbol{\alpha}^k\right\|_2^2+\frac{\rho}{2}\left\|\mathbf{w}^k-\mathbf{w}^*\right\|_2^2-\frac{\rho}{2}\left\|\mathbf{w}^{k+1}-\mathbf{w}^*\right\|_2^2+\frac{\rho}{2}\left\|\mathbf{w}^k-\mathbf{w}^{k+1}\right\|_2^2 \quad (97)$$
$$=\left\|\mathbf{u}^k-\mathbf{u}^*\right\|_{\boldsymbol{\Lambda}}^2-\left\|\mathbf{u}^{k+1}-\mathbf{u}^*\right\|_{\boldsymbol{\Lambda}}^2-\frac{c}{2}\left\|\mathbf{x}^k-\mathbf{x}^{k+1}\right\|_2^2-\frac{c-\rho}{2}\left\|\mathbf{w}^k-\mathbf{w}^{k+1}\right\|_2^2-\frac{1}{4\rho}\left\|\boldsymbol{\alpha}^{k+1}-\boldsymbol{\alpha}^k\right\|_2^2. \quad (98)$$

Combining (75), (79) and (98), we obtain:

$$\frac{1}{L}\left\|\nabla f\left(\mathbf{x}^k\right)-\nabla f(\mathbf{x}^*)\right\|_2^2+\frac{1}{M}\left\|\nabla g\left(\mathbf{w}^k\right)-\nabla g(\mathbf{w}^*)\right\|_2^2$$
$$\le\left\|\mathbf{u}^k-\mathbf{u}^*\right\|_{\boldsymbol{\Lambda}}^2-\left\|\mathbf{u}^{k+1}-\mathbf{u}^*\right\|_{\boldsymbol{\Lambda}}^2-\frac{c}{2}\left\|\mathbf{x}^k-\mathbf{x}^{k+1}\right\|_2^2-\frac{c-\rho}{2}\left\|\mathbf{w}^k-\mathbf{w}^{k+1}\right\|_2^2-\frac{1}{4\rho}\left\|\boldsymbol{\alpha}^{k+1}-\boldsymbol{\alpha}^k\right\|_2^2$$
$$+\left(\mathbf{x}^k-\mathbf{x}^{k+1}\right)^\mathsf{T}\left(\nabla f\left(\mathbf{x}^k\right)-\nabla f(\mathbf{x}^*)\right)+\left(\mathbf{w}^k-\mathbf{w}^{k+1}\right)^\mathsf{T}\left(\nabla g\left(\mathbf{w}^k\right)-\nabla g(\mathbf{w}^*)\right) \quad (99)$$
$$\le\left\|\mathbf{u}^k-\mathbf{u}^*\right\|_{\boldsymbol{\Lambda}}^2-\left\|\mathbf{u}^{k+1}-\mathbf{u}^*\right\|_{\boldsymbol{\Lambda}}^2-\frac{c}{2}\left\|\mathbf{x}^k-\mathbf{x}^{k+1}\right\|_2^2-\frac{c-\rho}{2}\left\|\mathbf{w}^k-\mathbf{w}^{k+1}\right\|_2^2-\frac{1}{4\rho}\left\|\boldsymbol{\alpha}^{k+1}-\boldsymbol{\alpha}^k\right\|_2^2$$
$$+\frac{L}{4}\left\|\mathbf{x}^k-\mathbf{x}^{k+1}\right\|_2^2+\frac{1}{L}\left\|\nabla f\left(\mathbf{x}^k\right)-\nabla f(\mathbf{x}^*)\right\|_2^2+\frac{M}{4}\left\|\mathbf{w}^k-\mathbf{w}^{k+1}\right\|_2^2+\frac{1}{M}\left\|\nabla g\left(\mathbf{w}^k\right)-\nabla g(\mathbf{w}^*)\right\|_2^2 \quad (100)$$

We can rewrite (100) as:

$$\left\|\mathbf{u}^{k+1}-\mathbf{u}^*\right\|_{\boldsymbol{\Lambda}}^2\le\left\|\mathbf{u}^k-\mathbf{u}^*\right\|_{\boldsymbol{\Lambda}}^2-\left(\frac{c}{2}-\frac{L}{4}\right)\left\|\mathbf{x}^k-\mathbf{x}^{k+1}\right\|_2^2-\left(\frac{c-\rho}{2}-\frac{M}{4}\right)\left\|\mathbf{w}^k-\mathbf{w}^{k+1}\right\|_2^2-\frac{1}{4\rho}\left\|\boldsymbol{\alpha}^{k+1}-\boldsymbol{\alpha}^k\right\|_2^2 \quad (101)$$

References

[1] A. H. Sayed, "Adaptive networks," *Proc. IEEE*, vol. 102, no. 4, pp. 460–497, Apr. 2014.

[2] A. G. Dimakis, S. Kar, J. M. Moura, M. G. Rabbat, and A. Scaglione, "Gossip algorithms for distributed signal processing," *Proc. IEEE*, vol. 98, no. 11, pp. 1847–1864, Nov. 2010.

[3] J. B. Predd, S. R. Kulkarni, and H. V. Poor, "A collaborative training algorithm for distributed learning," *IEEE Trans. Inf. Theory*, vol. 55, no. 4, pp. 1856–1871, Apr. 2009.

[4] Y. Guo, L. Tong, W. Wu, H. Sun, and B. Zhang, "Hierarchical multi-area state estimation via sensitivity function exchanges," *IEEE Trans. Power Syst.*, vol. 32, no. 1, pp. 442–453, Jan. 2017.

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

[6] C. Shen, T.-H. Chang, K.-Y. Wang, Z. Qiu, and C.-Y. Chi, "Distributed robust multicell coordinated beamforming with imperfect CSI: An ADMM approach," *IEEE Trans. Signal Process.*, vol. 60, no. 6, pp. 2988–3003, Jun. 2012.

[7] J. Huang, R. A. Berry, and M. L. Honig, "Distributed interference compensation for wireless networks," *IEEE J. Select. Areas Commun.*, vol. 24, no. 5, pp. 1074–1084, May 2006.

[8] E. Wei, A. Ozdaglar, and A. Jadbabaie, "A distributed newton method for network utility maximization-I: Algorithm," *IEEE Trans. Automat. Control*, vol. 58, no. 9, pp. 2162–2175, Sep. 2013.

[9] E. Wei, A. Ozdaglar, and A. Jadbabaie, "A distributed newton method for network utility maximization part II: Convergence," *IEEE Trans. Automat. Control*, vol. 58, no. 9, pp. 2176–2188, Sep. 2013.

[10] J. Zhang, D. Zheng, and M. Chiang, "The impact of stochastic noisy feedback on distributed network utility maximization," *IEEE Trans. Inf. Theory*, vol. 54, no. 2, pp. 645–665, Feb. 2008.

[11] D. Niu and B. Li, "An asynchronous fixed-point algorithm for resource sharing with coupled objectives," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2593–2606, Oct. 2016.

[12] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Trans. Automat. Control*, vol. 54, no. 1, pp. 48–61, Jan. 2009.

[13] J. C. Duchi, A. Agarwal, and M. J. Wainwright, "Dual averaging for distributed optimization: Convergence analysis and network scaling," *IEEE Trans. Automat. control*, vol. 57, no. 3, pp. 592–606, Mar. 2012.

[14] T. Erseghe, D. Zennaro, E. Dall'Anese, and L. Vangelista, "Fast consensus by the alternating direction multipliers method," *IEEE Trans. Signal Process.*, vol. 59, no. 11, pp. 5523–5537, Nov. 2011.

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

[16] 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.

[17] Q. Ling, W. Shi, G. Wu, and A. Ribeiro, "DLM: Decentralized linearized alternating direction method of multipliers," *IEEE Trans. Signal Process.*, vol. 63, no. 15, pp. 4051–4064, Aug. 2015.

[18] A. Mokhtari, W. Shi, Q. Ling, and A. Ribeiro, "DQM: Decentralized quadratically approximated alternating direction method of multipliers," *IEEE Trans. Signal Process.*, vol. 64, no. 19, pp. 5158–5173, Oct. 2016.

[19] 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.

[20] A. Makhdoumi and A. Ozdaglar, "Convergence rate of distributed ADMM over networks," *IEEE Trans. Automat. Control*, vol. 62, no. 10, pp. 5082–5095, Oct. 2017.

[21] T.-H. Chang, M. Hong, W.-C. Liao, and X. Wang, "Asynchronous distributed ADMM for large-scale optimization part I: Algorithm and convergence analysis," *IEEE Trans. Signal Process.*, vol. 64, no. 12, pp. 3118–3130, Jun. 2016.

[22] S. Kumar, R. Jain, and K. Rajawat, "Asynchronous optimization over heterogeneous networks via consensus ADMM," *IEEE Trans. Signal Inf. Process. over Netw.*, vol. 3, no. 1, pp. 114–129, Mar. 2017.

[23] Q. Ling and A. Ribeiro, "Decentralized dynamic optimization through the alternating direction method of multipliers," *IEEE Trans. Signal Process.*, vol. 62, no. 5, pp. 1185–1197, Mar. 2014.

[24] M. Hong, Z.-Q. Luo, and M. Razaviyayn, "Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems," *SIAM J. Optim.*, vol. 26, no. 1, pp. 337–364, 2016.

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

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

[27] J. Chen, C. Richard, and A. H. Sayed, "Multitask diffusion adaptation over networks," *IEEE Trans. Signal Process.*, vol. 62, no. 16, pp. 4129–4144, Aug. 2014.

[28] 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.

[29] S. Monajemi, K. Eftaxias, S. Sanei, and S.-H. Ong, "An informed multitask diffusion adaptation approach to study tremor in parkinson's disease," *IEEE J. Select. Topics Signal Process.*, vol. 10, no. 7, pp. 1306–1314, Oct. 2016.

[30] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[31] W. Deng and W. Yin, "On the global and linear convergence of the generalized alternating direction method of multipliers," *J. Sci. Comput.*, vol. 66, no. 3, pp. 889–916, 2016.

[32] J. Zhang, S. Nabavi, A. Chakrabortty, and Y. Xin, "ADMM optimization strategies for wide-area oscillation monitoring in power systems under asynchronous communication delays," *IEEE Trans. Smart Grid*, vol. 7, no. 4, pp. 2123–2133, Jul. 2016.

[33] T.-H. Chang, "A proximal dual consensus ADMM method for multi-agent constrained optimization," *IEEE Trans. Signal Process.*, vol. 64, no. 14, pp. 3719–3734, Jul. 2016.

[34] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation: numerical methods*, vol. 23, Englewood Cliffs, NJ, USA: Prentice-Hall, 1989.

[35] S. Haykin, *Adaptive Filter Theory,*. 3rd ed Upper Saddle River, NJ, USA: Prentice-Hall, 1996.

[36] N. Parikh *et al.*, "Proximal algorithms," *Found. Trends Optim.*, vol. 1, no. 3, pp. 127–239, 2014.

[37] X. Gao, B. Jiang, and S. Zhang, "On the information-adaptive variants of the ADMM: an iteration complexity perspective," *Optim. Online*, vol. 12, pp. 1–37, 2014.

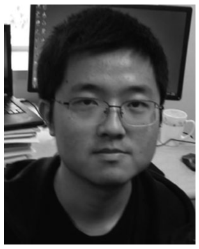
[38] L. Vandenberghe, "Gradient method," *lecture notes of EE236C*, Los Angeles, CA, USA: Univ. California, Los Angeles, 2016.

[39] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 2.1.," Mar. 2014, [Online]. Available: http://cvxr.com/cvx

[40] M. Grant and S. Boyd, "Graph implementations for nonsmooth convex programs," in *Recent Advances in Learning and Control*, (Lecture Notes in Control and Information Sciences), V. Blondel, S. Boyd, and H. Kimura, Eds., New York, NY, USA: Springer, 2008, pp. 95–110.

[41] D. J. Watts and S. H. Strogatz, "Collective dynamics of small-world networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[42] E. Wei and A. Ozdaglar, "Distributed alternating direction method of multipliers," in *Proc. 2012 IEEE 51st Annu. Conf. Decision Control*, 2012, pp. 5445–5450.



Xuanyu Cao received the Bachelor's degree from Shanghai Jiao Tong University, Shanghai, China, in 2013, and the M.S. and Ph.D. degrees from the University of Maryland, College Park, MD, USA, in 2016 and 2017, respectively, all in electrical engineering. Since 2017, he has been a Postdoctoral Research Associate with the Department of Electrical Engineering, Princeton University, Princeton, NJ, USA. His research interests include the mechanism design, optimization, game theory, signal processing, probabilistic methods, with applications to information systems and networks.

K. J. Ray Liu (F'03) was a Distinguished Scholar-Teacher with the University of Maryland, College Park, MD 20742 USA, in 2007, where he is a Christine Kim Eminent Professor of Information Technology. He leads the Maryland Signals and Information Group conducting research encompassing broad areas of information and communications technology with recent focus on smart radios for smart life.

He was a recipient of the 2016 IEEE Leon K. Kirchmayer Award on graduate teaching and mentoring, IEEE Signal Processing Society 2014 Award, IEEE Signal Processing Society 2009 Technical Achievement Award, and over a dozen of best paper awards. Recognized by Web of Science as a Highly Cited Researcher, he is a Fellow of AAAS. His invention of Time-Reversal Machine by Origin Wireless Inc., won the 2017 CEATEC Grand Prix Award.

He is the IEEE Vice President, Technical Activities – Elect. He was the President of IEEE Signal Processing Society, where he has served as the Vice President Publications and Board of Governor, and a member of IEEE Board of Director as Division IX Director. He has also served as the Editor-in-Chief of IEEE Signal Processing Magazine.

He also received teaching and research recognitions from University of Maryland including university-level Invention of the Year Award; and college-level Poole and Kent Senior Faculty Teaching Award, Outstanding Faculty Research Award, and Outstanding Faculty Service Award, all from A. James Clark School of Engineering.