Distributed Efficient Optimization for General Network Cost Minimization Problems

Xuanyu Cao and K. J. Ray Liu

***Abstract*—In this work, 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. To obtain a decentralized algorithm for this 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, we propose a distributed linearized ADMM (DLADMM) algorithm, in which each iteration only involves closed-form computations and avoids local optimization problems. This greatly reduces the computational complexity and makes the proposed DLADMM amenable to devices with low computational capability, such as vastly deployed sensors, to which the computationally intensive traditional DADMM is not applicable. 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.**

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] and decentralized power system monitoring [4] as well as signal processing for communication networks [5]. 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. Thus, 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, e.g., the link capacity constraint in communication networks [6]–[8]. 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 [9]–[13].

In most existing works on network optimization, only costs or gains at nodes are considered while the costs or gains of links are ignored. For example, in consensus optimization, the network cost is only comprised of local cost at each node and the effect of the link costs is not incorporated. 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 [14], 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 proximity 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). By increasing the weights of these link costs, we recover the consensus optimization problem widely examined in [11]–[13] as a special case.

In this paper, we study the network cost minimization problem, where the network cost encompasses both node costs and link costs. To obtain a distributed algorithm for the problem, we resort to the distributed alternating direction method of multipliers (DADMM) [15], which generally converges faster than distributed subgradient method [9]. 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 approximated ADMM for consensus optimization [11], [16], 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, i.e., first order approximations. This significantly reduces the computational complexity and makes the proposed DLADMM amenable to devices with low computational capability, such as cheap sensors vastly deployed, to which traditional computationally intensive DADMM is not applicable. Under certain technical assumptions, we demonstrate that the DLADMM algorithm is guaranteed to converge to an optimal point and indeed has linear convergence rate. We note that an analogous DLADMM algorithm has been proposed in [11] 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. 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 presented. In SectionIV, numerical simulations are conducted. In Section V, we conclude this work.

II. Problem Statement and Algorithm Development

In this section, we first 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 developed. Finally, to reduce the computational burden of the DADMM, we propose a distributed linearized ADMM (DLADMM) algorithm.

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 [11] 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. For instance, 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 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., $\|\mathbf{x}_i-\mathbf{x}_j\|_2^2$ [14]. In addition, in many network resource allocation problems, the notions of node costs and link costs also arise. For ease of reference, we define the following assumptions, all of which are standard in the literature of numerical optimization [10].

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) $\forall i, \mathbf{x}_i, \mathbf{x}_i' \in \mathbb{R}^p$:*

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

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

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

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 $i=1,...,n$:*

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

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

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

B. Preliminaries of ADMM

ADMM is an optimization framework widely applied to various signal processing and control applications [4], [5]. It enjoys fast convergence speed under mild technical conditions [17] and is especially suitable for the development of distributed algorithms [15]. ADMM solves problems of the following form:

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

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:

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

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 [15]. It is recently shown that global linear convergence can be ensured provided additional assumptions on problem (6) holds [17].

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:

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

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

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

Further introducing Lagrange multipliers $\boldsymbol{\lambda}_i, \boldsymbol{\mu}_{ij} \in \mathbb{R}^p, \forall i = 1, ..., n, j \in \Omega_i$ associated with constraints (12) and (13), respectively, we can form the augmented Lagrangian of the above optimization problem and derive the correspnding ADMM updates based on (8), (9) and (10) as follows: $\forall i, j \in \Omega_i$,

$$\mathbf{x}_i^{k+1} = \arg\min_{\mathbf{x}_i} \Bigg\{ f_i(\mathbf{x}_i) + \boldsymbol{\lambda}_i^{k\mathsf{T}}\mathbf{x}_i + \sum_{l\in\Omega_i} \boldsymbol{\mu}_{li}^{k\mathsf{T}}\mathbf{x}_i + \frac{\rho}{2}\left\|\mathbf{x}_i - \mathbf{y}_i^k\right\|_2^2 + \frac{\rho}{2}\sum_{l\in\Omega_i}\left\|\mathbf{x}_i - \mathbf{z}_{li}^k\right\|_2^2 \Bigg\}. \tag{14}$$

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

$$\boldsymbol{\lambda}_i^{k+1} = \boldsymbol{\lambda}_i^k + \rho\left(\mathbf{x}_i^{k+1} - \mathbf{y}_i^{k+1}\right), \tag{16}$$

$$\boldsymbol{\mu}_{ij}^{k+1} = \boldsymbol{\mu}_{ij}^k + \rho\left(\mathbf{x}_j^{k+1} - \mathbf{z}_{ij}^{k+1}\right). \tag{17}$$

Equations (14), (15), (16) and (17) 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 $j \in \Omega_i$.

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 (14) 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 (15) 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 (16) and (17), respectively. Transmit $\boldsymbol{\mu}_{ij}^{k+1}$ to the neighbor node $j$ for each $j \in \Omega_i$.
6:     $k \leftarrow k + 1$.
```

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 (14) and (15), 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 $g(\mathbf{y}, \mathbf{z}) = \sum_{i=1}^n \sum_{j\in\Omega_i} g_{ij}(\mathbf{y}_i, \mathbf{z}_{ij})$, where $\mathbf{x}$ is the concatenation of all $\mathbf{x}_i$'s and $\mathbf{y}, \mathbf{z}$ are similarly defined. 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 $\mathbf{0}_{p\times p}$. 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, (11) can be rewritten as:

$$\text{Minimize } \; f(\mathbf{x}) + g(\mathbf{w}) \tag{18}$$

$$\text{s.t. } \; \mathbf{B}\mathbf{x} - \mathbf{w} = \mathbf{0}. \tag{19}$$

The augmented Lagrangian can be written as:

$$\mathfrak{L}_\rho(\mathbf{x}, \mathbf{w}, \boldsymbol{\alpha}) = f(\mathbf{x}) + g(\mathbf{w}) + \alpha^\mathsf{T}(\mathbf{B}\mathbf{x} - \mathbf{w}) + \frac{\rho}{2}\|\mathbf{B}\mathbf{x} - \mathbf{w}\|_2^2, \tag{20}$$

where $\boldsymbol{\alpha} = \left[\boldsymbol{\lambda}^\mathsf{T}, \boldsymbol{\mu}^\mathsf{T}\right]^\mathsf{T}$ is the Lagrange multiplier. The original DADMM algorithm necessitates solving local optimization problems involving f and g. To avoid this burden, 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:

$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \Bigg\{ \nabla f\left(\mathbf{x}^k\right)^\mathsf{T}\left(\mathbf{x} - \mathbf{x}^k\right) + \frac{c}{2}\left\|\mathbf{x} - \mathbf{x}^k\right\|_2^2 + \boldsymbol{\alpha}^{k\mathsf{T}}\mathbf{B}\mathbf{x} + \frac{\rho}{2}\left\|\mathbf{B}\mathbf{x} - \mathbf{w}^k\right\|_2^2 \Bigg\}, \tag{21}$$

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 $\mathbf{x}^k$. The optimization problem (21) can be decomposed across nodes and be solved in closed form:

$$\mathbf{x}_i^{k+1} = \frac{1}{c+\rho+\rho|\Omega_i|}\Bigg[-\nabla f_i\left(\mathbf{x}_i^k\right)+c\mathbf{x}_i^k-\lambda_i^k-\sum_{l\in\Omega_i}\boldsymbol{\mu}_{li}^k+ \rho\mathbf{y}_i^k+\rho\sum_{l\in\Omega_i}\mathbf{z}_{li}^k\Bigg]. \tag{22}$$

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

$$\mathbf{w}^{k+1} = \arg\min_{\mathbf{w}}\Bigg\{\nabla g\left(\mathbf{w}^k\right)^\mathsf{T}\left(\mathbf{w}-\mathbf{w}^k\right)+\frac{c}{2}\left\|\mathbf{w}-\mathbf{w}^k\right\|_2^2 - \boldsymbol{\alpha}^{k\mathsf{T}}\mathbf{w}+\frac{\rho}{2}\left\|\mathbf{w}-\mathbf{B}\mathbf{x}^{k+1}\right\|_2^2\Bigg\}, \tag{23}$$

The optimization problem (23) can be decomposed across nodes and be solved in closed-form:

$$\mathbf{y}_i^{k+1} = \frac{1}{c+\rho}\Bigg[-\sum_{j\in\Omega_i}\nabla_{\mathbf{y}_i}g_{ij}\left(\mathbf{y}_i^k,\mathbf{z}_{ij}^k\right)+c\mathbf{y}_i^k+\boldsymbol{\lambda}_i^k+\rho\mathbf{x}_i^{k+1}\Bigg], \tag{24}$$

$$\mathbf{z}_{ij}^{k+1} = \frac{1}{c+\rho}\left[-\nabla_{\mathbf{z}_{ij}}g_{ij}\left(\mathbf{y}_i^k,\mathbf{z}_{ij}^k\right)+c\mathbf{z}_{ij}^k+\boldsymbol{\mu}_{ij}^k+\rho\mathbf{x}_j^{k+1}\right]. \tag{25}$$

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

$$\boldsymbol{\alpha}^{k+1} = \boldsymbol{\alpha}^k+\rho\left(\mathbf{B}\mathbf{x}^{k+1}-\mathbf{w}^{k+1}\right), \tag{26}$$

which can be implemented in a decentralized manner as in (16) and (17). In other words, the update of the dual variables in DLADMM is the same as that of DADMM. Combining (22), (24), (25), (16) and (17) 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.

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 (22) 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 (24) and
       (25), 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 (16) and
       (17), respectively. Transmit $\boldsymbol{\mu}_{ij}^{k+1}$ to the neighbor node $j$
       for each $j\in\Omega_i$.
6:     $k\leftarrow k+1$.
```

III. Convergence Guarantees

In this section, we present theoretical results regarding the convergence behaviors of the proposed DLADMM algorithm, i.e., Algorithm 2, for the network cost minimization problem (1). We demonstrate the convergence of the DLADMM in Theorem 1 and a linear convergence rate of the DLADMM in Theorem 2.

We define a diagonal positive definite matrix $\boldsymbol{\Lambda}$ as $\boldsymbol{\Lambda} = \mathrm{diag}(\frac{c}{2}\mathbf{I}_{np}, \frac{\rho+c}{2}\mathbf{I}_{np+mp}, \frac{1}{2\rho}\mathbf{I}_{np+mp})$. 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 theorem regarding the convergence of the proposed DLADMM.

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* (18)*, i.e., there exists a primal/dual optimal point of problem* (18) $\mathbf{u}^*$ *such that* $\lim_{k\to\infty}\mathbf{u}^k=\mathbf{u}^*$.

With the strong convexity assumption, we can further guarantee linear convergence rate of the DLADMM algorithm. The strong convexity of f and g implies that there exists a unique primal/dual optimal point $\mathbf{u}^*$ for problem (18). 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.

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

$$\left\|\mathbf{u}^{k+1}-\mathbf{u}^*\right\|_{\boldsymbol{\Lambda}}^2 \le \frac{1}{1+\delta}\left\|\mathbf{u}^k-\mathbf{u}^*\right\|_{\boldsymbol{\Lambda}}^2. \tag{27}$$

In (27)*,* $\delta > 0$ *is a positive constant defined as:*

$$\delta = \min\left\{\frac{\tau-\frac{L^2}{2c}}{\frac{c}{2}+\frac{3\rho\mu\Gamma^2}{\mu-1}}, \frac{\tau-\frac{\beta}{2}}{\frac{c+\rho}{2}+\frac{3\rho\mu}{\mu-1}+\frac{2M^2\mu}{\rho}}, \frac{1}{4}\right\}, \tag{28}$$

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

$$\frac{\tau-\frac{\beta}{2}}{\frac{c+\rho}{2}+\frac{3\rho\mu}{\mu-1}+\frac{2M^2\mu}{\rho}} = \frac{\frac{c-\rho}{2}-\frac{M^2}{2\beta}}{\frac{3c^2\mu}{\rho(\mu-1)}+\frac{2M^2\mu}{\rho}}, \tag{29}$$

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

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 $\mathbf{u}_i$

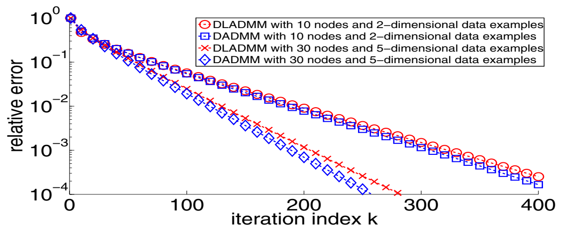
is $\Pr(t_i = 1|\mathbf{u}_i) = \frac{1}{1+\exp\{-\mathbf{u}_i^\mathsf{T}\mathbf{x}_i\}}$, where $\mathbf{x}_i$ is the classifier for node i. Our goal is to estimate the classifiers of all nodes and thus, together with a decision threshold, we can achieve a input-output mapping at each node. Moreover, we note that neighbor nodes tend to have similar classifiers. Incorporating this prior knowledge into the maximum likelihood estimator of the logistic regression yields:

$$\begin{aligned} \text{Minimize}_{\{\mathbf{x}_i\}_{i=1,\ldots,n}} \quad & \sum_{i=1}^{n}\sum_{l=1}^{q} \log\left(1+\exp\left(-t_{il}\mathbf{u}_{il}^\mathsf{T}\mathbf{x}_i\right)\right) \\ & + \beta \sum_{i=1}^{n}\sum_{j\in\Omega_i} \|\mathbf{x}_i - \mathbf{x}_j\|_2^2. \end{aligned} \tag{30}$$

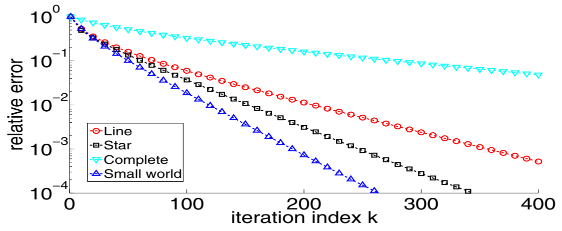
The problem (30) is clearly in the form of (1) with: $f_i(\mathbf{x}_i) = \sum_{l=1}^{q} \log\left(1+\exp\left(-t_{il}\mathbf{u}_{il}^\mathsf{T}\mathbf{x}_i\right)\right)$ and $g_{ij}(\mathbf{x}_i, \mathbf{x}_j) = \beta\|\mathbf{x}_i - \mathbf{x}_j\|_2^2$. We note that f_i and g_{ij} are all convex, 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. 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.

We first conduct an experiment to compare the performance of the DLADMM and the DADMM algorithms. 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). In Fig. 1-(a), we compare the relative errors $\frac{\|\mathbf{x}^k - \mathbf{x}^*\|_2}{\|\mathbf{x}^*\|_2}$ ($\mathbf{x}^*$ is the optimal point of (30) obtained by solving the centralized optimization problem with the CVX package [18]) of the DADMM algorithm and the DLADMM algorithm. 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. Actually, it takes more than 5 hours for the DADMM to finish 400 iterations while the DLADMM only needs about 5 seconds.

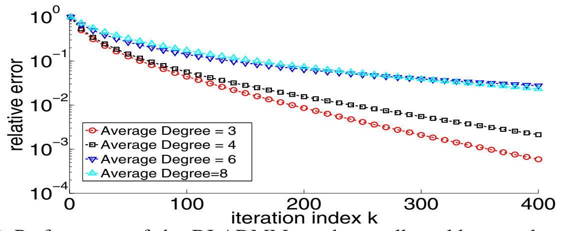
Next, we investigate the impact of network topology on the performance of the DLADMM. We set the total number of nodes to be $n = 20$ and the algorithm parameters to be $\rho = 100, c = 50$. We consider four network topologies: the line network, the star network, the complete network and the small-world 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 [19]. The convergence curves of the DLADMM on different network topologies are shown in Fig. 1-(b). 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



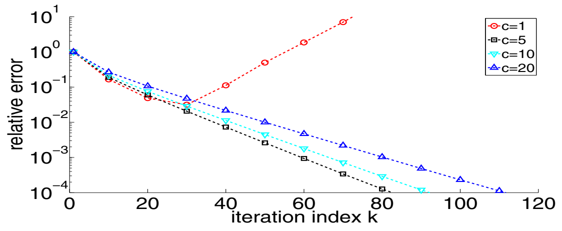
(a) Comparison between the DLADMM and DADMM



(b) Performance of the DLADMM on different network topologies

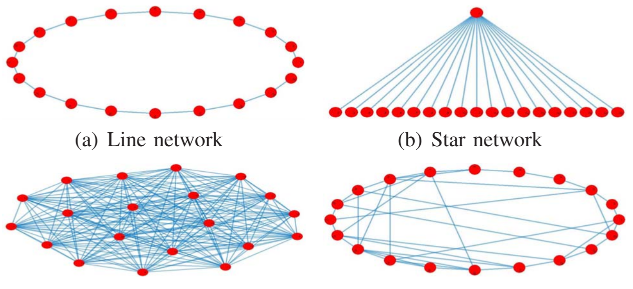


(c) Performance of the DLADMM on the small-world networks with different average degrees



(d) Performance of the DLADMM with different values of c

Fig. 1: Experimental results



(a) Line network (b) Star network (c) Complete network (d) Small-world network

Fig. 2: Different network topologies

remark that though the DLADMM converges at different speeds for different network topologies, it converges linearly to the optimal point in all circumstances.

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. 1-(c). 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.

Finally, in Fig. 1-(d), 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 [20] and adaptive signal processing [1]. 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. 1-(d).

V. Conclusions

In this paper, we study the generic form of network cost minimization problem, which includes both node costs and link costs. The formulated problem has broad applications in distributed signal processing and control over networks. 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 certain technical assumptions, we analyze the convergence properties of the proposed 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.

References

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

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

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

[4] 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 Transactions on Smart Grid*, vol. 7, no. 4, pp. 2123–2133, 2016.

[5] 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 Transactions on signal processing*, vol. 60, no. 6, pp. 2988–3003, 2012.

[6] E. Wei, A. Ozdaglar, and A. Jadbabaie, "A distributed newton method for network utility maximization–i: Algorithm," *IEEE Transactions on Automatic Control*, vol. 58, no. 9, pp. 2162–2175, 2013.

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

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

[9] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, 2009.

[10] W. Shi, Q. Ling, K. Yuan, G. Wu, and W. Yin, "On the linear convergence of the admm in decentralized consensus optimization," *IEEE Transactions on Signal Processing*, vol. 62, no. 7, pp. 1750–1761, 2014.

[11] Q. Ling, W. Shi, G. Wu, and A. Ribeiro, "Dlm: Decentralized linearized alternating direction method of multipliers," *IEEE Transactions on Signal Processing*, vol. 63, no. 15, pp. 4051–4064, 2015.

[12] T.-H. Chang, M. Hong, and X. Wang, "Multi-agent distributed optimization via inexact consensus admm," *IEEE Transactions on Signal Processing*, vol. 63, no. 2, pp. 482–497, 2015.

[13] D. Jakovetic, J. Xavier, and J. M. Moura, "Fast distributed gradient methods," *IEEE Transactions on Automatic Control*, vol. 59, no. 5, pp. 1131–1146, 2014.

[14] J. Chen, C. Richard, and A. H. Sayed, "Multitask diffusion adaptation over networks," *IEEE Transactions on Signal Processing*, vol. 62, no. 16, pp. 4129–4144, 2014.

[15] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[16] A. Mokhtari, W. Shi, Q. Ling, and A. Ribeiro, "Dqm: Decentralized quadratically approximated alternating direction method of multipliers," *IEEE Transactions on Signal Processing*, vol. 64, no. 19, pp. 5158–5173, 2016.

[17] W. Deng and W. Yin, "On the global and linear convergence of the generalized alternating direction method of multipliers," *Journal of Scientific Computing*, vol. 66, no. 3, pp. 889–916, 2016.

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

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

[20] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.