Distributed Newton's Method for Network Cost Minimization

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

***Abstract*—In this article, we examine a novel generic network cost minimization problem, in which every node has a local decision vector to optimize. Each node incurs a cost associated with its decision vector, while each link incurs a cost related to the decision vectors of its two end nodes. All nodes collaborate to minimize the overall network cost. The formulated network cost minimization problem has broad applications in distributed signal processing and control, in which the notion of link costs often arises. To solve this problem in a decentralized manner, we develop a distributed variant of Newton's method, which possesses faster convergence than alternative first-order optimization methods such as gradient descent and alternating direction method of multipliers. The proposed method is based on an appropriate splitting of the Hessian matrix and an approximation of its inverse, which is used to determine the Newton step. Global linear convergence of the proposed algorithm is established under several standard technical assumptions on the local cost functions. Furthermore, analogous to classical centralized Newton's method, a quadratic convergence phase of the algorithm over a certain time interval is identified. Finally, numerical simulations are conducted to validate the effectiveness of the proposed algorithm and its superiority over other first-order methods, especially when the cost functions are ill-conditioned. Complexity issues of the proposed distributed Newton's method and alternative first-order methods are also discussed.**

***Index Terms*—Decentralized optimization, linear convergence, network cost minimization, network optimization, Newton's methodquadratic convergence.**

I. Introduction

The advancement of decentralized signal processing and control in multiagent systems relies on the development of various distributed optimization methods. Multiagent optimization problems arise in many applications in networked systems such as adaptive signal processing over networks [1], distributed estimation over sensor networks [2], [3], and wireless communication networks [4], [5]. In these scenarios, data are inherently distributed over individual nodes across the network. Centralized data processing relying on some central entity suffers from prohibitively high communication overhead and is vulnerable to link failures and network congestions. Therefore, optimizing and processing data in a decentralized manner with only local information exchanges among neighbors is more favorable due to its robustness to failures, scalability to large networks, and efficiency in communications.

Manuscript received January 3, 2019; revised October 2, 2019; accepted April 17, 2020. Date of publication April 20, 2020; date of current version February 26, 2021. Recommended for publication by Associate Editor Prof. Fabian Wirth. *(Corresponding author: Xuanyu Cao.)*

Xuanyu Cao is with the Coordinated Science Lab, University of Illinois at Urbana-Champaign, Champaign, IL 61801 USA (e-mail: xyc@illinois.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).

Color versions of one or more of the figures in this article are available online at https://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TAC.2020.2989266

Owing to its importance, distributed optimization over networks has been extensively studied in the literature. One major class of distributed optimization problems is distributed network utility maximization (NUM), in which each agent has some utility related to its local decision variable. Agents cooperatively maximize the total utilities of the entire network subject to some coupling resource constraints such as the link capacity constraints in the flow scheduling problems of communication networks. Various optimization decomposition techniques have been employed to solve NUM in communication networks in a decentralized fashion and these decompositions lead to elegant architectural modularity and layering of communication systems [6]–[9]. In addition, Wei *et al.* propose and analyze a distributed Newton's method in [10], while the effect of noisy information exchange has been studied in [11]. Recently, Niu and Li present an asynchronous decentralized algorithm with pricing interpretations for NUM [12].

Another category of distributed optimization problems more related to this article is consensus optimization, in which all agents share the same decision variables but have different local cost functions. The goal of consensus optimization is to maximize the total costs of the whole network collaboratively. To this end, Nedic and Ozdaglar propose a decentralized subgradient method for consensus optimization in their seminal work [13], while a dual-averaging method is presented in [14]. A convergence analysis of the decentralized gradient descent algorithm for consensus optimization is provided in [15]. Moreover, consensus optimization has been studied using the distributed Nesterov gradient algorithm in [16] and the distributed alternating direction method of multipliers (ADMM) in [17]. Later, variants of the distributed ADMM have been proposed for consensus optimization, including the quadratically approximated ADMM [18], the inexact ADMM [19], the asynchronous ADMM [20]–[22], and the proximal dual ADMM [23]. Moreover, the second-order optimization algorithm based on Newton's method is proposed for consensus optimization in [24], and is further extended to an asynchronous setting in [25]. Distributed quasi-Newton method (BFGS) has also been proposed in [26], where second-order information is not readily available. There, first-order information (gradient) is exploited to approximate Newton's method in a decentralized manner.

In the aforementioned works, only costs or utilities at nodes are taken into account, while the costs or gains of links are ignored. For instance, in consensus optimization, the network cost, that is the objective function, is only comprised of local cost at each node while the effect of the link is not incorporated. Nevertheless, the notion of link costs or link utilities may arise in many practical signal processing and control problems. For example, in distributed multitask adaptive learning [27], each node i aims at estimating its own weight vector $\mathbf{w}_i$, which, unlike consensus optimization, is different from other nodes' weight vectors. In most cases, neighboring nodes incline to have similar weight vectors. To incorporate this prior information 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 connected by an edge. This term is tantamount to a link cost of the link (i, j).

Despite its usefulness, the notion of link costs (or utilities) is not well studied except for some specific applications such as multitask

adaptive estimation [27]. The generic form of network cost minimization problem incorporating link costs has been examined in [28] recently and a distributed (linearized) ADMM algorithm has been proposed and analyzed. However, as observed in centralized setting [29], first-order optimization methods such as ADMM, gradient descent and their variants often suffer from slow convergence, especially when the problem data are ill-conditioned, that is, when the objective function has large condition number. We are thus motivated to invoke Newton's method for the network cost minimization problem in which both node costs and link costs take place. Inspired by the recent work on network Newton algorithm for decentralized consensus optimization [24], we develop a distributed variant of Newton's method for the generic network cost minimization problem in this article. Since the formulated network cost minimization problem encompasses consensus optimization as a special case, the proposed distributed Newton's method in this article can be viewed as an extension of the network Newton algorithm in [24] to optimization problems with link costs. Our contributions are summarized in the following.

1) A distributed Newton's method (Algorithm 1) is developed for a novel generic network cost minimization problem, which takes both node costs and link costs into consideration. The proposed algorithm is based on appropriate splitting of the Hessian matrix and a corresponding approximation of its inverse so that the computation of the Newton step can be distributed to each node in parallel.
2) Performance analysis of the proposed distributed Newton's method is presented. In particular, global linear convergence of the algorithm is guaranteed under some standard assumptions on the local cost functions (Theorem 1). Moreover, analogous to the classical centralized Newton's method [29], a quadratic convergence phase of the algorithm over a certain time interval is identified (Theorem 2).
3) Numerical experiments on quadratic programming are implemented to corroborate the effectiveness of the proposed algorithm, which outperforms alternative first-order optimization methods [namely, the distributed ADMM and the distributed gradient descent (DGD)] significantly in terms of both convergence time and number of per-node information exchanges. Impact of the condition number of the cost functions and the network topology is also investigated empirically through simulations.

The key difference between this article and existing literature is the joint optimization of the general node/link cost functions, which necessitates a new analysis of Newton-type algorithms. We note that the matrix splitting based Newton-type methods has been proposed to solve different problems in prior works, for example [10] for NUM, [30] for network flow optimization, [24] for consensus optimization. Yet none of these existing works consider the joint optimization of generic node/link cost functions, which are of interest in this article. The organization of the rest of this article is as follows. In Section II, the network cost minimization problem is formally formulated and a distributed Newton's method is developed to solve it. Convergence analysis of the proposed algorithm is conducted in Section III while numerical results are presented in Section IV. Complexity issues of the proposed algorithm and alternative first-order methods are discussed in Section V and we conclude this article in Section VI.

Notations: Denote $\{1, 2, \ldots, n\}$ as $[n]$. $\|\mathbf{x}\|_2$ means the Euclidean norm of vector $\mathbf{x}$ while $\|\mathbf{A}\|_2$ means the spectral norm (maximum singular value) of matrix $\mathbf{A}$. $\rho(\mathbf{A})$ is the spectral radius of $\mathbf{A} \in \mathbb{R}^{n\times n}$, that is, $\rho(\mathbf{A}) = \max_{i\in[n]} |\lambda_i(\mathbf{A})|$, where $\lambda_i(\mathbf{A})$'s are the eigenvalues of $\mathbf{A}$. Denote the sets of $n \times n$ symmetric matrices, positive semidefinite matrices, and positive definite matrices as $\mathbb{S}^n$, $\mathbb{S}^n_+$, and $\mathbb{S}^{n++}$, respectively. For two symmetric matrices $\mathbf{A}, \mathbf{B} \in \mathbb{S}^n$, $\mathbf{A} \preceq \mathbf{B}$ means $\mathbf{B} - \mathbf{A}$ is positive semidefinite. For a twice differentiable function $\phi : \mathbb{R}^a \times \mathbb{R}^b \mapsto \mathbb{R}$ and $\mathbf{x} \in \mathbb{R}^a, \mathbf{y} \in \mathbb{R}^b$, we define matrix $\nabla^2_{\mathbf{x},\mathbf{y}}\phi(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^{a\times b}$ according to $[\nabla^2_{\mathbf{x},\mathbf{y}}\phi(\mathbf{x}, \mathbf{y})]_{ij} = \frac{\partial^2\phi(\mathbf{x},\mathbf{y})}{\partial x_i \partial y_j}$ and matrix $\nabla^2_{\mathbf{y},\mathbf{x}}\phi(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^{b\times a}$ according to $[\nabla^2_{\mathbf{y},\mathbf{x}}\phi(\mathbf{x}, \mathbf{y})]_{ij} = \frac{\partial^2\phi(\mathbf{x},\mathbf{y})}{\partial y_i \partial x_j}$. Thus, we have $\nabla^2{}_{\mathbf{x},\mathbf{y}}\phi(\mathbf{x}, \mathbf{y}) = \nabla^2_{\mathbf{y},\mathbf{x}}\phi(\mathbf{x}, \mathbf{y})^\mathsf{T}$. Define $\nabla^2_{\mathbf{x}}\phi(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^{a\times a}$ as $[\nabla^2_{\mathbf{x}}\phi(\mathbf{x}, \mathbf{y})]_{ij} = \frac{\partial^2\phi(\mathbf{x},\mathbf{y})}{\partial x_i \partial x_j}$. Analogous definition applies to $\nabla^2_{\mathbf{y}}\phi(\mathbf{x}, \mathbf{y})$. Define $\nabla^2\phi(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^{(a+b)\times(a+b)}$ to be the complete Hessian matrix with respect to the joint vector $[\mathbf{x}^\mathsf{T}, \mathbf{y}^\mathsf{T}]^\mathsf{T}$

$$\nabla^2\phi(\mathbf{x}, \mathbf{y}) = \begin{bmatrix} \nabla^2_{\mathbf{x}}\phi(\mathbf{x}, \mathbf{y}) & \nabla^2_{\mathbf{x},\mathbf{y}}\phi(\mathbf{x}, \mathbf{y}) \\ \nabla^2_{\mathbf{y},\mathbf{x}}\phi(\mathbf{x}, \mathbf{y}) & \nabla^2_{\mathbf{y}}\phi(\mathbf{x}, \mathbf{y}) \end{bmatrix}. \tag{1}$$

II. Problem Formulation and Algorithm Development

In this section, the network cost minimization problem is formulated formally and its applications are discussed. Afterwards, by appropriate splitting and approximation of the Hessian matrix of the objective function, we develop a distributed variant of Newton's method for the formulated network cost minimization problem.

A. Problem Formulation

Consider a network of n nodes. Assume the network is a simple graph, that is, the network is undirected with no self-loop and there is at most one edge between any pair of nodes. Denote the set of neighbors of node i (those who are linked with node i with an edge) as Ω_i. The network can be either connected or disconnected (there does not necessarily exist a path connecting every pair of nodes). Each node i has a p-dimensional local decision variable $\mathbf{x}_i \in \mathbb{R}^p$. Given $\mathbf{x}_i$, the cost of node i is $f_i(\mathbf{x}_i)$, where f_i is the node cost function of node i. Furthermore, for two linked 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 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:

$$\text{Minimize} \sum_{i=1}^{n} f_i(\mathbf{x}_i) + \sum_{i=1}^{n}\sum_{j\in\Omega_i} g_{ij}(\mathbf{x}_i, \mathbf{x}_j). \tag{2}$$

We note that the consensus optimization problems in [13]–[24], [31] are special cases of the network cost minimization problem (2) here. In fact, 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. Additionally, in [24], [25], the consensus optimization problem is transformed into $\min_{\mathbf{x}} \alpha \sum_{i=1}^n f_i(\mathbf{x}_i) + \frac{1}{2}\mathbf{x}^\mathsf{T}(\mathbf{I} - \mathbf{Z})\mathbf{x}$, where α is some positive constant; $\mathbf{x}$ is the concatenation of all $\mathbf{x}_i$'s; and $\mathbf{Z}$ is the block weight matrix specifying the combination weights of neighbors. The second term will enforce consensus and the parameter α can adjust the consensus level. A prominent difference between this problem and problem (2) is that the second term of the former problem is quadratic with a particular coefficient matrix structure (identity minus block weight matrix) to enforce consensus. In contrast, the link cost functions g_{ij}'s in problem (2) can be general as long as they satisfy the standard assumptions to be specified later. Further, an average consensus based distributed Newton's method has been proposed in [32] to solve consensus optimization problems with asynchronous and lossy communications. The approach in [32] is tailored to consensus optimization and cannot be readily applied to the network cost minimization problem (2) in this article. Instead, we take an alternative approach and

approximate the Newton step in a distributed manner by computing a truncation of the Taylor expansion of the inverse Hessian matrix. The problem formulation (2) has broad applications, among which we name two in the following.

1) 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, that is, $f_i(\mathbf{x}_i)$, may be some squared error or more generally 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 proximity between neighboring nodes, for example, $\|\mathbf{x}_i - \mathbf{x}_j\|_2^2$ in multitask adaptive networks in [27].
2) In resource allocation over networks, $\mathbf{x}_i$ corresponds to some resources consumed by 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 quantify 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) represents the cost incurred by mutual interference in wireless communications.

Recently, a distributed linearized ADMM algorithm has been proposed to solve the network cost minimization problem (2) in a decentralized and computationally efficient manner [28]. We note that the first-order methods such as variants of ADMM and subgradient methods generally have slower convergence than the second-order methods (e.g., Newton's method or quasi-Newton methods) do, especially when the objective function is ill-conditioned [29]. Motivated by this fact and inspired by the recent work [24] on network Newton algorithm for consensus optimization, we develop a distributed variant of Newton's method for the generic network cost minimization problem (2), which takes link costs into account and encompasses consensus optimization as a special case. Moreover, we make the following two technical assumptions that are standard in the literature of numerical optimization [29].

Assumption 1: There exist two positive constants $0 < m < M$ such that, for any $i \in [n], j \in \Omega_i$ and $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^p$

$$m\mathbf{I} \preceq \nabla^2 f_i(\mathbf{x}_i) \preceq M\mathbf{I} \tag{3}$$

$$m\mathbf{I} \preceq \nabla^2 g_{ij}(\mathbf{x}_i, \mathbf{x}_j) \preceq M\mathbf{I}. \tag{4}$$

Assumption 2: There exists a positive constant $L > 0$ such that the Hessian matrices of all f_i's and g_{ij}'s are L-Lipschitz continuous, that is, for any $i \in [n], j \in \Omega_i$ and $\mathbf{x}_i, \mathbf{x}_i{}', \mathbf{x}_j, \mathbf{x}_j'$

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

$$\|\nabla^2 g_{ij}(\mathbf{x}_i, \mathbf{x}_j) - \nabla^2 g_{ij}(\mathbf{x}_i,' \mathbf{x}_j')\|_2 \leq L \left\| \begin{bmatrix} \mathbf{x}_i \\ \mathbf{x}_j \end{bmatrix} - \begin{bmatrix} \mathbf{x}_i' \\ \mathbf{x}_j' \end{bmatrix} \right\|_2.$$

We note that Assumption 1 is a bit strong and is not satisfied by all applications, for example, affine functions are not strongly convex and thus do not satisfy Assumption 1. Without Assumption 1, ADMM can still be shown to converge [20]. Nevertheless, to guarantee global linear convergence of ADMM, Assumption 1 (i.e., strong convexity and Lipschitz continuous gradient) is usually needed [17]. In this article, we will show global linear convergence of the proposed distributed Newton's method and thus we let Assumption 1 hold.

B. Algorithm Development

Define $\mathbf{x} \in \mathbb{R}^{np}$ as the concatenation of all the $\mathbf{x}_i$'s. Denote the objective function of (2) as $F(\mathbf{x}) := \sum_{i=1}^n f_i(\mathbf{x}_i) + \sum_{i=1}^n \sum_{j\in\Omega_i} g_{ij}(\mathbf{x}_i, \mathbf{x}_j)$. Denote the unique minimizer of F as $\mathbf{x}^*$, where the uniqueness results from the strong convexity assumption, that is, Assumption 1. In the rest of the article, unless explicitly specified, we use $[\cdot]_i$ to denote the i-th p-dimensional subvector of a vector and use $[\cdot]_{i,j}$ to denote the (i, j)th $p \times p$ block of a matrix. To apply Newton's method to (2), we compute the gradient of F as

$$[\nabla F(\mathbf{x})]_i = \nabla f_i(\mathbf{x}_i) + \sum_{j\in\Omega_i} [\nabla_{\mathbf{x}_i} g_{ij}(\mathbf{x}_i, \mathbf{x}_j) + \nabla_{\mathbf{x}_i} g_{ji}(\mathbf{x}_j, \mathbf{x}_i)].$$

Denote $\mathbf{H}(\mathbf{x}) := \nabla^2 F(\mathbf{x})$ the Hessian matrix of F, which can be computed as

$$[\mathbf{H}(\mathbf{x})]_{ik} = \begin{cases} \nabla^2 f_i(\mathbf{x}_i) + \sum_{j\in\Omega_i} [\nabla^2_{\mathbf{x}_i} g_{ij}(\mathbf{x}_i, \mathbf{x}_j) + \nabla^2_{\mathbf{x}_i} g_{ji}(\mathbf{x}_j, \mathbf{x}_i)] \\ \qquad\qquad\qquad\qquad \text{if } i = k \\ \nabla^2_{\mathbf{x}_i,\mathbf{x}_k} g_{ik}(\mathbf{x}_i, \mathbf{x}_k) + \nabla^2_{\mathbf{x}_i,\mathbf{x}_k} g_{ki}(\mathbf{x}_k, \mathbf{x}_i), \text{ if } k \in \Omega_i \\ \mathbf{0}, \text{otherwise.} \end{cases}$$

We note that $\mathbf{H}(\mathbf{x})$ is positive definite (according to Assumption 1) and block sparse with the sparsity pattern of the network. We further define a block diagonal matrix $\mathbf{D}(\mathbf{x})$ as

$$[\mathbf{D}(\mathbf{x})]_{ik} = \begin{cases} \nabla^2 f_i(\mathbf{x}_i) + 2\sum_{j\in\Omega_i} [\nabla^2_{\mathbf{x}_i} g_{ij}(\mathbf{x}_i, \mathbf{x}_j) + \nabla^2_{\mathbf{x}_i} g_{ji}(\mathbf{x}_j, \mathbf{x}_i)] \\ \qquad\qquad\qquad\qquad \text{if } i = k \\ \mathbf{0}, \text{otherwise} \end{cases}$$

and a block sparse matrix $\mathbf{B}(\mathbf{x})$ as

$$[\mathbf{B}(\mathbf{x})]_{ik} = \begin{cases} \sum_{j\in\Omega_i} [\nabla^2_{\mathbf{x}_i} g_{ij}(\mathbf{x}_i, \mathbf{x}_j) + \nabla^2_{\mathbf{x}_i} g_{ji}(\mathbf{x}_j, \mathbf{x}_i)], \text{if} i = k \\ -\nabla^2_{\mathbf{x}_i,\mathbf{x}_k} g_{ik}(\mathbf{x}_i, \mathbf{x}_k) - \nabla^2_{\mathbf{x}_i,\mathbf{x}_k} g_{ki}(\mathbf{x}_k, \mathbf{x}_i), \text{if} k \in \Omega_i \\ \mathbf{0}, \text{otherwise.} \end{cases}$$

Thus, we obtain a splitting of the Hessian matrix as $\mathbf{H}(\mathbf{x}) = \mathbf{D}(\mathbf{x}) - \mathbf{B}(\mathbf{x})$. According to Assumption 1, it is easy to see that $\mathbf{D}(\mathbf{x})$ is positive definite. So, we can write $\mathbf{H}(\mathbf{x}) = \mathbf{D}(\mathbf{x})^{\frac{1}{2}}[\mathbf{I} - \mathbf{D}(\mathbf{x})^{-\frac{1}{2}}\mathbf{B}(\mathbf{x})\mathbf{D}(\mathbf{x})^{-\frac{1}{2}}]\mathbf{D}(\mathbf{x})^{\frac{1}{2}}$. To invoke Newton's method, we need to calculate $\mathbf{H}(\mathbf{x})^{-1} = \mathbf{D}(\mathbf{x})^{-\frac{1}{2}}[\mathbf{I} - \mathbf{D}(\mathbf{x})^{-\frac{1}{2}}\mathbf{B}(\mathbf{x})\mathbf{D}(\mathbf{x})^{-\frac{1}{2}}]^{-1}\mathbf{D}(\mathbf{x})^{-\frac{1}{2}}$. Unfortunately, $\mathbf{H}(\mathbf{x})^{-1}$ is not necessarily block sparse so that the exact Newton's method for minimizing $F(\mathbf{x})$ cannot be implemented in a distributed fashion. Therefore, to obtain a distributed algorithm, we resort to some approximated version of $\mathbf{H}(\mathbf{x})^{-1}$. To this end, if $\rho(\mathbf{D}(\mathbf{x})^{-\frac{1}{2}}\mathbf{B}(\mathbf{x})\mathbf{D}(\mathbf{x})^{-\frac{1}{2}}) < 1$ (which will be shown later in Section III), we can rewrite $\mathbf{H}(\mathbf{x})^{-1}$ as

$$\mathbf{H}(\mathbf{x})^{-1} = \mathbf{D}(\mathbf{x})^{-\frac{1}{2}} \sum_{k=0}^{\infty} \left[\mathbf{D}(\mathbf{x})^{-\frac{1}{2}}\mathbf{B}(\mathbf{x})\mathbf{D}(\mathbf{x})^{-\frac{1}{2}}\right]^k \mathbf{D}(\mathbf{x})^{-\frac{1}{2}}. \tag{5}$$

Truncating the first $K + 1$ ($K \geq 0$) terms of the summation in (5), we note that $\mathbf{D}(\mathbf{x})^{-\frac{1}{2}} \sum_{k=0}^K [\mathbf{D}(\mathbf{x})^{-\frac{1}{2}}\mathbf{B}(\mathbf{x})\mathbf{D}(\mathbf{x})^{-\frac{1}{2}}]^k \mathbf{D}(\mathbf{x})^{-\frac{1}{2}}$ is still positive definite. As such, we can define a positive definite approximated Hessian $\hat{\mathbf{H}}(\mathbf{x})$ as

$$\hat{\mathbf{H}}(\mathbf{x}) := \left\{ \mathbf{D}(\mathbf{x})^{-\frac{1}{2}} \sum_{k=0}^{K} \left[\mathbf{D}(\mathbf{x})^{-\frac{1}{2}}\mathbf{B}(\mathbf{x})\mathbf{D}(\mathbf{x})^{-\frac{1}{2}}\right]^k \mathbf{D}(\mathbf{x})^{-\frac{1}{2}} \right\}^{-1}. \tag{6}$$

Denote the iterate at time t as $\mathbf{x}_t$. Define $\mathbf{h}_t = \nabla F(\mathbf{x}_t)$ and $\hat{\mathbf{H}}_t = \hat{\mathbf{H}}(\mathbf{x}_t)$. Thus, the approximated Newton direction is $\mathbf{d}_t = -\hat{\mathbf{H}}_t{}^{-1}\mathbf{h}_t$ and the approximated Newton update is $\mathbf{x}_{t+1} = \mathbf{x}_t + \epsilon\mathbf{d}_t$, where $\epsilon > 0$ is the step size. Next, we demonstrate that the approximated Newton direction $\mathbf{d}_t$ can be computed in a distributed and recursive manner. To this end, define the kth ($k \geq 0$) order approximated Hessian matrix $\hat{\mathbf{H}}_k(\mathbf{x})$:

$$\hat{\mathbf{H}}_k(\mathbf{x}) := \left\{ \mathbf{D}(\mathbf{x})^{-\frac{1}{2}} \sum_{l=0}^{k} \left[\mathbf{D}(\mathbf{x})^{-\frac{1}{2}}\mathbf{B}(\mathbf{x})\mathbf{D}(\mathbf{x})^{-\frac{1}{2}}\right]^l \mathbf{D}(\mathbf{x})^{-\frac{1}{2}} \right\}^{-1}.$$

Furthermore, define $\mathbf{D}_t = \mathbf{D}(\mathbf{x}_t)$, $\mathbf{B}_t = \mathbf{B}(\mathbf{x}_t)$, $\mathbf{H}_t = \mathbf{H}(\mathbf{x}_t)$, $\hat{\mathbf{H}}_{k,t} = \hat{\mathbf{H}}_k(\mathbf{x}_t)$, and $\mathbf{d}_{k,t} = -\hat{\mathbf{H}}_{k,t}^{-1}\mathbf{h}_t$. Thus, $\mathbf{d}_t = \mathbf{d}_{K,t}$. The approximated Newton direction can be calculated recursively as

$$\mathbf{d}_{k+1,t} = -\mathbf{D}_t^{-\frac{1}{2}}\left[\mathbf{I} + \sum_{l=1}^{k+1}\left(\mathbf{D}_t^{-\frac{1}{2}}\mathbf{B}_t\mathbf{D}_t^{-\frac{1}{2}}\right)^l\right]\mathbf{D}_t^{-\frac{1}{2}}\mathbf{h}_t. \tag{7}$$

$$= -\mathbf{D}_t^{-1}\mathbf{h}_t + \mathbf{D}_t^{-1}\mathbf{B}_t\mathbf{d}_{k,t} \tag{8}$$

$$= \mathbf{D}_t^{-1}(\mathbf{B}_t\mathbf{d}_{k,t} - \mathbf{h}_t) \tag{9}$$

Algorithm 1 Distributed Newton's Method for Network Cost Minimization: Procedures at Node i.

1: Initialize $\mathbf{x}_{0,i}$ and step size ϵ
2: **for** $t = 0, 1, 2, \ldots$ **do**
3: Exchange the iterate $\mathbf{x}_{t,i}$ with neighbors $j \in \Omega_i$.
4: Compute:

$$\mathbf{D}_{t,ii} = \nabla^2 f_i(\mathbf{x}_{t,i}) + 2\sum_{j\in\Omega_i}\left[\nabla^2_{\mathbf{x}_i} g_{ij}(\mathbf{x}_{t,i},\mathbf{x}_{t,j}) + \nabla^2_{\mathbf{x}_i} g_{ji}(\mathbf{x}_{t,j},\mathbf{x}_{t,i})\right],$$

$$\mathbf{B}_{t,ii} = \sum_{j\in\Omega_i}[\nabla^2_{\mathbf{x}_i} g_{ij}(\mathbf{x}_{t,i},\mathbf{x}_{t,j}) + \nabla^2_{\mathbf{x}_i} g_{ji}(\mathbf{x}_{t,j},\mathbf{x}_{t,i})], \tag{10}$$

$$\mathbf{B}_{t,ij} = -\nabla^2_{\mathbf{x}_i,\mathbf{x}_j} g_{ij}(\mathbf{x}_{t,i},\mathbf{x}_{t,j}) - \nabla^2_{\mathbf{x}_i,\mathbf{x}_j} g_{ji}(\mathbf{x}_{t,j},\mathbf{x}_{t,i}), \quad \forall j \in \Omega_i, \tag{11}$$

$$\mathbf{h}_{t,i} = \nabla f_i(\mathbf{x}_{t,i}) + \sum_{j\in\Omega_i}\left[\nabla_{\mathbf{x}_i} g_{ij}(\mathbf{x}_{t,i},\mathbf{x}_{t,j}) + \nabla_{\mathbf{x}_i} g_{ji}(\mathbf{x}_{t,j},\mathbf{x}_{t,i})\right], \tag{12}$$

$$\mathbf{d}_{0,t,i} = -\mathbf{D}_{t,ii}^{-1}\mathbf{h}_{t,i}. \tag{13}$$

5: **for** $k = 0, 1, \ldots, K-1$ **do**
6: Exchange the iterate $\mathbf{d}_{k,t,i}$ with neighbors $j \in \Omega_i$.
7: Compute:

$$\mathbf{d}_{k+1,t,i} = \mathbf{D}_{t,ii}^{-1}\left(\sum_{j\in\Omega_i\cup\{i\}}\mathbf{B}_{t,ij}\mathbf{d}_{k,t,j} - \mathbf{h}_{t,i}\right). \tag{14}$$

8: **end for**
9: Set $\mathbf{d}_{t,i} = \mathbf{d}_{K,t,i}$.
10: Update $\mathbf{x}_{t+1,i} = \mathbf{x}_{t,i} + \epsilon\mathbf{d}_{t,i}$.
11: **end for**

Noting that $\mathbf{D}_t$ is block diagonal, we have

$$\mathbf{d}_{k+1,t,i} = \mathbf{D}_{t,ii}^{-1}\left(\sum_{j\in\Omega_i\cup\{i\}}\mathbf{B}_{t,ij}\mathbf{d}_{k,t,j} - \mathbf{h}_{t,i}\right). \tag{15}$$

Equation (15) indicates that the approximated Newton direction $\mathbf{d}_t$ can be computed in a distributed and recursive way. Thus, a distributed Newton's method for the network cost minimization problem (2) can be developed and the proposed algorithm is detailed in Algorithm 1 from the perspective of node i.

The overall framework of the analysis of Algorithm 1 will follow a path analogous to that of [24]. The main distinction of this article is the introduction of general link cost functions as opposed to the quadratic link cost functions used in [24] to enforce consensus. Algorithmically, the expressions of $\mathbf{D}_t, \mathbf{B}_t, \mathbf{h}_t$ take more general forms in Algorithm 1 as they depend on the general link cost functions. Therefore, some structural properties of the algorithm in [24] no longer hold for Algorithm 1 in this article. For instance, the submatrices $\mathbf{B}_{t,ii}$ and $\mathbf{B}_{t,ij}$ (i, j are neighbors) are scalar multiples of identity matrix in [24], while these submatrices can be general symmetric matrices in Algorithm 1. These differences and generalizations in algorithms require new analysis, which is a nontrivial extension of the analysis in [24] and leads to new features of the convergence results. For instance, the proof of Proposition 2 in [24] has exploited the special form of the quadratic link cost functions (the double stochasticity of the combination weight matrix $\mathbf{W}$ in particular) and the Gershgorin circle theorem (cf. equations (58), (59) in [24]) to bound the eigenvalues. Such a technique no longer works in the proof of Lemma 2 in this article (the counterpart of Proposition 2 in [24]), since the link cost functions are generally nonquadratic. To show Lemma 2, we have to adopt other techniques to bound the eigenvalues of the involved matrices and the details of the proof are different from that of Proposition 2 in [24]. Further, the convergence results in [24] do not depend on the network connectivity, while the convergence results (e.g., the value of ξ in Theorem 1) in this article depend on the network connectivity explicitly through the maximum node degree C.

In [30], matrix splitting based Newton's method has also been proposed for network flow optimization problem, which only involves link costs and the link rates need to satisfy the flow conservation constraints. Though Algorithm 1 in this article shares similar spirit with that in [30], the specific algorithmic implementations are very different due to the different optimization problems. As such, new convergence analysis is required for the proposed Algorithm 1.

III. Convergence Analysis

In this section, we analyze the convergence properties of the proposed distributed Newton's method for network cost minimization, that is, Algorithm 1. Specifically, we demonstrate global linear convergence of the objective function value $F(\mathbf{x}_t)$ to the optimal value $F(\mathbf{x}^*)$. Furthermore, we show that Algorithm 1 possesses a quadratic convergence phase, which is a generic theoretical advantage of the second-order optimization methods over first-order ones [29], [33]. The proofs of all the lemmas and theorems are relegated to the supplementary material due to space limitation.

A. The Global Linear Convergence

In this subsection, we demonstrate global linear convergence of Algorithm 1. We first establish bounds on the matrices $\mathbf{B}(\mathbf{x}), \mathbf{H}(\mathbf{x}), \mathbf{D}(\mathbf{x})$ in the following lemma. Define $C = \max_{i\in[n]}|\Omega_i|$ to be the maximum node degree.

Lemma 1: For any $\mathbf{x} \in \mathbb{R}^{np}$:

$$\mathbf{0} \preceq \mathbf{B}(\mathbf{x}) \preceq 2MC\mathbf{I} \tag{16}$$

$$m\mathbf{I} \preceq \mathbf{H}(\mathbf{x}) \preceq M(1+2C)\mathbf{I} \tag{17}$$

$$m\mathbf{I} \preceq \mathbf{D}(\mathbf{x}) \preceq (1+4C)M\mathbf{I}. \tag{18}$$

In order to ensure that the series in (5) are convergent, we need to guarantee that the spectral radius of $\mathbf{D}(\mathbf{x})^{-\frac{1}{2}}\mathbf{B}(\mathbf{x})\mathbf{D}(\mathbf{x})^{-\frac{1}{2}}$ is strictly smaller than 1, as shown in the following lemma.

Lemma 2: For any $\mathbf{x} \in \mathbb{R}^{np}$:

$$\mathbf{0} \preceq \mathbf{D}(\mathbf{x})^{-\frac{1}{2}}\mathbf{B}(\mathbf{x})\mathbf{D}(\mathbf{x})^{-\frac{1}{2}} \preceq \eta\mathbf{I}, \tag{19}$$

where $\eta = 1 - \frac{m}{M(1+4C)} \in (0,1)$ is a constant. Therefore, we have

$$\rho\left(\mathbf{D}(\mathbf{x})^{-\frac{1}{2}}\mathbf{B}(\mathbf{x})\mathbf{D}(\mathbf{x})^{-\frac{1}{2}}\right) \le \eta < 1. \tag{20}$$

Lemma 2 guarantees the convergence of the series in (5) and justifies the truncated approximation of Hessian in (6). Then, a natural question is about the approximation accuracy of the approximated Hessian $\hat{\mathbf{H}}(\mathbf{x})$. To quantify this accuracy, we define the error matrix $\mathbf{E}(\mathbf{x}) \in \mathbb{S}^{np}$

as

$$\mathbf{E}(\mathbf{x}) := \mathbf{I} - \hat{\mathbf{H}}(\mathbf{x})^{-\frac{1}{2}}\mathbf{H}(\mathbf{x})\hat{\mathbf{H}}(\mathbf{x})^{-\frac{1}{2}}. \tag{21}$$

Define $\mathbf{E}_t = \mathbf{E}(\mathbf{x}_t)$. Then, we have the following bound for the error matrix $\mathbf{E}(\mathbf{x})$.

Lemma 3: For any $\mathbf{x} \in \mathbb{R}^{np}$

$$\mathbf{0} \preceq \mathbf{E}(\mathbf{x}) \preceq \eta^{K+1}\mathbf{I}. \tag{22}$$

In accordance with one's intuition, Lemma 3 indicates that the larger the order of approximation K, the smaller the approximation error of the Hessian matrix. This benefit comes at the expense of higher communication and computation overhead of Algorithm 1 when calculating the approximated Newton step $\mathbf{d}_t$ recursively by (14), that is, there exists an accuracy-complexity tradeoff. Furthermore, analogous to Lemma 1, we can also bound the inverse of the approximated Hessian matrix $\hat{\mathbf{H}}(\mathbf{x})^{-1}$ as follows.

Lemma 4: For any $\mathbf{x} \in \mathbb{R}^{np}$

$$\gamma_1\mathbf{I} \preceq \hat{\mathbf{H}}(\mathbf{x})^{-1} \preceq \gamma_2\mathbf{I}, \tag{23}$$

where the two positive constants γ_1 and γ_2 are given as $\gamma_1 = \frac{1}{(1+4C)M}$ and $\gamma_2 = \frac{1-\eta^{K+1}}{m(1-\eta)}$.

Moreover, we can translate the Lipschitz continuity of the Hessian matrices of the local functions in Assumption 2 to Lipschitz continuity of the global Hessian matrix $\mathbf{H}(\mathbf{x})$.

Lemma 5: For any $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^{np}$

$$\|\mathbf{H}(\mathbf{x}) - \mathbf{H}(\mathbf{x}')\|_2 \leq L(1+2C)\|\mathbf{x} - \mathbf{x}'\|_2. \tag{24}$$

That is, $\mathbf{H}(\cdot)$ is Lipschitz continuous with modulus $L(1+2C)$.

We are now ready to show the first main theorem regarding the global linear convergence of Algorithm 1.

Theorem 1: If the stepsize $\epsilon > 0$ of Algorithm 1 is chosen such that

$$\epsilon < \min\left\{1, \sqrt{\frac{2m\gamma_1}{L(1+2C)\gamma_2^3(2M(1+2C))^{\frac{3}{2}}\sqrt{F(\mathbf{x}_0) - F(\mathbf{x}^*)}}}\right\} \tag{25}$$

then $F(\mathbf{x}_t)$, that is, the objective function values generated by Algorithm 1, converges linearly to the optimal objective function value $F(\mathbf{x}^*)$, or more specifically, for any $t \in \mathbb{N}$

$$0 \leq F(\mathbf{x}_t) - F(\mathbf{x}^*) \leq \xi^t[F(\mathbf{x}_0) - F(\mathbf{x}^*)] \tag{26}$$

where $0 < \xi < 1$ is some constant specified as

$$\xi = 1 - m\gamma_1(2\epsilon - \epsilon^2) + \frac{\epsilon^3}{2}L(1+2C)\gamma_2^3(2M(1+2C))^{\frac{3}{2}}\sqrt{F(\mathbf{x}_0) - F(\mathbf{x}^*)}. \tag{27}$$

In practice, we seldom use the upper bound in (25) to determine the stepsize ϵ since this upper bound is difficult to compute in most applications and satisfying this upper bound is only a sufficient (not necessary) condition for linear convergence. In fact, a practically good choice of ϵ can be larger than the theoretical bound in (25). What this bound shows is that, as long as ϵ is *sufficiently small*, the global linear convergence of the proposed distributed Newton's method can be guaranteed. In practice, we usually determine ϵ empirically by trial and error so that it is neither too large (to avoid divergence) nor too small (to avoid very slow convergence). We note that this empirical choice of stepsize is common in many existing optimization methods.

Further, we note that the upper bound on ϵ in (25) depends on the connectivity of the network. This dependence is through the maximum node degree C and γ_1, γ_2 (γ_1 relies on C; γ_2 relies on η, which further depends on C).

B. The Quadratic Convergence Phase

A classical theoretical explanation of the advantage of the second-order optimization methods (e.g., Newton's method) over the first-order alternatives (e.g., gradient descent method) is that the former possesses a quadratic convergence region [29], [33], in which the algorithms converge very fast. In this subsection, we also identify a quadratic convergence phase of Algorithm 1 as a theoretical justification of its superiority over other the first-order methods. To this end, we first present a lemma regarding the Lipschitz continuity of $\mathbf{D}(\mathbf{x})$.

Lemma 6: For any $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^{np}$:

$$\|\mathbf{D}(\mathbf{x}) - \mathbf{D}(\mathbf{x}')\|_2 \leq L(1+4C)\|\mathbf{x} - \mathbf{x}'\|_2 \tag{28}$$

that is, $\mathbf{D}(\cdot)$ is Lipschitz continuous with modulus $L(1+4C)$.

Define two constants $\mu_1 \geq 0$ and $\mu_2 > 0$:

$$\begin{cases} \mu_1 = \frac{1}{m}[\epsilon L(1+4C)\gamma_1\gamma_2]^{\frac{1}{2}}(2M(1+2C))^{\frac{1}{4}}[F(\mathbf{x}_0) - F(\mathbf{x}^*)]^{\frac{1}{4}} \\ \mu_2 = \frac{\epsilon^2 L(1+2C)\gamma_1\gamma_2^2}{2\sqrt{m}}. \end{cases}$$

Define a sequence $\psi_t = (1 - \epsilon + \epsilon\eta^{K+1})(1 + \mu_1\xi^{\frac{t-1}{4}})$. Suppose ϵ satisfies the condition (25) in Theorem 1. Then, we have $0 < \xi < 1$ and thus ψ_t is a decreasing sequence with limit $\lim_{t\to\infty}\psi_t = 1 - \epsilon + \epsilon\eta^{K+1} \in (0,1)$. So, for t large enough, we have $\psi_t < 1$. Define $t_0 := \arg\min\{t|\psi_t < 1\}$. We state our main theorem regarding the quadratic convergence phase of Algorithm 1 in the following theorem.

Theorem 2: Let ϵ be chosen in accordance with the condition (25). Suppose there exists a time interval $[t_1, t_2]$ with $t_1 \geq t_0$ such that, for any $t \in [t_1, t_2]$

$$\frac{\sqrt{\psi_t}(1-\sqrt{\psi_t})}{\mu_2} \leq \left\|\mathbf{D}_{t-1}^{-\frac{1}{2}}\mathbf{h}_t\right\|_2 \leq \frac{1-\sqrt{\psi_t}}{\mu_2}. \tag{29}$$

Then, for $t \in [t_1, t_2+1]$, we have

$$F(\mathbf{x}_t) - F(\mathbf{x}^*) \leq \frac{\delta^{2^{t-t_1}}}{\mu_2\sqrt{\gamma_1}}\|\mathbf{x}_t - \mathbf{x}^*\|_2 \tag{30}$$

where $\delta := \frac{\mu_2}{1-\sqrt{\psi_{t_1}}}\|\mathbf{D}_{t_1-1}^{-\frac{1}{2}}\mathbf{h}_{t_1}\|_2 \in [0,1)$ and $\lim_{\tau\to\infty}\|\mathbf{x}_\tau - \mathbf{x}^*\|_2 = 0$. In other words, Algorithm 1 converges quadratically over the time interval $[t_1, t_2+1]$. Furthermore, we have $\lim_{\tau\to\infty}\|\mathbf{D}_{\tau-1}^{-\frac{1}{2}}\mathbf{h}_\tau\|_2 = 0$.

Remark 1: From $\lim_{t\to\infty}\|\mathbf{D}_{t-1}^{-\frac{1}{2}}\mathbf{h}_t\|_2 = 0$, $\lim_{t\to\infty}\frac{\sqrt{\psi_t}(1-\sqrt{\psi_t})}{\mu_2} = \frac{\sqrt{1-\epsilon+\epsilon\eta^{K+1}}(1-\sqrt{1-\epsilon+\epsilon\eta^{K+1}})}{\mu_2} > 0$ and $\lim_{t\to\infty}\frac{1-\sqrt{\psi_t}}{\mu_2} = \frac{1-\sqrt{1-\epsilon+\epsilon\eta^{K+1}}}{\mu_2} > 0$, we know that $\|\mathbf{D}_{t-1}^{-\frac{1}{2}}\mathbf{h}_t\|_2$ will eventually be smaller than both bounds in (29) for large enough t. Typically, as t increases, $\|\mathbf{D}_{t-1}^{-\frac{1}{2}}\mathbf{h}_t\|_2$ will first become smaller than the right bound of (29), but still remain larger than the left bound of (29), i.e., (29) holds. Theorem (2) says, in such a case, Algorithm 1 converges quadratically. After that, as t further increases, $\|\mathbf{D}_{t-1}^{-\frac{1}{2}}\mathbf{h}_t\|_2$ becomes even smaller than the left bound of (29) so that (29) does not hold any more and the quadratic convergence phase is terminated. In such a case, we can only guarantee linear convergence rate, which is a global property of Algorithm 1 (Theorem 1).

IV. Numerical Tests

In this section, we empirically investigate the performance of the proposed distributed Newton's method (DNM, i.e., Algorithm 1) on the following quadratic program:

$$\operatorname{Min}_{\mathbf{x}} \sum_{i=1}^{n}\left(\mathbf{x}_i^\intercal\mathbf{A}_i\mathbf{x}_i + 2\mathbf{b}_i^\intercal\mathbf{x}_i\right) + \sum_{i=1}^{n}\sum_{j\in\Omega_i}\beta_{ij}\|\mathbf{x}_i - \mathbf{x}_j\|_2^2 \tag{31}$$

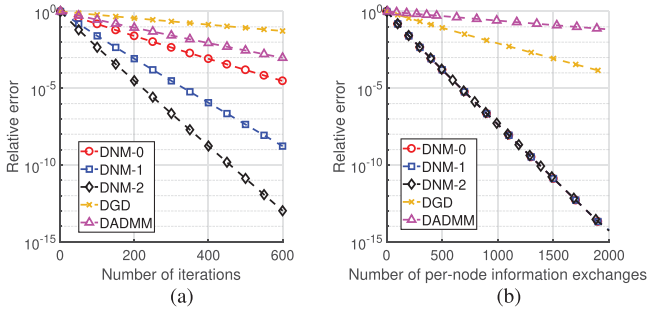


Fig. 1. Comparison between the proposed distributed Newton's method ($K = 0, 1, 2$), the DGD, and the distributed ADMM ($d = 2$). (a) Relative error versus number of iterations. (b) Relative error versus number of per-node information exchanges.

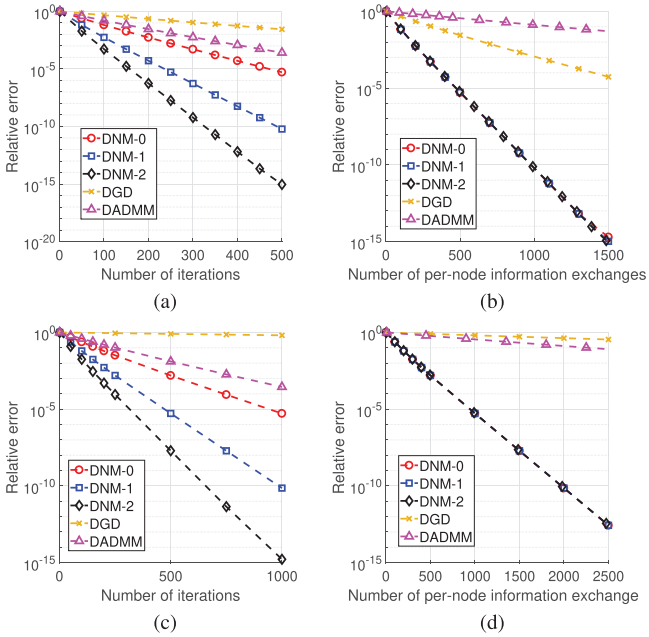


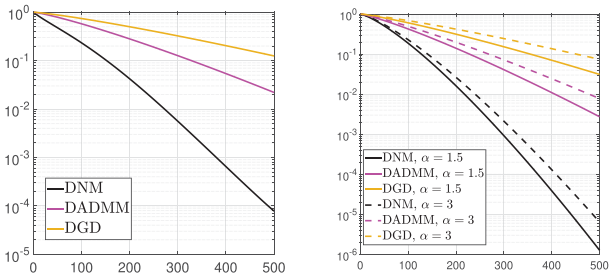
Fig. 2. Impact of the condition number on the performance of the proposed distributed Newton's method ($K = 0, 1, 2$), the DGD, and the distributed ADMM. (a) Relative error versus number of iterations for $d = 1$. (b) Relative error versus number of per-node information exchanges for $d = 1$. (c) Relative error versus number of iterations for $d = 3$. (d) Relative error versus number of per-node information exchanges for $d = 3$.

where $\mathbf{A}_i \in \mathbb{S}^p_{++}$ is some positive definite matrix and $\mathbf{b}_i \in \mathbb{R}^p$. $\beta_{ij} > 0$ is some positive constant controlling the proximity between neighbors' variables. Problem (31) has broad applications in many signal processing scenarios. For instance, consider a sensor network in which each node i uses linear regression to estimate some unknown vector $\mathbf{x}_i$. If we want to enforce the prior knowledge that neighboring nodes have similar unknown vectors, the corresponding optimization problem will be in the form of (31). In fact, as real-time dynamic variants of problem (31), multitask adaptive learning has been studied extensively in the recent literature [27].

Problem (31) is in the form of generic network cost minimization problem (2) by setting $f_i(\mathbf{x}_i) = \mathbf{x}_i^\mathsf{T}\mathbf{A}_i\mathbf{x}_i + 2\mathbf{b}_i^\mathsf{T}\mathbf{x}_i$ and $g_{ij}(\mathbf{x}_i, \mathbf{x}_j) = \beta_{ij}\|\mathbf{x}_i - \mathbf{x}_j\|_2^2, \forall i, j \in \Omega_i$. In the following experiments, we set $\mathbf{A}_i$ to be a diagonal matrix with the first $\frac{p}{2}$ diagonal entries uniformly and randomly chosen from $\{1, 10^{-1}, \ldots, 10^{-d}\}$ and the last $\frac{p}{2}$ diagonal entries uniformly and randomly chosen from $\{1, 10, \ldots, 10^d\}$. Here, d is a positive integer controlling the condition number of the node cost function f_i: the larger the d, the more ill-conditioned the cost functions. In addition, entries of $\mathbf{b}_i$ are uniformly and randomly chosen from the interval [0,1], while β_{ij} are uniformly and randomly selected from the interval [0.5, 1.5]. We set the network topology to be a random graph (links are uniformly and randomly generated) with $n = 100$ nodes and average degree of 4. The dimension of the decision variables is $p = 20$. The stepsize ϵ is chosen to be 1 unless otherwise noted. This values of ϵ is chosen empirically to roughly optimize the performance of the distributed Newton's method. For comparison purposes, we also apply the DGD [13], [34] and the distributed ADMM (DADMM) [28], [35] to the quadratic program (31). When implementing the DADMM for the quadratic program (31), we use direct closed-form solutions to compute the iterates instead of using numerical solvers. The ADMM parameter ρ is chosen to be 9 to empirically optimize the performance of the DADMM. The performance of the proposed DNM-K ($K = 0, 1, 2$), the DGD, and the DADMM is shown in Fig. 1 for $d = 2$. The relative errors $\frac{\|\mathbf{x}_t - \mathbf{x}^*\|_2}{\|\mathbf{x}^*\|_2}$ versus the number of iterations and the number of per-node information exchanges are shown in Fig. 1(a) and (b), respectively. Here, one unit of information exchange is the transmission of one p-dimensional vector. The numbers of per-node (node i) information exchanges for the proposed DNM, the DGD, and the DADMM are $K + 1$, 1, and $2|\Omega_i| + 1$, respectively. In our network topology, the average node degree is 4 so that the average number of per-node information exchanges for the DADMM is 9.

From the results in Fig. 1, we can first see the effect of K, i.e., the approximation order of the Hessian matrix, on the performance of the DNM. From Fig. 1(a), we observe that the DNM converges faster with respect to the number of iterations for larger values of K. This is reasonable as larger K implies more accurate approximation of the Hessian matrix in the DNM (cf. Lemma 3). From Fig. 1(b), an interesting observation is that DNM-K's ($K = 0, 1, 2$) have virtually the same convergence curve with respect to the number of per-node information exchanges. This suggests that K does not affect the performance of DNM much as far as communication complexity is concerned. Second, we remark that the DNM outperforms the DGD significantly in terms of both the number of iterations and the number of information exchanges. Specifically, to achieve the same relative error, the number of iterations and the number of information exchanges needed by the DGD is larger than those needed by the DNM-2 by an order of magnitude. Third, the DNM also outperforms the DADMM remarkably, especially in terms of number of information exchanges. In particular, to achieve the same relative error, the number of per-node information exchanges needed by the DADMM is larger than those needed by the DNM by almost two orders of magnitude. These comparisons demonstrate the advantage of the DNM, a second-order optimization method, over other first-order primal or primal/dual optimization methods such as the DGD and the DADMM.

Next, we examine the impact of the condition number (controlled by d) on the performance of the DNM, the DGD, and the DADMM. The performance of these algorithms with respect to the number of iterations and the number of per-node information exchanges is shown in Fig. 2 for both $d = 1$ and $d = 3$. First, we remark that for either value of d, the DNM always remarkably outperforms the DGD and the DADMM in terms of both the number of iterations and the number of information exchanges. Second, we observe that the DNM is much more robust to large condition number than the DGD. In particular, when the condition number increases, i.e., when d increases from 1 to 3, to achieve the same relative error, the number of iterations or information exchanges needed by the DNM increases by twice while that needed by the DGD increases by around 15 times. This observation is analogous to the classical one for centralized Newton's method and gradient descent stating that the



(a) (b)

Fig. 3. Problems with nonquadratic cost functions. (a) Problem (32). (b) Problem (33).

latter is much more sensitive to the condition number of the objective function than the former [29]. Our observation extends this property to the distributed network cost minimization problem (2).

We further consider another network cost minimization problem as follows:

$$\begin{aligned}\min_{\mathbf{x}} \; & -\sum_{i=1}^{n}\sum_{l=1}^{p} a_{il}\log(x_{il}) \\ & +\sum_{i=1}^{n}\sum_{j\in\Omega_i}\log\left(\sum_{l=1}^{p}\left(e^{b_{il}x_{il}}+e^{b_{jl}x_{jl}}\right)\right)\end{aligned} \tag{32}$$

where x_{il} is the lth entry of $\mathbf{x}_i$, and $\{a_{il}, b_{il}\}$ are nonnegative constants. Problem (32) can be used to model the resource allocation problem in a communication network, in which each user has p types of resources (e.g., power and bandwidth) to consume. The term $\log(x_{il})$ is the utility of user i if she consumes x_{il} amount of resource l and the link cost $\log(\sum_{l=1}^{p}(e^{b_{il}x_{il}}+e^{b_{jl}x_{jl}}))$ can model the cost of mutual interference between the neighbors i, j. Unlike Problem (31), the node cost and link cost functions of Problem (32) are not quadratic. The performance of distributed Newton's method ($K = 2, \epsilon = 0.5$) on Problem (32) is shown in Fig. 3(a), in which the network is a random graph with average degree equal to 3. For comparison, we also show the performance of DADMM and DGD. It can be observed that the convergence for Problem (32) is slower than that of the quadratic program (31). This is the consequence of the more complicated objective function (logarithms and exponentials) in (32) than the quadratic functions in (31). The proposed distributed Newton's method still outperforms DADMM and DGD for the nonquadratic problem (32). Additionally, we consider another network cost minimization problem with a different form of link cost functions as follows:

$$\min_{\mathbf{x}} \; -\sum_{i=1}^{n}\sum_{l=1}^{p} a_{il}\log(x_{il}) + \sum_{i=1}^{n}\sum_{j\in\Omega_i}\sum_{l=1}^{p}(b_{ij,l}x_{il}+b_{ji,l}x_{jl})^{\alpha} \tag{33}$$

where $\{a_{il}, b_{ij,l}\}$ are positive constants and $\alpha \geq 1$ is constant. The link cost functions are not quadratic as long as $\alpha \neq 2$ and can be used to model the cost of mutual interference in different scenarios by tuning α. For $\alpha = 1.5$ and $\alpha = 3$, we show the performance of the distributed Newton's method, DADMM, and DGD in Fig. 3(b). It can be observed that the distributed Newton's method still outperforms DADMM and DGD consistently. These experiments corroborate the advantage of distributed Newton's method for problems with nonquadratic cost functions.

V. Communication and Computational Complexity

In this section, we discuss about the communication and computational complexity of the proposed distributed Newton's method (DNM) and alternative first-order optimization methods including the DGD, the distributed ADMM (DADMM), and the distributed linearized ADMM (DLADMM) [28] for solving the network cost minimization problem (2).

A. Communication Complexity

At each iteration of the DNM, each node i needs to broadcast $K+1$ vectors of p dimension, namely $\mathbf{x}_{t,i}, \mathbf{d}_{0,t,i}, \ldots, \mathbf{d}_{K-1,t,i}$, to its neighbors. Therefore, the per-iteration communication complexity of the DNM increases linearly with the approximation order K. This observation suggests a complexity-accuracy tradeoff for the choice of K in the DNM. In particular, increasing the value of K will enhance the approximation accuracy (and thus per-iteration performance) of the DNM and incur higher communication burden simultaneously. This tradeoff for choices of K has been studied empirically through simulations in Section IV. It is observed that though the per-iteration performance of the DNM enhances with increasing K, its per-information-exchange performance is insensitive to K for $K = 0, 1, 2$. As a comparison, in the DGD for problem (2), each node only needs to broadcast one p-dimensional vector at each iteration. Moreover, in DADMM or DLADMM for problem (2), each node i broadcasts $2|\Omega_i|+1$ p-dimensional vectors at each iteration [28]. We note that the number of information exchanges for the DADMM or DLADMM depends on the degree of the node since there are primal/dual link variables in the reformulated problem of (2) suitable for application of the ADMM (cf. Algorithm 1 in [28]). To achieve the same performance, the advantage of the DNM over the aforementioned first-order optimization methods (DGD and DADMM) in terms of communication complexity has been highlighted through numerical experiments in Section IV.

B. Computational Complexity

In the DNM, each node needs to evaluate not only the gradients but also the Hessian matrices of the local node/link cost functions. Besides, each node needs to compute the inversion of a $p \times p$ matrix (i.e., $\mathbf{D}_{t,ii}^{-1}$ in (13) and (14)) at each iteration. In contrast, in the DGD and the DLADMM [28], every node only needs to evaluate the gradients of the local cost functions and is free of any matrix inversion. Furthermore, the computational burden of the DADMM can be very high in general because, in each iteration, each node needs to solve a nonlinear optimization problem numerically. In contrast, the proposed distributed Newton's method is free of solving any optimization subproblems and thus enjoys lower computational complexity or shorter execution time than DADMM generally. Nevertheless, in some special cases such as the quadratic program (31), the DADMM iterates can be computed in closed form and do not need to resort to numerical solvers. In such cases, the computational complexity or the execution time of the DADMM is also low, similar to the distributed Newton's method. These comparisons suggest that, relative to the first-order optimization methods, the superior convergence performance of the DNM comes at the expense of moderately high computational complexity. This is analogous to the complexity-accuracy tradeoff between classical Newton's method and gradient descent algorithm in the centralized setting [29].

VI. Conclusion

In this article, a novel generic network cost minimization problem incorporating both node costs and link costs is studied. A distributed Newton's method (Algorithm 1) is proposed to solve the network cost minimization problem in a decentralized manner by splitting and approximating the Hessian matrix of the objective function appropriately. Under some standard technical assumptions, we theoretically establish the global linear convergence of Algorithm 1 to the optimal point (Theorem 1). Furthermore, we show that Algorithm 1 possesses a quadratic convergence phase over a certain time interval (Theorem 2). Numerical experiments are carried out to corroborate the effectiveness of Algorithm 1, which outperforms other first-order primal or primal/dual optimization methods remarkably and is robust to ill-conditioned cost functions. Complexity issues of the proposed distributed Newton's method and alternative first-order methods are also discussed.

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] 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.

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

[6] M. Chiang, S. H. Low, A. R. Calderbank, and J. C. Doyle, "Layering as optimization decomposition: A mathematical theory of network architectures," *Proc. IEEE*, vol. 95, no. 1, pp. 255–312, Jan. 2007.

[7] D. P. Palomar and M. Chiang, "A tutorial on decomposition methods for network utility maximization," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 8, pp. 1439–1451, Aug. 2006.

[8] M. Chiang, "Balancing transport and physical layers in wireless multihop networks: Jointly optimal congestion control and power control," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 1, pp. 104–116, Jan. 2005.

[9] D. P. Palomar and M. Chiang, "Alternative distributed algorithms for network utility maximization: Framework and applications," *IEEE Trans. Autom. Control*, vol. 52, no. 12, pp. 2254–2269, Dec. 2007.

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

[11] 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.

[12] 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.

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

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

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

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

[17] 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.

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

[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] T.-H. Chang, W.-C. Liao, M. Hong, and X. Wang, "Asynchronous distributed ADMM for large-scale optimization—Part II: Linear convergence analysis and numerical performance," *IEEE Trans. Signal Process.*, vol. 64, no. 12, pp. 3131–3144, Jun. 2016.

[23] 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. 2014.

[24] 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.

[25] F. Mansoori and E. Wei, "Superlinearly convergent asynchronous distributed network Newton method," in *Proc. IEEE 56th Annu. Conf. Decis. Control*, Melbourne, Australia 2017, pp. 2874–2879.

[26] M. Eisen, A. Mokhtari, and A. Ribeiro, "Decentralized quasi-Newton methods," *IEEE Trans. Signal Process.*, vol. 65, no. 10, pp. 2613–2628, May 2017.

[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] X. Cao and K. J. R. Liu, "Distributed linearized ADMM for network cost minimization," *IEEE Trans. Signal Inf. Process. Over Netw.*, vol. 4, no. 3, pp. 626–638, Sep. 2018.

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

[30] M. Zargham, A. Ribeiro, A. Ozdaglar, and A. Jadbabaie, "Accelerated dual descent for network flow optimization," *IEEE Trans. Autom. Control*, vol. 59, no. 4, pp. 905–920, Apr. 2014.

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

[32] R. Carli, G. Notarstefano, L. Schenato, and D. Varagnolo, "Analysis of Newton-Raphson consensus for multi-agent convex optimization under asynchronous and lossy communications," in *Proc. 54th IEEE Conf. Decis. Control*, Osaka, Japan 2015, pp. 418–424.

[33] 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.

[34] S. Boyd, L. Xiao, and A. Mutapcic, "Subgradient methods," *Lecture Notes of EE392o, Stanford University, Autumn Quarter*, vol. 2004, 2003.

[35] 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.