Generic Network Cost Minimization: A Decentralized Newton's Method

Xuanyu Cao and K. J. Ray Liu

***Abstract*—In this work, we examine a 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 the Newton's method. 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 and 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.**

***Index Terms*—Decentralized optimization, Newton's method, network cost minimization**

I. Introduction

The advancement of decentralized signal processing and control in multi-agent systems relies on the development of various distributed optimization methods. Owing to its importance, distributed optimization over networks has been extensively studied in the literature. One important category of distributed optimization problems 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 [1]. Moreover, consensus optimization has been studied by using the distributed Nesterov gradient algorithm in [2] and the distributed alternating direction method of multipliers (ADMM) in [3]. Later, variants of the distributed ADMM have been proposed for consensus optimization [4]–[7]. In addition, second order optimization algorithm based on Newton's method has been proposed for consensus optimization in [8].

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, i.e., 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 [9], [10], 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 [9], [10]. Therefore, we are motivated to examine 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 [8], we develop a distributed variant of Newton's method for the generic network cost minimization problem in this paper. 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. 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 [11], [12], a quadratic convergence phase of the algorithm over a certain time interval is identified (Theorem 2). Numerical experiments on quadratic programming are implemented to corroborate the effectiveness of the proposed algorithm, which outperforms alternative first order optimization methods significantly in terms of both convergence time and number of per-node information exchanges.

The organization of the rest of this paper 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. We conclude this work in Section V.

Notations: Denote $\{1,2,...,n\}$ 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 $\mathbf{A}\in\mathbb{R}^{n\times n}$, i.e., $\rho(\mathbf{A})=\max_{i\in[n]}|\lambda_i(\mathbf{A})|$, 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.

978-1-5386-0579-0/18/$31.00 ©2018 IEEE

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, i.e., 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:

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

We note that the consensus optimization problems in [1]–[8], [13] are special cases of the network cost minimization problem (1) 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. 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 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, e.g., $\|\mathbf{x}_i - \mathbf{x}_j\|_2^2$ in multitask adaptive networks in [9], [10].

Inspired by the recent work [8] on network Newton algorithm for consensus optimization, we develop a distributed variant of Newton's method for the generic network cost minimization problem (1), which takes link costs into account and encompasses consensus optimization as a special case. We make the following two technical assumptions which are standard in the literature of numerical optimization [11], [12].

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

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

$$m\mathbf{I} \preceq \nabla^2 g_{ij}(\mathbf{x}_i, \mathbf{x}_j) \preceq M\mathbf{I}. \tag{3}$$

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, i.e., for any $i \in [n], j \in \Omega_i$ and $\mathbf{x}_i, \mathbf{x}_i', \mathbf{x}_j, \mathbf{x}_j'$:*

$$\|\nabla^2 f_i(\mathbf{x}_i) - \nabla^2 f_i(\mathbf{x}_i')\|_2 \le L\|\mathbf{x}_i - \mathbf{x}_i'\|_2, \tag{4}$$

$$\|\nabla^2 g_{ij}(\mathbf{x}_i, \mathbf{x}_j) - \nabla^2 g_{ij}(\mathbf{x}_i', \mathbf{x}_j')\|_2 \le L \left\| \begin{bmatrix} \mathbf{x}_i \\ \mathbf{x}_j \end{bmatrix} - \begin{bmatrix} \mathbf{x}_i' \\ \mathbf{x}_j' \end{bmatrix} \right\|_2 \tag{5}$$

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 (1) 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, i.e., Assumption 1. In the rest of the paper, 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 (1), we compute the gradient of F as follows:

$$[\nabla F(\mathbf{x})]_i = \nabla f_i(\mathbf{x}_i) + \sum_{j\in\Omega_i} [\nabla_{\mathbf{x}_i} g_{ij}(\mathbf{x}_i, \mathbf{x}_j) + \nabla_{\mathbf{x}_i} g_{ji}(\mathbf{x}_j, \mathbf{x}_i)]. \tag{6}$$

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

$$[\mathbf{H}(\mathbf{x})]_{ik} = \begin{cases} \nabla^2 f_i(\mathbf{x}_i) + \sum_{j\in\Omega_i} [\nabla^2_{\mathbf{x}_i} g_{ij}(\mathbf{x}_i, \mathbf{x}_j) + \nabla^2_{\mathbf{x}_i} g_{ji}(\mathbf{x}_j, \mathbf{x}_i)], & \text{if } i = k, \\ \nabla^2_{\mathbf{x}_i,\mathbf{x}_k} g_{ik}(\mathbf{x}_i, \mathbf{x}_k) + \nabla^2_{\mathbf{x}_i,\mathbf{x}_k} g_{ki}(\mathbf{x}_k, \mathbf{x}_i), & \text{if } k \in \Omega_i, \\ \mathbf{0}, & \text{otherwise.} \end{cases} \tag{7}$$

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 $\mathbf{D}(\mathbf{x})$:

$$[\mathbf{D}(\mathbf{x})]_{ik} = \begin{cases} \nabla^2 f_i(\mathbf{x}_i) + 2\sum_{j\in\Omega_i} [\nabla^2_{\mathbf{x}_i} g_{ij}(\mathbf{x}_i, \mathbf{x}_j) + \nabla^2_{\mathbf{x}_i} g_{ji}(\mathbf{x}_j, \mathbf{x}_i)], & \text{if } i = k, \\ \mathbf{0}, & \text{otherwise,} \end{cases} \tag{8}$$

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

$$[\mathbf{B}(\mathbf{x})]_{ik} = \begin{cases} \sum_{j\in\Omega_i} [\nabla^2_{\mathbf{x}_i} g_{ij}(\mathbf{x}_i, \mathbf{x}_j) + \nabla^2_{\mathbf{x}_i} g_{ji}(\mathbf{x}_j, \mathbf{x}_i)], & \text{if } i = k, \\ -\nabla^2_{\mathbf{x}_i,\mathbf{x}_k} g_{ik}(\mathbf{x}_i, \mathbf{x}_k) - \nabla^2_{\mathbf{x}_i,\mathbf{x}_k} g_{ki}(\mathbf{x}_k, \mathbf{x}_i), & \text{if } k \in \Omega_i, \\ \mathbf{0}, & \text{otherwise.} \end{cases} \tag{9}$$

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 $\mathbf{H}(\mathbf{x}) = \mathbf{D}(\mathbf{x})^{\frac{1}{2}} \left[\mathbf{I} - \mathbf{D}(\mathbf{x})^{-\frac{1}{2}} \mathbf{B}(\mathbf{x}) \mathbf{D}(\mathbf{x})^{-\frac{1}{2}}\right] \mathbf{D}(\mathbf{x})^{\frac{1}{2}}$. To invoke Newton's method, we need to calculate $\mathbf{H}(\mathbf{x})^{-1} = \mathbf{D}(\mathbf{x})^{-\frac{1}{2}} \left[\mathbf{I} - \mathbf{D}(\mathbf{x})^{-\frac{1}{2}} \mathbf{B}(\mathbf{x}) \mathbf{D}(\mathbf{x})^{-\frac{1}{2}}\right]^{-1} \mathbf{D}(\mathbf{x})^{-\frac{1}{2}}$.

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 $\rho\left(\mathbf{D}(\mathbf{x})^{-\frac{1}{2}}\mathbf{B}(\mathbf{x})\mathbf{D}(\mathbf{x})^{-\frac{1}{2}}\right) < 1$ (which will be shown later in Section III), we can rewrite $\mathbf{H}(\mathbf{x})^{-1}$ as:

$$\mathbf{H}(\mathbf{x})^{-1} = \mathbf{D}(\mathbf{x})^{-\frac{1}{2}} \sum_{k=0}^{\infty} \left[\mathbf{D}(\mathbf{x})^{-\frac{1}{2}}\mathbf{B}(\mathbf{x})\mathbf{D}(\mathbf{x})^{-\frac{1}{2}}\right]^k \mathbf{D}(\mathbf{x})^{-\frac{1}{2}}. \tag{10}$$

Truncating the first $K+1$ ($K \geq 0$) terms of the summation in (10), we note that $\mathbf{D}(\mathbf{x})^{-\frac{1}{2}} \sum_{k=0}^{K} \left[\mathbf{D}(\mathbf{x})^{-\frac{1}{2}}\mathbf{B}(\mathbf{x})\mathbf{D}(\mathbf{x})^{-\frac{1}{2}}\right]^k \mathbf{D}(\mathbf{x})^{-\frac{1}{2}}$ is still positive definite. As such, we can define a positive definite approximated Hessian $\hat{\mathbf{H}}(\mathbf{x})$:

$$\hat{\mathbf{H}}(\mathbf{x}) := \left\{\mathbf{D}(\mathbf{x})^{-\frac{1}{2}} \sum_{k=0}^{K} \left[\mathbf{D}(\mathbf{x})^{-\frac{1}{2}}\mathbf{B}(\mathbf{x})\mathbf{D}(\mathbf{x})^{-\frac{1}{2}}\right]^k \mathbf{D}(\mathbf{x})^{-\frac{1}{2}}\right\}^{-1}. \tag{11}$$

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 k-th ($k \geq 0$) order approximated Hessian matrix $\hat{\mathbf{H}}_k(\mathbf{x})$:

$$\hat{\mathbf{H}}_k(\mathbf{x}) := \left\{\mathbf{D}(\mathbf{x})^{-\frac{1}{2}} \sum_{l=0}^{k} \left[\mathbf{D}(\mathbf{x})^{-\frac{1}{2}}\mathbf{B}(\mathbf{x})\mathbf{D}(\mathbf{x})^{-\frac{1}{2}}\right]^l \mathbf{D}(\mathbf{x})^{-\frac{1}{2}}\right\}^{-1}. \tag{12}$$

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:

$$\mathbf{d}_{k+1,t} = -\mathbf{D}_t^{-\frac{1}{2}}\left[\mathbf{I} + \sum_{l=1}^{k+1}\left(\mathbf{D}_t^{-\frac{1}{2}}\mathbf{B}_t\mathbf{D}_t^{-\frac{1}{2}}\right)^l\right]\mathbf{D}_t^{-\frac{1}{2}}\mathbf{h}_t. \tag{13}$$

$$= -\mathbf{D}_t^{-1}\mathbf{h}_t + \mathbf{D}_t^{-1}\mathbf{B}_t\mathbf{d}_{k,t} \tag{14}$$

$$= \mathbf{D}_t^{-1}(\mathbf{B}_t\mathbf{d}_{k,t} - \mathbf{h}_t) \tag{15}$$

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

$$\mathbf{d}_{k+1,t,i} = \mathbf{D}_{t,ii}^{-1}\left(\sum_{j\in\Omega_i\cup\{i\}} \mathbf{B}_{t,ij}\mathbf{d}_{k,t,j} - \mathbf{h}_{t,i}\right). \tag{16}$$

Equation (16) 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 (1) can be developed and the proposed algorithm is detailed in Algorithm 1 from the perspective of node i.

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

$$\mathbf{D}_{t,ii} = \nabla^2 f_i(\mathbf{x}_{t,i}) + 2\sum_{j\in\Omega_i}\left[\nabla^2_{\mathbf{x}_i} g_{ij}(\mathbf{x}_{t,i},\mathbf{x}_{t,j}) + \nabla^2_{\mathbf{x}_i} g_{ji}(\mathbf{x}_{t,j},\mathbf{x}_{t,i})\right], \tag{17}$$

$$\mathbf{B}_{t,ii} = \sum_{j\in\Omega_i}[\nabla^2_{\mathbf{x}_i} g_{ij}(\mathbf{x}_{t,i},\mathbf{x}_{t,j}) + \nabla^2_{\mathbf{x}_i} g_{ji}(\mathbf{x}_{t,j},\mathbf{x}_{t,i})], \tag{18}$$

$$\mathbf{B}_{t,ij} = -\nabla^2_{\mathbf{x}_i,\mathbf{x}_j} g_{ij}(\mathbf{x}_{t,i},\mathbf{x}_{t,j}) - \nabla^2_{\mathbf{x}_i,\mathbf{x}_j} g_{ji}(\mathbf{x}_{t,j},\mathbf{x}_{t,i}), \quad \forall j \in \Omega_i, \tag{19}$$

$$\mathbf{h}_{t,i} = \nabla f_i(\mathbf{x}_{t,i}) + \sum_{j\in\Omega_i}\left[\nabla_{\mathbf{x}_i} g_{ij}(\mathbf{x}_{t,i},\mathbf{x}_{t,j}) + \nabla_{\mathbf{x}_i} g_{ji}(\mathbf{x}_{t,j},\mathbf{x}_{t,i})\right], \tag{20}$$

$$\mathbf{d}_{0,t,i} = -\mathbf{D}_{t,ii}^{-1}\mathbf{h}_{t,i}. \tag{21}$$

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

$$\mathbf{d}_{k+1,t,i} = \mathbf{D}_{t,ii}^{-1}\left(\sum_{j\in\Omega_i\cup\{i\}} \mathbf{B}_{t,ij}\mathbf{d}_{k,t,j} - \mathbf{h}_{t,i}\right). \tag{22}$$

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**

III. Convergence Analysis

In this section, we analyze the convergence properties of the proposed distributed Newton's method for network cost minimization, i.e., 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 second order optimization methods over first order ones [11], [12], [14].

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, which will be frequently used in the development of later results. Define $C = \max_{i\in[n]} |\Omega_i|$ to be the maximum node degree.

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

$$\mathbf{0} \preceq \mathbf{B}(\mathbf{x}) \preceq 2MC\mathbf{I}, \tag{23}$$

$$m\mathbf{I} \preceq \mathbf{H}(\mathbf{x}) \preceq M(1+2C)\mathbf{I}, \tag{24}$$

$$m\mathbf{I} \preceq \mathbf{D}(\mathbf{x}) \preceq (1+4C)M\mathbf{I}. \tag{25}$$

The proof is Lemma 1 is omitted. In order to ensure that the series in (10) 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, whose proof is also omitted.

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

$$\mathbf{0} \preceq \mathbf{D}(\mathbf{x})^{-\frac{1}{2}}\mathbf{B}(\mathbf{x})\mathbf{D}(\mathbf{x})^{-\frac{1}{2}} \preceq \eta\mathbf{I}, \tag{26}$$

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

$$\rho\left(\mathbf{D}(\mathbf{x})^{-\frac{1}{2}}\mathbf{B}(\mathbf{x})\mathbf{D}(\mathbf{x})^{-\frac{1}{2}}\right) \leq \eta < 1. \tag{27}$$

Lemma 2 guarantees the convergence of the series in (10) and justifies the truncated approximation of Hessian in (11). 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:

$$\mathbf{E}(\mathbf{x}) := \mathbf{I} - \hat{\mathbf{H}}(\mathbf{x})^{-\frac{1}{2}}\mathbf{H}(\mathbf{x})\hat{\mathbf{H}}(\mathbf{x})^{-\frac{1}{2}}. \tag{28}$$

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 $\mathbf{x} \in \mathbb{R}^{np}$:*

$$\mathbf{0} \preceq \mathbf{E}(\mathbf{x}) \preceq \eta^{K+1}\mathbf{I}. \tag{29}$$

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 (22), i.e., 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 $\mathbf{x} \in \mathbb{R}^{np}$:*

$$\gamma_1\mathbf{I} \preceq \hat{\mathbf{H}}(\mathbf{x})^{-1} \preceq \gamma_2\mathbf{I}, \tag{30}$$

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 $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^{np}$:*

$$\|\mathbf{H}(\mathbf{x}) - \mathbf{H}(\mathbf{x}')\|_2 \leq L(1+2C)\|\mathbf{x} - \mathbf{x}'\|_2., \tag{31}$$

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

Based on Lemmas 3, 4 and 5, we are able to show the first main theorem regarding the global linear convergence of Algorithm 1, whose proof is omitted due to space limitation.

Theorem 1. *If the step size ϵ of Algorithm 1 is chosen such that:*

$$0 < \epsilon < \min\left\{1, \sqrt{\frac{2m\gamma_1}{L(1+2C)\gamma_2^3(2M(1+2C))^{\frac{3}{2}}\sqrt{F(\mathbf{x}_0) - F(\mathbf{x}^*)}}}\right\}, \tag{32}$$

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

$$0 \leq F(\mathbf{x}_t) - F(\mathbf{x}^*) \leq \xi^t[F(\mathbf{x}_0) - F(\mathbf{x}^*)], \tag{33}$$

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

$$\xi = 1 - m\gamma_1(2\epsilon - \epsilon^2) + \frac{\epsilon^3}{2}L(1+2C)\gamma_2^3(2M(1+2C))^{\frac{3}{2}}\sqrt{F(\mathbf{x}_0) - F(\mathbf{x}^*)}. \tag{34}$$

B. The Quadratic Convergence Phase

A classical theoretical explanation of the advantage of second order optimization methods (e.g., Newton's method) over first order alternatives (e.g., gradient descent method) is that the former possesses a quadratic convergence region [11], [12], [14], 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.

Define a sequence $\psi_t = (1 - \epsilon + \epsilon\eta^{K+1})(1 + \mu_1\xi^{\frac{t-1}{4}})$. Suppose ϵ satisfies the condition (32) 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 can state our main theorem regarding the quadratic convergence phase of Algorithm 1 as follows.

Theorem 2. *Let ϵ be chosen in accordance with the condition (32). Suppose there exists a time interval $[t_1, t_2]$ with $t_1 \geq t_0$ such that, for any $t \in [t_1, t_2]$:*

$$\frac{\sqrt{\psi_t}(1-\sqrt{\psi_t})}{\mu_2} \leq \left\|\mathbf{D}_{t-1}^{-\frac{1}{2}}\mathbf{h}_t\right\|_2 \leq \frac{1-\sqrt{\psi_t}}{\mu_2}. \tag{35}$$

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

$$F(\mathbf{x}_t) - F(\mathbf{x}^*) \leq \frac{\delta^{2^{t-t_1}}}{\mu_2\sqrt{\gamma_1}}\|\mathbf{x}_t - \mathbf{x}^*\|_2, \tag{36}$$

where $\delta := \frac{\mu_2}{1-\sqrt{\psi_{t_1}}}\left\|\mathbf{D}_{t_1-1}^{-\frac{1}{2}}\mathbf{h}_{t_1}\right\|_2 \in [0,1)$ 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 $\lim_{\tau\to\infty}\left\|\mathbf{D}_{\tau-1}^{-\frac{1}{2}}\mathbf{h}_\tau\right\|_2 = 0$.

Remark 1. *From $\lim_{t\to\infty}\left\|\mathbf{D}_{t-1}^{-\frac{1}{2}}\mathbf{h}_t\right\|_2 = 0$, $\lim_{t\to\infty}\frac{\sqrt{\psi_t}(1-\sqrt{\psi_t})}{\mu_2} = \frac{\sqrt{1-\epsilon+\epsilon\eta^{K+1}}(1-\sqrt{1-\epsilon+\epsilon\eta^{K+1}})}{\mu_2} > 0$ 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 $\left\|\mathbf{D}_{t-1}^{-\frac{1}{2}}\mathbf{h}_t\right\|_2$ will eventually be smaller than both bounds in (35) for large enough t. Typically, as t increases, $\left\|\mathbf{D}_{t-1}^{-\frac{1}{2}}\mathbf{h}_t\right\|_2$*

will first become smaller than the right bound of (35)*, but still remain larger than the left bound of* (35)*, i.e.,* (35) *holds. Theorem* (2) *says, in such a case, Algorithm 1 converges quadratically. After that, as* t *further increases,* $\left\|\mathbf{D}_{t-1}^{-\frac{1}{2}}\mathbf{h}_t\right\|_2$ *becomes even smaller than the left bound of* (35) *so that* (35) *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

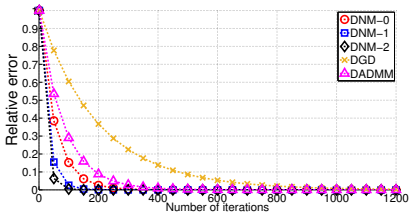
In this section, we empirically investigate the performance of the proposed distributed Newton's method (DNM, i.e., Algorithm 1) on the following quadratic program:

$$\text{Minimize}_{\mathbf{x}} \sum_{i=1}^{n}\left(\mathbf{x}_i^{\mathsf{T}}\mathbf{A}_i\mathbf{x}_i+2\mathbf{b}_i^{\mathsf{T}}\mathbf{x}_i\right)+\sum_{i=1}^{n}\sum_{j\in\Omega_i}\beta_{ij}\|\mathbf{x}_i-\mathbf{x}_j\|_2^2, \tag{37}$$

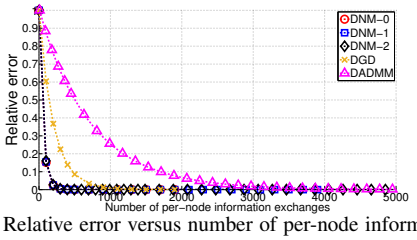
where $\mathbf{A}_i\in\mathbb{S}_{++}^p$ is some positive definite matrix and $\mathbf{b}_i\in\mathbb{R}^p$. $\beta_{ij}>0$ is some positive constant controlling the proximity between neighbors' variables. Problem (37) has broad applications in many signal processing scenarios, such as multitask adaptive signal processing [9], [10].

Problem (37) is in the form of generic network cost minimization problem (1) 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 $\{1,10^{-1},...,10^{-d}\}$ and the last $\frac{p}{2}$ diagonal entries uniformly and randomly chosen from $\{1,10,...,10^{d}\}$. 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 $p=20$. For comparison purposes, we also apply the distributed gradient descent (DGD) [1], [15] and the distributed alternating direction method of multipliers (DADMM) [16], [17] to the quadratic program (37). 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 Fig. 1-(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



(a) Relative error versus number of iterations

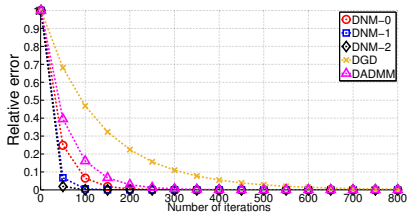


(b) Relative error versus number of per-node information exchanges

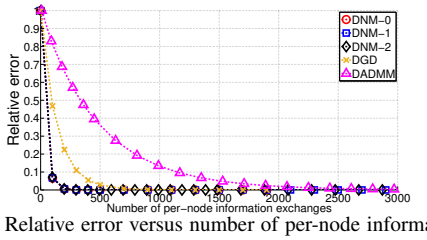
Fig. 1: Comparison between the proposed distributed Newton's method ($K=0,1,2$), the distributed gradient descent and the distributed ADMM ($d=2$).

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 (c.f. 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.

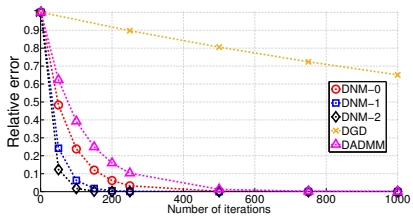
Next, we examine the impact of the condition number (controlled by d) on the performance of the DNM, the DGD and the DADMM. The performance of these algorithms with respect to the number of iterations and the number of per-node information exchanges are shown in Fig. 2 for both $d=1$ and $d=3$. First, 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



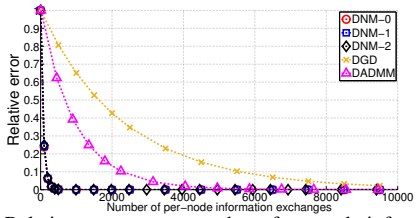
(a) Relative error versus number of iterations for $d = 1$



(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 $d = 3$

Fig. 2: Impact of the condition number on the performance of the proposed distributed Newton's method ($K = 0, 1, 2$), the distributed gradient descent and the distributed ADMM.

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 obervation is analogous to the classical one for centralized Newton's method and gradient descent stating that the latter is much more sensitive to the condition number of the objective function than the former [11]. Our observation extends this property to the distributed network cost minimization problem (1).

V. Conclusion

In this paper, 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. We establish the global linear convergence and a quadratic convergence phase of Algorithm 1 theoretically. 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.

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] T.-H. Chang, M. Hong, W.-C. Liao, and X. Wang, "Asynchronous distributed admm for large-scale optimizationpart i: Algorithm and convergence analysis," *IEEE Transactions on Signal Processing*, vol. 64, no. 12, pp. 3118–3130, 2016.

[8] A. Mokhtari, Q. Ling, and A. Ribeiro, "Network newton distributed optimization methods," *IEEE Transactions on Signal Processing*, vol. 65, no. 1, pp. 146–161, 2017.

[9] 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.

[10] J. Chen, C. Richard, and A. H. Sayed, "Diffusion lms over multitask networks," *IEEE Transactions on Signal Processing*, vol. 63, no. 11, pp. 2733–2748, 2015.

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

[12] D. P. Bertsekas, *Nonlinear programming*. Athena scientific Belmont, 1999.

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

[14] A. Mokhtari, W. Shi, Q. Ling, and A. Ribeiro, "A decentralized second-order method with exact linear convergence rate for consensus optimization," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 2, no. 4, pp. 507–522, 2016.

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

[16] X. Cao and K. J. R. Liu, "Distributed linearized admm for network cost minimization," *arXiv preprint arXiv:1702.03367*, 2017.

[17] 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.