A Distributed Newton Method for Network Utility Maximization

Ermin Wei[†] , Asuman Ozdaglar[†], and Ali Jadbabaie[‡]

Abstract— **Most existing work uses dual decomposition and subgradient methods to solve Network Utility Maximization (NUM) problems in a distributed manner, which suffer from slow rate of convergence properties. This work develops an alternative distributed Newton-type fast converging algorithm for solving network utility maximization problems with self-concordant utility functions. By using novel matrix splitting techniques, both primal and dual updates for the Newton step can be computed using iterative schemes in a decentralized manner with limited information exchange. Similarly, the stepsize can be obtained via an iterative consensus-based averaging scheme. We show that even when the Newton direction and the stepsize in our method are computed within some error (due to finite truncation of the iterative schemes), the resulting objective function value still converges superlinearly to an explicitly characterized error neighborhood. Simulation results demonstrate significant convergence rate improvement of our algorithm relative to the existing subgradient methods based on dual decomposition.**

I. Introduction

Most of today's communication networks are large-scale and comprise of agents with local information and heterogeneous preferences, making centralized control and coordination impractical. This motivates much interest in developing and studying distributed algorithms for various problems, including but not limited to Internet packets routing, collaboration in sensor networks, and cross-layer communication network design. This work focuses on the rate control problem in wireline networks, also referred to as the *Network Utility Maximization (NUM)* problem in the literature (see [5], [11], [12]). In NUM problems, the network topology and routes are predetermined, each source in the network has a local utility, which is a function of the rate at which it sends information over the network. The objective is to determine the source rates in the network that maximize the sum of the utilities, subject to link capacity constraints. The standard approach for solving NUM problems relies on using dual decomposition and subgradient (or first-order) methods, which through a dual price exchange mechanism yields algorithms that operate on the basis of local information [10], [12], [13]. One major shortcoming of this approach is the slow rate of convergence.

In this paper we propose a novel Newton-type second order method for solving the NUM problem in a distributed manner, which is significantly faster in convergence. Our method involves transforming the inequality constrained NUM problem to an equality-constrained one through introducing slack variables and using logarithmic barrier functions, and using an equality-constrained Newton method for the reformulated problem. There are two challenges for implementing this method in a distributed manner. First challenge is the computation of the Newton direction. This computation involves matrix inversion, which is both costly and requires global information. In order to avoid this, the linear system is solved by utilizing an iterative scheme based on novel matrix splitting techniques and dual price exchange scheme. The second challenge is the global information required in the computation of the stepsize. This is resolved by using a consensus-based local averaging scheme.[1]

Since our algorithm uses iterative schemes to compute the stepsize and the Newton direction, exact computation is not feasible. Another major contribution of our work is to consider truncated version of these schemes and present convergence rate analysis of the constrained Newton method when the stepsize and the Newton direction are estimated with some error. We show that when these errors are sufficiently small, the value of the objective function converges superlinearly to a neighborhood of the optimal objective function value, whose size is explicitly quantified as a function of the errors and bounds on them.

Other than the papers cited above, our paper is related to [3] and [8]. In [3], the authors have developed a distributed Newton-type method for the NUM problem using belief propagation algorithm. While the belief propagation algorithm is known to converge in most practical applications, there is no provable guarantee. Our paper differs from this by developing a standalone distributed Newton-type algorithm and providing analysis for the convergence properties thereof. Similarly, [8] considers applying distributed Newton problem to an equality-constrained network optimization problem under Lipschitz assumptions. Our analysis is novel in that we have an inequality-constrained problem and we do not impose Lipschitz condition, instead we will utilize properties of self-concordant functions.

The rest of the paper is organized as follows: Section II defines the problem formulation and equivalent transformations thereof. Section III presents the exact constrained primal-dual Newton method for this problem. Section IV presents a distributed iterative scheme for computing the dual Newton step and the distributed inexact Newton-type algorithm. Section V analyzes the rate of convergence of our algorithm. Section VI presents simulation results to demonstrate convergence speed

This research was funded in part by National Science Foundation grants DMI-0545910, and by the DARPA ITMANET program.

[†]Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology

[‡]Electrical and Systems Engineering University of Pennsylvania

[1]Consensus-based schemes have been used extensively in recent literatures as distributed mechanisms for aligning the values held by multiple agents, see [7], [8], [15], [16], [17]

improvement of our algorithm to the existing methods with linear convergence rates. Section VII contains our concluding remarks.

Due to space constraints, some of the proofs of the results in this paper are omitted. We refer the reader to [19] for the missing proofs.

Basic Notation and Notions:

A vector is viewed as a column vector, unless clearly stated otherwise. We write $\mathbb{R}_+$ to denote the set of nonnegative real numbers, i.e., $\mathbb{R}_+ = [0, \infty)$. We denote by x_i the i^{th} component of a vector x. When $x_i \geq 0$ for all components i of a vector x, we write $x \geq 0$. For a matrix A, we write A_{ij} to denote the matrix entry in the i^{th} row and j^{th} column. We write $I(n)$ to denote the identity matrix of dimension $n \times n$. We use x' to denote the transpose of a vector x. For a real-valued function $f : \mathbb{R}^n \to \mathbb{R}$, the gradient vector and the Hessian matrix of f at $x \in \mathbb{R}^n$ are denoted by $\nabla f(x)$, and $\nabla^2 f(x)$ respectively.

A real-valued convex function $g : \mathbb{R} \to \mathbb{R}$ is said to be *self-concordant* if $|g'''(x)| \leq 2g''(x)^{\frac{3}{2}}$ for all x in its domain. For real-valued functions in $\mathbb{R}^n$, a convex function $g : \mathbb{R}^n \to \mathbb{R}$ is self-concordant if it is self-concordant along every direction in its domain, i.e., if the function $\tilde{g}(t) = g(x + tv)$ is self-concordant in t for all x and v. Operations that preserve self-concordance property include summing, scaling by a factor $\alpha \geq 1$, and composition with affine transformation (see [4] for more details).

II. Network Utility Maximization Problem

We consider a network represented by a set $\mathcal{L} = \{1, ..., L\}$ of directed links of finite capacity given by $c = [c_l]_{l \in \mathcal{L}}$, where these links form a strongly connected graph. The network is shared by a set $\mathcal{S} = \{1, ..., S\}$ of sources, each of which transmits information along a predetermined route. For each link l, let $S(l)$ denote the set of sources using it. For each source i, let $L(i)$ denote the set of links it uses. We also denote the nonnegative source rate vector by $s = [s_i]_{i \in \mathcal{S}}$. The capacity constraint at the links can be compactly expressed as $Rs \leq c$, where R is the *routing matrix* of dimension $L \times S$, i.e.,

$$R_{ij} = \begin{cases} 1 & \text{if link } i \text{ is on the route for source } j, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

We associate a utility function $U_i : \mathbb{R}_+ \to \mathbb{R}$ with each source i, i.e., $U_i(s_i)$ denotes the utility of source i as a function of the source rate s_i. We assume the utility functions are additive, such that the overall utility of the network is given by $\sum_{i=1}^{S} U_i(s_i)$. Thus the Network Utility Maximization(NUM) problem can be formulated as

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{S} U_i(s_i) \\ \text{subject to} \quad & Rs \leq c, \\ & s \geq 0. \end{aligned} \tag{2}$$

We adopt the following standard assumption.

Assumption 1: The utility functions $U_i : \mathbb{R}_+ \to \mathbb{R}$ are strictly concave, monotonically nondecreasing, twice continuously differentiable, and self-concordant.

To facilitate development of a distributed Newton-type method, we reformulate the problem into one with only equality constraints, by introducing nonnegative slack variables $[y_l]_{l \in \mathcal{L}}$, such that $\sum_{j=1}^{S} R_{lj} s_j + y_l = c_l$ for $l = 1, 2 \ldots L$, and using logarithmic barrier functions for nonnegativity constraints. We denote the new decision variable vector by $x = ([s_i]'_{i \in \mathcal{S}}, [y_l]'_{l \in \mathcal{L}})'$. Problem (2) then can be rewritten as

$$\begin{aligned} \text{minimize} \quad & -\sum_{i=1}^{S} U_i(x_i) - \mu \sum_{i=1}^{S+L} \log(x_i) \\ \text{subject to} \quad & Ax = c, \end{aligned} \tag{3}$$

where $A = [R \ I(L)]$, and μ is a nonnegative constant coefficient for the barrier functions. We denote by p^* the optimal objective value for the equality constrained problem (3). Notice that by Assumption 1 and the properties of logarithmic functions, the objective function for problem (3),

$$f(x) = -\sum_{i=1}^{S} U_i(x_i) - \mu \sum_{i=1}^{S+L} \log(x_i),$$

is separable, strictly convex, twice continuously differentiable, and has a positive definite diagonal Hessian matrix. The function $f(x)$ is also self-concordant for $\mu \geq 1$, since it is a sum of self-concordant functions.

One can show that as the coefficient μ approaches 0, the optimal solution of problem (3) approaches that of problem (2) [2]. Therefore, in this paper, our goal is to investigate *iterative distributed methods* for solving problem (3) for a fixed μ. In order to preserve the self-concordant property of the function f, which will be used to prove convergence of our distributed algorithm, we assume the coefficient $\mu \geq 1$ for the rest of the paper.

III. Exact Newton Method

We consider solving problem (3) using a (feasible start) equality-constrained Newton method (see [4] Chapter 10). In our iterative method, we use x^k to denote the solution vector at the k^{th} step.

A. Feasible Initialization

To initialize the algorithm, we start with some feasible and strictly positive vector $x^0 > 0$. For example, one possible such choice is given by

$$x_i^0 = \frac{\min_k\{c_k\}}{S+1} \quad \text{for } i = 1, 2 \ldots S,$$

$$x_{i+S}^0 = c_i - \sum_{j=1}^{S} R_{ij} \frac{\min_k\{c_k\}}{S+1} \quad \text{for } i = 1, 2 \ldots L,$$

where c_k is the finite capacity for link k, S is the total number of sources in the network, and R is routing matrix [cf. Eq. (1)].

B. Iterative Update Rule

Given an initial feasible vector x^0, the algorithm generates the iterates by

$$x^{k+1} = x^k + s^k \Delta x^k, \tag{4}$$

where s^k is a positive stepsize, Δx^k is the Newton direction given as the solution to the following system of linear equations:[2]

$$\begin{pmatrix} \nabla^2 f(x^k) & A' \\ A & 0 \end{pmatrix} \begin{pmatrix} \Delta x^k \\ w^k \end{pmatrix} = - \begin{pmatrix} \nabla f(x^k) \\ 0 \end{pmatrix}. \tag{5}$$

In the rest of the paper, we let $H_k = \nabla^2 f(x^k)$ for notational convenience. The vector $[w_l^k]_{l \in \mathcal{L}}$ are the dual variables for the link capacity constraints. Solving for x^k and w^k in the preceding system yields

$$\Delta x^k = -H_k^{-1}(\nabla f(x^k) + A' w^k), \text{ and} \tag{6}$$

$$(A H_k^{-1} A') w^k = -A H_k^{-1} \nabla f(x^k). \tag{7}$$

Since the objective function f is separable in x_i, the matrix H_k^{-1} is a diagonal matrix with entries $[H_k^{-1}]_{ii} = (\frac{\partial^2 f}{(\partial x_i^k)^2})^{-1}$. Therefore given the vector w^k, the Newton direction Δx^k can be computed using local information. However, the computation of the vector w^k at a given primal solution x^k cannot be implemented in a decentralized manner in view of the fact that the evaluation of the matrix inverse $(A H_k^{-1} A')^{-1}$ requires global information. The following section provides a distributed inexact Newton method, based on an iterative scheme to compute the vector w^k using a decentralized scheme.

IV. Distributed Inexact Newton Method

Our inexact Newton method uses the same initialization as presented in Section III-A, however, it computes the dual variables and the primal direction using a distributed iterative scheme with some error. The construction of these schemes relies on novel ideas from matrix splitting, which we introduce as follows.

A. Preliminaries on Matrix Splitting

Matrix splitting can be used to solve a system of linear equations given by $Gy = b$, where G is a matrix of dimension $n \times n$ and b is a vector of length n. Suppose that the matrix G can be expressed as the sum of two matrices M and N, i.e.,

$$G = M + N. \tag{8}$$

Let y_0 be an arbitrary vector of length n. A sequence $\{y^k\}$ can be generated by the following iteration:

$$y^{k+1} = M^{-1} b - M^{-1} N y^k. \tag{9}$$

It can be seen that the sequence $\{y^k\}$ converges as $k \to \infty$ if and only if the spectral radius of the matrix $M^{-1}N$ is strictly bounded above by 1. When the sequence $\{y^k\}$ converges, its limit y^* solves the original linear system, i.e., $Gy^* = b$ (see [1] and [6] for more details). Hence, the key to solve the linear equation via matrix splitting is the bound on the spectral radius of the matrix $M^{-1}N$. Such a bound can be obtained using the following result (see Theorem 2.5.3 from [6]).

Theorem 4.1: Let G be a real symmetric matrix. Let M and N be matrices such that $G = M + N$ and assume that both matrices $M + N$ and $M - N$ are positive definite. Then the spectral radius of $M^{-1}N$, denoted by $\rho(M^{-1}N)$, satisfies $\rho(M^{-1}N) < 1$.

By the above theorem, if G is a real, symmetric and positive definite matrix, then one sufficient condition for the iteration (9) to converge is that the matrix $M - N$ is positive definite. This can be guaranteed using Gershgorin Circle Theorem, which we introduce next (see [18] for more details).

Theorem 4.2: (Gershgorin Circle Theorem) Let G be an $n \times n$ matrix, and define $r_i(G) = \sum_{j \neq i} |G_{ij}|$. Then, each eigenvalue of G lies in one of the Gershgorin sets $\{\Gamma_i\}$, with Γ_i defined as disks in the complex plane, i.e.,

$$\Gamma_i = \{z \in C \mid |z - G_{ii}| \leq r_i(G)\}.$$

One corollary of the above theorem is that if a matrix G is strictly diagonally dominant, i.e., $|G_{ii}| > \sum_{j \neq i} |G_{ij}|$, and $G_{ii} > 0$ for all i, then the real parts of all the eigenvalues lie in the positive half of the real line, and thus the matrix is positive definite. Hence a sufficient condition for the matrix $M - N$ to be positive definite is that $M - N$ is strictly diagonally dominant with strictly positive diagonal entries.

B. Distributed Computation of the Dual Variables

We use the matrix splitting scheme introduced in the preceding section to compute the dual variables w^k in Eq. (7) in a distributed manner. Let D_k be a diagonal matrix, with diagonal entries

$$(D_k)_{ll} = (A H_k^{-1} A')_{ll}, \tag{10}$$

and matrix B_k be given by

$$B_k = A H_k^{-1} A' - D_k. \tag{11}$$

Let matrix $\bar{B}_k$ be a diagonal matrix, with diagonal entries

$$(\bar{B}_k)_{ii} = \sum_{j=1}^{L} (B_k)_{ij}. \tag{12}$$

By splitting the matrix $A H_k^{-1} A'$ as the sum of $D_k + \bar{B}_k$ and $B_k - \bar{B}_k$, we obtain the following result.

Theorem 4.3: For a given $k > 0$, let D_k, B_k, $\bar{B}_k$ be the matrices defined in Eqs. (10), (11) and (12). Let $w(0)$ be an arbitrary initial vector and consider the sequence $\{w(t)\}$ generated by the iteration

$$\begin{aligned} w(t+1) = (D_k + \bar{B}_k)^{-1}(-A H_k^{-1} \nabla f(x^k)) \\ -(D_k + \bar{B}_k)^{-1}(B_k - \bar{B}_k) w(t), \end{aligned} \tag{13}$$

for all $t \geq 0$. Then the sequence $\{w(t)\}$ converges as $t \to \infty$, and its limit is the solution to Eq. (7).

Proof: We use a matrix splitting scheme given by

$$(A H_k^{-1} A') = (D_k + \bar{B}_k) + (B_k - \bar{B}_k) \tag{14}$$

[2]This is essentially a primal-dual method with the vectors Δx^k and w^k acting as primal and dual steps.

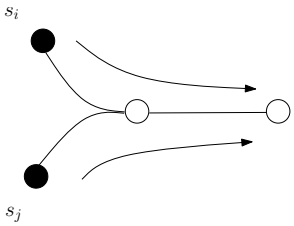

Fig. 1. In step 1, each source sends information to the links it uses.

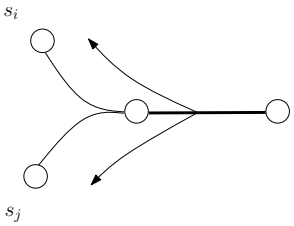

Fig. 2. In step 2, each link sends information back to the sources that use it.

and the iterative scheme presented in Eqs. (8) and (9) to solve Eq. (7). For all k, both the real matrix H_k and its inverse, H_k^{-1}, are positive definite and diagonal. The matrix A has full row rank and is element-wise nonnegative. Therefore the product $AH_k^{-1}A'$ is real, symmetric, element-wise nonnegative and positive definite. We let

$$Q_k = (D_k + \bar{B}_k) - (B_k - \bar{B}_k) = D_k + 2\bar{B}_k - B_k \quad (15)$$

denote the difference matrix. By definition of $\bar{B}_k$ [cf. Eq. (12)], the matrix $2\bar{B}_k - B_k$ is diagonally dominant, with nonnegative diagonal entries. Due to strict positivity of the second derivatives of the logarithmic barrier functions, we have $(D_k)_{ii} > 0$ for all i. Therefore the matrix Q_k is strictly diagonally dominant. By Theorem 4.2, such matrices are positive definite. Therefore, by Theorem 4.1, the spectral radius of the matrix $(D_k + \bar{B}_k)^{-1}(B_k - \bar{B}_k)$ is strictly bounded above by 1. Hence the splitting scheme (14) guarantees the sequence $\{w(t)\}$ generated by iteration (13) to converge to the solution of Eq. (7). ■

There can be many ways to split the matrix $AH_k^{-1}A'$, the particular one in Eq. (14) is chosen here due to two desirable features. First it guarantees that the difference matrix Q_k [cf. Eq. (15)] is strictly diagonally dominant, and hence ensures convergence of the sequence $\{w(t)\}$. Second, with this splitting scheme, the matrix $D_k + \bar{B}_k$ is diagonal, which eliminates the need for global information and computational complexity when calculating its inverse matrix.

We next describe a computation and information exchange procedure to show that with this splitting scheme w^k can be computed in a distributed manner. Given an arbitrary vector of dual variables $w(0)$, the iterates $w(t)$ are generated by:

Step 1: Each source i sends its gradient $\nabla_i f(x^k)$ (the i^{th} component of the gradient vector $\nabla f(x^k)$), Hessian H_{ii}^k, routing information $[A_{ij}]_{j=1\ldots L}$, and the vector of dual variables $[w_l(t)]_{l\in L(i)}$ to the links it is using, i.e., $l \in L(i)$. The direction of information flow in this step is depicted in Figure 1.

Step 2: Each link l computes $(D_k)_{ll}$ [cf. Eq. (10)], the l^{th} row of the matrix B_k [cf. Eq. (11)] and $(\bar{B}_k)_{ll}$ [cf. Eq. (12)], and updates $w_l(t)$ according to iteration (13). Link l then sends the updated dual variable $w_l(t+1)$ to the sources that use link l, i.e. $i \in S(l)$. Figure 2 illustrates the direction of information flow in this step.

It can be seen that the above procedure can be implemented in a distributed manner, see [19] for more details.

Remarks:

1. The sources only need to send their computed gradient and Hessian information once per dual variable calculation, since those values are constant during the iterations.
2. The routing information only needs to be sent once in the entire algorithm, unless there is dynamic network topology variation.
3. This algorithm requires slightly more information exchange than the existing first order algorithms applied to the NUM problem (2) (see [10], [12], [13] for more details), in which only the sum of dual variables of links along a source path is fed back to the source. This mild extra information exchange, however, speeds up the algorithm significantly, as we show in Section V.

C. Distributed Computation of the Newton Primal Direction

After obtaining the vector of dual variables w^k using the above scheme, based on Eq. (6) primal Newton direction Δx^k can be calculated in a distributed way. However, because our distributed dual variable computation involves an iterative scheme, the exact value for w^k is not available, which implies the resulting Newton direction may violate the equality constraint in problem (3). Therefore, the calculation for the *inexact Newton direction*, which we denote by $\Delta\tilde{x}^k$, is separated into two stages to maintain feasibility.

In the first stage, the first S components of $\Delta\tilde{x}^k$ is computed via Eq. (6) using the dual variables obtained in the preceding section. Then in the second stage, the last L components of $\Delta\tilde{x}^k$, corresponding to the slack variables, are solved explicitly by the links to guarantee the condition $A\Delta\tilde{x}^k = 0$ is satisfied. This calculation can be easily performed due to the nature of slack variables and the system is guaranteed to have a solution because the matrix A has full row rank and $\Delta\tilde{x}^k$ can be negative.

Our distributed Newton-type algorithm is defined as: starting from an initial feasible vector x^0, the primal solution x is updated as follows,

$$x^{k+1} = x^k + s^k\Delta\tilde{x}^k, \quad (16)$$

where s^k is a positive stepsize, and $\Delta\tilde{x}^k$ is the inexact Newton direction at the k^{th} iteration.

We refer to the exact solution to the system of equations (5) the *exact Newton direction*, denoted by Δx^k. The inexact Newton direction $\Delta\tilde{x}^k$ from our algorithm is a feasible estimate of Δx^k. At a given primal vector x^k, we define the *exact Newton decrement* $\lambda(x^k)$ as

$$\lambda(x^k) = \sqrt{(\Delta x^k)'\nabla^2 f(x^k)\Delta x^k}. \quad (17)$$

Similarly, the *inexact Newton decrement* $\tilde{\lambda}(x^k)$ is given by

$$\tilde{\lambda}(x^k) = \sqrt{(\Delta\tilde{x}^k)'\nabla^2 f(x^k)\Delta\tilde{x}^k}. \quad (18)$$

Observe that both $\lambda(x^k)$ and $\tilde{\lambda}(x^k)$ are nonnegative and well defined, due to the fact that the matrix $\nabla^2 f(x^k)$ is positive definite.

Our stepsize choice will be based on the inexact Newton decrement $\tilde{\lambda}(x^k)$, as we will show in Section V, this choice can ensure rate of convergence of our algorithm. Therefore, we first need to compute $\tilde{\lambda}(x^k)$ in a distributed way. Notice that the inexact Newton decrement can be viewed as the norm of inexact Newton direction $\Delta\tilde{x}^k$, weighted by the Hessian matrix $\nabla^2 f(x^k)$. Therefore, the inexact Newton decrement $\tilde{\lambda}(x^k)$ can be computed via a distributed iterative averaging consensus-based scheme. Due to space constraints, we omit the details of the consensus algorithm, interested readers should refer to [16], [7], [15] for further information. We denote the computed value for $\tilde{\lambda}(x^k)$ from consensus algorithm as θ^k. The stepsize in our algorithm is given by

$$s^k = \begin{cases} \frac{c}{\theta^k+1} & \text{if} \quad \theta^k \geq \frac{1}{4}, \\ 1 & \text{otherwise}, \end{cases} \tag{19}$$

where c is some positive scalar that satisfies $\frac{5}{6} < c < 1$. The lower bound $\frac{5}{6}$ is chosen here to guarantee $x^k > 0$ for all k, and also convergence of the algorithm, as will show in Theorem 4.5 and Section V-C respectively.

Due to the iterative nature of our algorithm in both primal and dual domains, in practice infinite precision of the dual variable vector w^k, primal direction Δx^k and stepsize choice s^k cannot be achieved. There are three sources of inexactness in the algorithm. First is the iterative computation of the dual variable w^k, which in turn affects the primal Newton direction. Second source of error stems from the way we maintain feasibility in the algorithm. Finally, stepsize s^k depends on the value of θ^k, which is an inexact estimate for $\tilde{\lambda}(x^k)$ obtained via an iterative scheme. We quantify the bounds on these errors as follows.

Assumption 2: For all k, the inexact Newton direction $\Delta\tilde{x}^k$ produced by our algorithm can be written as $\Delta x^k = \Delta\tilde{x}^k + \gamma$, where γ is bounded by $|\gamma' \nabla^2 f(x^k)\gamma| \leq p^2(\Delta\tilde{x}^k)'\nabla^2 f(x^k)\Delta\tilde{x}^k + \epsilon$. for some positive scalars $p < 1$ and ϵ.

This assumption imposes a bound on the weighted norm of the Newton direction error γ as a function of the weighted norm of $\Delta\tilde{x}^k$ and a constant ϵ. Note that without the constant ϵ, we would require the error to vanish when x^k is close to the optimal solution, *i.e.* when $\Delta\tilde{x}^k$ is very small, which is impractical for implementation purpose.

We bound the error in the inexact Newton decrement calculation as follows.

Assumption 3: Denote the error in the Newton decrement calculation as τ^k, i.e., $\tau^k = \tilde{\lambda}(x^k) - \theta^k$, then for all k, τ^k satisfies $|\tau^k| \leq \left(\frac{1}{c} - 1\right)\frac{5}{4}$.

The constant $\frac{5}{4}$ is chosen here to ensure our objective function f is well defined throughout the algorithm. For the rest of the paper, we assume the conditions in Assumptions 1-3 hold.

We now show that the stepsize choice in (19) will guarantee positivity of the primal variable, i.e., $x^k > 0$, which in turn ensures that the logarithmic barrier functions in the objective function of problem (3) are well defined. We proceed by first establishing a bound on the error in the stepsize calculation.

Lemma 4.4: Let $\tilde{\lambda}(x^k)$ be the inexact Newton decrement defined in (18), θ^k be the computed value of $\tilde{\lambda}(x^k)$ and c, satisfying $\frac{5}{6} < c < 1$, be the constant used in stepsize choice (19). For $\theta^k \geq \frac{1}{4}$, the following relation holds

$$(2c-1)/(\tilde{\lambda}(x^k)+1) \leq \frac{c}{\theta^k+1} \leq 1/(\tilde{\lambda}(x^k)+1). \tag{20}$$

This lemma follows from Assumption 3 and the fact that $\frac{5}{6} < c < 1$. With this bound on the error in the stepsize calculation, we can show that starting with a positive feasible solution, the primal variable generated by our algorithm remains positive for all k, i.e., $x^k > 0$.

Theorem 4.5: Let x^0 be a positive feasible primal variable, x^k be the sequence of primal variables updated using iteration (16), i.e., $x^{k+1} = x^k + s^k\Delta\tilde{x}^k$, where $\Delta\tilde{x}^k$ be the inexact Newton direction defined in Section IV-C, and s^k is defined as in (19). Then for all k, the primal variable satisfies $x^k > 0$.

The main step in proving the above theorem is to establish a bound on the stepsize given by $s^k \leq \min_i |\frac{x_i^k}{\Delta x_i^k}|$.

V. Convergence Analysis

We next present our convergence analysis for the inexact Newton algorithm defined in Eq. (16). For the k^{th} iteration, we define the function $\tilde{f}_k : \mathbb{R} \to \mathbb{R}$ as

$$\tilde{f}_k(t) = f(x^k + t\Delta\tilde{x}^k), \tag{21}$$

which is self-concordant, because the objective function f is self-concordant. Note that the value $\tilde{f}_k(0)$ and $\tilde{f}_k(s^k)$ are the objective function values at x^k and x^{k+1} respectively. Therefore $\tilde{f}_k(s^k) - \tilde{f}_k(0)$ measures the decrease in objective function value at the k^{th} iteration. Before proceeding further, we first introduce some relevant background information on self-concordant functions and properties of the Newton decrement, both of which will be used extensively in our convergence analysis.

A. Preliminaries

Using the definition of a self-concordant function, we can relate the Newton decrement at the current step and the next step in an unconstrained Newton method through the following lemma. This lemma extends results in [9] and [14] to allow inexactness in the Newton direction and reflects the effect of the error at the current step at the Newton decrement of the next step.

Lemma 5.1: Let $f : \mathbb{R}^n \to \mathbb{R}$ be a self-concordant convex function. Consider the unconstrained optimization problem $\min_{x\in\mathbb{R}^n} f(x)$. Let Δx be the exact Newton direction at x. Let $\Delta\tilde{x}$ denote any direction with $\gamma = \Delta x - \Delta\tilde{x}$, and $x(t) = x + t\Delta\tilde{x}$ for $t \in [0,1]$. Let z be the exact Newton direction at $x + \Delta\tilde{x}$. If $\tilde{\lambda} = (\Delta\tilde{x}'\nabla^2 f(x)\Delta\tilde{x})^{\frac{1}{2}} < 1$, then we have the following relation,

$$z\nabla^2 f(x+\Delta\tilde{x})'z \leq \frac{\tilde{\lambda}^2}{1-\tilde{\lambda}}\sqrt{z'\nabla^2 f(x)z} + |\gamma'\nabla^2 f(x)'z|.$$

The above lemma will be used to guarantee quadratic rate of convergence for our distributed inexact Newton method [cf. Section V-D]. The next lemma plays a central role in

relating the suboptimality gap in the objective function value and the exact Newton decrement (see [4] for more details).

Lemma 5.2: Let $f : \mathbb{R}^n \to \mathbb{R}$ be a self-concordant function, and assume that Δx^k is the exact Newton direction at x^k. Let $\lambda(x^k)$ be the exact Newton decrement, defined as $\lambda(x^k) = \sqrt{(\Delta x^k)' \nabla^2 f(x^k) \Delta x^k}$. Let p^* denote the optimal objective function value. If $\lambda(x^k) \leq 0.68$, then the following relation holds,

$$p^* \geq f(x^k) - \lambda(x^k)^2. \tag{22}$$

The number 0.68 is obtained based on numerical simulation by [4]. The above lemmas are established for the unconstrained Newton method. However as shown in [4], they can also be applied for the constrained Newton method. We will use extensively these lemmas in the subsequent sections for rate of convergence analysis for our algorithm.

B. Basic Relations

We first introduce some key relations, which provides a bound on the error in the Newton direction computation. This will be used for both phases of the convergence analysis.

Lemma 5.3: Let $\tilde{\lambda}(x^k)$ be the inexact Newton decrement defined in Eq. (18). Then the following relation holds for all k:

$$|\gamma' \nabla^2 f(x^k) \Delta \tilde{x}^k| \leq p \tilde{\lambda}(x^k)^2 + \tilde{\lambda}(x^k) \sqrt{\epsilon},$$

where γ, p, and ϵ are the nonnegative scalars defined in Assumption 2.

The preceding lemma follows from generalized Cauchy-Schwarz inequality and Assumption 2. Using this lemma, the following basic relation can be established, which will be used to measure the improvement in the objective function value.

Lemma 5.4: Let $\tilde{f}_k(t)$ and $\tilde{\lambda}(x^k)$ be the functions defined in Eqs. (21) and (18) respectively. Then the following relation holds for all k with $0 \leq t < 1/\tilde{\lambda}(x^k)$,

$$\begin{aligned} \tilde{f}_k(t) \leq & \tilde{f}_k(0) - t(1-p)\tilde{\lambda}(x^k)^2 \\ & - (1-\sqrt{\epsilon}) t \tilde{\lambda}(x^k) - \log(1 - t\tilde{\lambda}(x^k)), \end{aligned} \tag{23}$$

where p, and ϵ are the nonnegative scalars defined in Assumption 2.

This lemma follows from the property of self-concordant functions and Lemma 5.3. By choosing the stepsize t carefully, the preceding lemma can guarantee a constant lower bound in the improvement in the objective function value at each iteration. We present the convergence properties of our algorithm in the following two sections.

C. Damped Convergent Phase

In this section, we consider the case when $\theta^k \geq \frac{1}{4}$ and stepsize $s^k = \frac{c}{\theta^k + 1}$ [cf. Eq. (19)]. We will prove the improvement in the objective function value is lower bounded by a constant. To this end, we first establish the improvement bound for the exact stepsize choice of $t = 1/(\tilde{\lambda}(x^k) + 1)$.

Theorem 5.5: Let $\tilde{f}_k$ be the function defined in Eq. (21), and $\tilde{\lambda}(x^k)$ be the inexact Newton decrement defined in Eq. (18). Let p and ϵ be the scalars defined in Assumption 2. Assume that $0 < p < \frac{1}{2}$ and $0 < \epsilon < \left(\frac{(0.5-p)(6c-5)}{4c}\right)^2$, where c is the constant in stepsize choice [cf. Eq. (19)]. Then for $\theta^k \geq \frac{1}{4}$ and $t = 1/(\tilde{\lambda}(x^k)+1)$, there exist a scalar $\alpha > 0$, such that the following relation holds,

$$\tilde{f}_k(t) - \tilde{f}_k(0) \leq -\frac{\alpha(1+p)\left(\frac{6c-5}{4c}\right)^2}{\left(1 + \frac{6c-5}{4c}\right)} \tag{24}$$

Proof: For notational simplicity, let $y = \tilde{\lambda}(x^k)$ in this proof. We will show that for any positive scalar α, such that $0 < \alpha \leq (\frac{1}{2} - p - \frac{4c\sqrt{\epsilon}}{6c-5})/(p+1)$, relation (24) holds. Such α exists by the fact that $\epsilon < (\frac{(0.5-p)(6c-5)}{4c})^2$.

Using the facts that $\alpha \leq (\frac{1}{2} - p - \frac{4c\sqrt{\epsilon}}{6c-5})/(p+1)$ and $c > \frac{5}{6}$, the inequality $\sqrt{\epsilon} \leq \frac{6c-5}{4c}(\frac{1}{2} - p - \alpha(1+p))$ is satisfied.

Also by Assumption 3, we have for $\theta^k \geq \frac{1}{4}$,

$$y \geq \theta^k - (\frac{1}{c} - 1)\frac{5}{4} \geq \frac{1}{4} - (\frac{1}{c} - 1)\frac{5}{4} = \frac{6c-5}{5c} > 0, \tag{25}$$

where the strict inequality follows from the fact that $c > \frac{5}{6}$. Hence the preceding two relations imply that $\sqrt{\epsilon} \leq y(\frac{1}{2} - p - \alpha(1+p))$. Using algebraic manipulation, this yields,

$$-(1-p)y - (1-\sqrt{\epsilon}) + (1+y) - \frac{y}{2} \leq -\alpha(1+p)y.$$

From relation (25), we have $y > 0$. We can therefore multiply by y and divide by $1+y$ both sides of the above inequality to obtain

$$-\frac{(1-p)y^2}{1+y} - \frac{(1-\sqrt{\epsilon})y}{1+y} + y - \frac{y^2}{2(1+y)} \leq -\frac{\alpha(1+p)y^2}{1+y} \tag{26}$$

Using second order Taylor expansion on $\log(1+y)$, we have for $y \geq 0$, $\log(1+y) \leq y - \frac{y^2}{2(1+y)}$.

Using this relation in Eq. (26) yields,

$$-\frac{1-p}{1+y}y^2 - \frac{1-\sqrt{\epsilon}}{1+y}y + \log(1+y) \leq -\alpha\frac{(1+p)y^2}{1+y}.$$

Substituting the value of $t = 1/(y+1)$, the above relation can be rewritten as

$$-(1-p)ty^2 - (1-\sqrt{\epsilon})ty - \log(1-ty) \leq -\alpha\frac{(1+p)y^2}{1+y}.$$

Using relation (23) from Lemma 5.4 and definition of y, the preceding relation implies that $\tilde{f}_k(t) - \tilde{f}_k(0) \leq -\alpha(1+p)\frac{y^2}{y+1}$. Observe that the function $h(y) = \frac{y^2}{y+1}$ is monotonically increasing in y, and for $\theta^k \geq \frac{1}{4}$ by relation (25) we have $y \geq \frac{6c-5}{4c}$. Therefore

$$-\alpha(1+p)\frac{y^2}{y+1} \leq -\alpha(1+p)(\frac{6c-5}{4c})^2/(1+\frac{6c-5}{4c}).$$

The preceding two relations shows the desired relation. ■

Note that our algorithm uses the stepsize $s^k = \frac{c}{\theta^k+1}$ for this damped convergent phase, which is an approximation to the stepsize $t = 1/(\tilde{\lambda}(x^k)+1)$ in the previous theorem. The error between the two is bounded by relation (20) as shown in Lemma 4.4. We next show that with this error in the stepsize computation, the improvement in the objective function value in the inexact algorithm is still lower bounded

at each iteration.

Recall that $\tilde{f}_k(t) = f(x^k + t\Delta\tilde{x}^k)$, where the function f is convex. Let $\beta = \frac{s^k}{t}$, where $t = 1/(\tilde{\lambda}(x^k)+1)$, by convexity of the function f, we have that $f(x^k+\beta t\Delta x^k) \leq \beta f(x^k + t\Delta x^k) + (1-\beta)f(x^k)$. Therefore the objective function value improvement is bounded by $f(x+\beta t\Delta x^k) - f(x^k) \leq \beta(\tilde{f}_k(t) - \tilde{f}_k(0))$. Using Lemma 4.4, we obtain bounds on β as $2c-1 \leq \beta \leq 1$. Hence combining this bound with Theorem 5.5, we obtain

$$f(x^{k+1}) - f(x^k) \leq \frac{-(2c-1)\alpha(1+p)(\frac{6c-5}{4c})^2}{(1+\frac{6c-5}{4c})}. \quad (27)$$

Hence in the damped convergent phase we can guarantee a lower bound on the object function value improvement at each iteration. This bound is monotonically increasing in c, therefore the closer the scalar c is to 1, the faster the objective function value improves, however this also requires the error in the inexact Newton decrement calculation, i.e., $\tilde{\lambda}(x^k) - \theta^k$, diminishes to 0 [cf. Assumption 3].

D. Quadratically Convergent Phase

In the phase when $\theta^k < \frac{1}{4}$, we show that the suboptimality diminishes quadratically to a neighborhood of optimal solution. We proceed by first establishing the following lemma for relating the exact and the inexact Newton decrement.

Lemma 5.6: Let p and ϵ be the nonnegative scalars defined in Assumption 2. Let functions λ and $\tilde{\lambda}$ be the exact and inexact Newton decrement defined in Eqs. (17) and (18) respectively. Then the following relation holds:

$$(1-p)\tilde{\lambda}(x^k) - \sqrt{\epsilon} \leq \lambda(x^k) \leq (1+p)\tilde{\lambda}(x^k) + \sqrt{\epsilon}, \quad (28)$$

for all x^k in the domain of the objective function f.

The above lemma follows from Lemmas 5.3 and 5.4. We impose the following bounds on the errors in this phase.

Assumption 4: In the quadratic convergence phase, i.e., when $\theta^k < \frac{1}{4}$, there exists a positive scalar ϕ, such that $\phi \leq 0.267$ and the following relations hold for all k,

$$(1+p)(\theta^k + \tau) + \sqrt{\epsilon} \leq \phi \quad (29)$$

$$p + \sqrt{\epsilon} \leq 1 - (4\phi^2)^{\frac{1}{4}} - \phi, \quad (30)$$

where $\tau > 0$ is a bound on the error in the Newton decrement calculation, i.e., for all k, $|\tau^k| = |\tilde{\lambda}(x^k) - \theta^k| \leq \tau$, and p and ϵ are the scalars defined in Assumption 2.

The upper bound of 0.267 on ϕ is necessary here to guarantee relation (30) can be satisfied by some positive scalars p and ϵ. Relation (29) will be used to guarantee the condition $\lambda(x^k) \leq 0.68$ is satisfied, so that we can use Lemma 5.2 to relate the suboptimality bound with the Newton decrement. Relation (30) will be used for establishing the quadratic rate of convergence of the objective function value.

By Assumption 4, we have $\tilde{\lambda}(x^k) = \theta^k + \tau^k \leq \theta^k + \tau$. Therefore, by relations (28) and (29), we have

$$\lambda(x^k) \leq (1+p)\tilde{\lambda}(x^k) + \sqrt{\epsilon} \leq \phi \leq 0.267. \quad (31)$$

Thus the condition $\lambda(x^k) \leq 0.68$ for Lemma 5.2 is satisfied. We can therefore apply relation (22) to bound suboptimality in our algorithm, i.e., $f(x^k) - p^*$, using the exact Newton decrement. We next show that under this assumption, the objective function value $f(x^k)$ generated by our algorithm converges quadratically to an error neighborhood of the optimal value p^*. We will need the following lemma, which relates the exact Newton decrement at the current and the next step.

Lemma 5.7: Let x^k be the iterates generated by the inexact Newton method [cf. Section IV]. Let λ and $\tilde{\lambda}$ be the exact and inexact Newton decrement defined in Eqs. (17) and (18) respectively. Let θ^k be the computed inexact value of $\tilde{\lambda}$ and let Assumption 4 hold. Then for all k with $\theta^k < \frac{1}{4}$, we have

$$\lambda(x^{k+1}) \leq v\lambda(x^k)^2 + \xi, \quad (32)$$

where $\xi = \frac{\phi p + \sqrt{\epsilon}}{1-p-\phi-\sqrt{\epsilon}} + \frac{2\phi\sqrt{\epsilon}+\epsilon}{(1-p-\phi-\sqrt{\epsilon})^2}$, $v = \frac{1}{(1-p-\phi-\sqrt{\epsilon})^2}$ and p and ϵ are the scalars defined in Assumption 2.

The above lemma can be established using Assumptions 2 and 4, and Lemma 5.1 and it can be used to show that our algorithm converges quadratically to an error neighborhood of optimality, with the error quantified as in the next theorem.

Theorem 5.8: Let λ and $\tilde{\lambda}$ be the exact and inexact Newton decrement defined in Eqs. (17) and (18) respectively. Let $f(x^k)$ be the objective function value at k^{th} iteration for the algorithm defined in Section IV and p^* be the optimal objective function value for problem (3). Let Assumption 4 hold. Let $\xi = \frac{\phi p + \sqrt{\epsilon}}{1-p-\phi-\sqrt{\epsilon}} + \frac{2\phi\sqrt{\epsilon}+\epsilon}{(1-p-\phi-\sqrt{\epsilon})^2}$, $v = \frac{1}{(1-p-\phi-\sqrt{\epsilon})^2}$. Assume that for some $\delta \in [0, 1/2)$, $\xi + v\xi \leq \frac{\delta}{4v}$. Then for all k with $\theta^k < \frac{1}{4}$, we have for $m > 0$,

$$\lambda(x^{k+m}) \leq \frac{1}{2^{2^m}v} + \xi + \frac{\delta}{v}\frac{2^{2^m-1}-1}{2^{2^m}}, \quad (33)$$

and $\text{limsup}_{m\to\infty} f(x^{k+m}) - p^* \leq \xi + \frac{\delta}{2v}$.

Proof: We prove relation (33) by induction. First for $m = 1$, by relation (30), we obtain $(1-p-\phi-\sqrt{\epsilon})^4 \geq 4\phi^2$. Using the definition of v, i.e., $v = \frac{1}{(1-p-\phi-\sqrt{\epsilon})^2}$, the above relation implies $v\phi^2 \leq \frac{1}{4v}$. Using relation (32) and (31), we have $\lambda(x^{k+1}) \leq v\lambda(x^k)^2 + \xi \leq v\phi^2 + \xi \leq \frac{1}{4v} + \xi$. This establishes relation (33) for $m = 1$.

We next assume that relation (33) holds for some some $m > 0$, and show that it also holds for $m+1$. By relation (32), we have $\lambda(x^{k+m+1}) \leq v\left(\frac{1}{2^{2^m}v} + \xi + \frac{\delta}{v}\frac{2^{2^m-1}-1}{2^{2^m}}\right)^2 + \xi$.

Using algebraic manipulations and the assumption that $\xi + v\xi \leq \frac{\delta}{4v}$, this yields $\lambda(x^{k+m+1}) \leq \frac{1}{2^{m+1}v} + \xi + \frac{\delta}{v}\frac{2^{2^{m+1}-1}-1}{2^{2^{m+1}}}$, thus completing the proof of relation (33).

Using relation (31), we have $\lambda(x^k) \leq \phi \leq 0.68$, we can therefore apply Lemma 5.2, and obtain an upper bound on suboptimality as follows, $f(x^{k+m}) - p^* \leq \left(\lambda(x^{k+m})\right)^2 \leq \lambda(x^{k+m}$. Combine this with relation (33), we obtain $f(x^{k+m}) - p^*v \leq \frac{1}{2^{2^m}v} + \xi + \frac{\delta}{v}\frac{2^{2^m-1}-1}{2^{2^m}}$. Taking limit superior on both sides of the preceding relation establishes the final result. ∎

The above theorem shows that the objective function value $f(x^k)$ generated by our algorithm converges quadratically to a neighborhood of the optimal value p^*, with the neighborhood of size $\xi + \frac{\delta}{2v}$, where $\xi = \frac{\phi p + \sqrt{\epsilon}}{1-p-\phi-\sqrt{\epsilon}} + \frac{2\phi\sqrt{\epsilon}+\epsilon}{(1-p-\phi-\sqrt{\epsilon})^2}$,

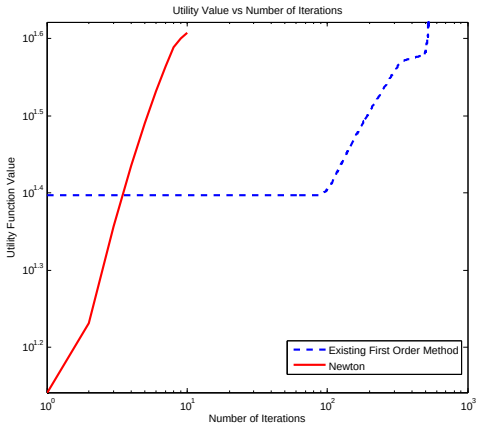

Fig. 3. One sample simulation of distributed Newton algorithm vs existing first order methods on a log scaled graph on a network consisting of 50 nodes.

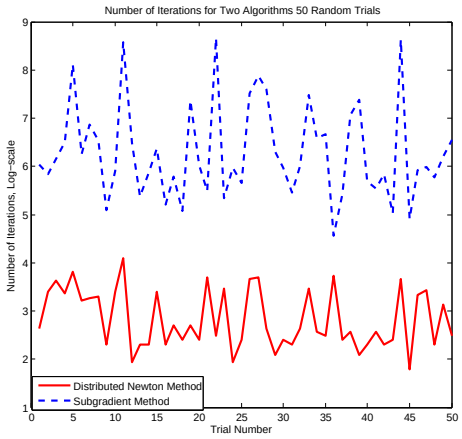

Fig. 4. Distributed Newton and existing first order methods are implemented over 50 randomly generated networks.

$v = \frac{1}{(1-p-\phi-\sqrt{\epsilon})^2}$, and the condition $\xi + v\xi \leq \frac{\delta}{4v}$ is satisfied. Note that with exact Newton algorithm, we have $p = \epsilon = 0$, which implies $\xi = 0$ and we can choose $\delta = 0$, which in turn leads to the size of the error neighborhood being 0. This confirms with the fact that exact Newton algorithm converges quadratically to the optimal objective function value.

VI. Simulation Results

Our simulation results demonstrate that the decentralized Newton method significantly outperforms the existing methods in terms of number of iterations. Here we count both the primal and dual iterations for our Newton method. Figure 3 shows the utility value after each iteration of Newton method and existing subgradient method on log scale for a network size of 50. The results shows newly developed method exhibits significant advantage over the traditional first order ones.

To see more generalized performance over random networks, distributed Newton and conventional first order methods are both implemented over 50 randomly generated networks of random size with the mean of 40, and random number of sources, with the mean of 10. Simulation results are presented in Figure 4 on a log scale. Overall distributed Newton method is about 3 order of magnitude faster than subgradient methods.

VII. Conclusions

This paper develops a distributed Newton-type algorithm for network utility maximization problems. We show that the computation of the dual Newton step can be implemented in a decentralized manner using novel matrix splitting technique. We also show that even when the Newton direction and stepsize are computed with some error, the objective function value converges superlinearly a neighborhood of the optimal value. Simulation results also indicates significant improvement over traditional distributed algorithms for network utility maximization problems. Possible future directions include to analyze the relationship between the rate of converge and the underlying topology of the network and to give explicit bounds on iteration count for the entire algorithm.

References

[1] A. Berman and R. J. Plemmons. *Nonnegative Matrices in the Mathematical Sciences*. Academic Press, New York, 1979.

[2] D. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999.

[3] D. Bickson, Y. Tock, A. Zyrnnis, S. Boyd, and D. Dolev. Distributed large scale network utility maximization. *Proceedings of the 2009 IEEE international conference on Symposium on Information Theory*, 2, 2009.

[4] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[5] M. Chiang, S. H. Low, A. R. Calderbank, and J.C. Doyle. Layering as optimization decomposition: A mathematical theory of network architectures. *Proceedings of the IEEE*, 95(1):255–312, 2007.

[6] R. Cottle, J. Pang, and R. Stone. *The Linear Complementarity Problem*. Academic Press, 1992.

[7] A. Jadbabaie, J. Lin, and S. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on Automatic Control*, 48(6):988–1001, 2003.

[8] A. Jadbabaie, A. Ozdaglar, and M. Zargham. A Distributed Newton method for network optimization. *Proc. of CDC*, 2009.

[9] F. Jarre. Interior-point methods for convex programming. *Applied Mathematics and Optimization*, 26:287–311, 1992.

[10] F. Kelly. Charging and rate control for elastic traffic. *European Transactions on Telecommunications*, 8:33–37, 1997.

[11] F. P. Kelly, A. K. Maulloo, and D. K. Tan. Rate control for communication networks: shadow prices, proportional fairness, and stability. *Journal of the Operational Research Society*, 49:237–252, 1998.

[12] S. H. Low and D. E. Lapsley. Optimization flow control, I: Basic algorithm and convergence. *IEEE/ACM Transaction on Networking*, 7(6):861–874, 1999.

[13] A. Nedic and A. Ozdaglar. *Convex Optimization in Signal Processing and Communications*, chapter 10 Cooperative distributed multi-agent optimization. Eds., Eldar, Y. and Palomar, D., Cambridge University Press, 2008.

[14] Y. Nesterov and A. Nemirovskii. *Interior-Point Polynomial Algorithms in Convex Programming*. SIAM, 2001.

[15] A. Olshevsky and J. Tsitsiklis. Convergence speed in distributed consensus and averaging. *SIAM Journal on Control and Optimization*, 48(1):33–35, 2009.

[16] J. N. Tsitsiklis. *Problems in Decentralized Decision Making and Computation*. PhD thesis, Dept. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1984.

[17] J. N. Tsitsiklis, D. P. Bertsekas, and M. Athans. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE Transactions on Automatic Control*, 31(9):803–812, 1986.

[18] R. Varga. *Gershgorin and His Circles*. Springer, 2004.

[19] E. Wei, A. Ozdaglar, and A. Jadbabaie. A distributed newton method for network utility maximization. *LIDS Report 2832*, 2010.