A distributed Newton method for Network Utility Maximization

49th IEEE Conference on Decision and Control
December 15-17, 2010
Hilton Atlanta Hotel, Atlanta, GA, USA

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 scalar 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 [1], [5], [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 [11], [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 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 way. First challenge is the computation of the Newton direction. This computation involves matrix inversion, which is costly and requires global information. We solve this problem by utilizing an iterative scheme based on novel matrix splitting techniques, which enables us to compute both primal and dual updates for the Newton step using decentralized algorithms that involves only limited scalar information exchange between sources and links. The second challenge is the global information required to compute the stepsize. We resolve this 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 [9]. 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, [9] 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 with the usage of barrier functions, Lipschitz-based results cannot be applied, instead we use properties of self-concordant functions to analyze the convergence behavior of our algorithm. Due to space constraints, some of the proofs and results are omitted in this paper, interested readers should refer to [18] for the complete version.

*Asu Ozdaglar's research was supported by National Science Foundation under Career grant DMI-0545910 and the DARPA ITMANET program. Ali Jadbabaie's research was supported by ONR MURI N000140810747 and by AFOSR Complex Networks Program.

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

‡Department of Electrical and Systems Engineering and GRASP Laboratory, 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 [8], [9], [15], [16], [17]

The rest of the paper is organized as follows: Section II defines the problem formulation and equivalent transformation. 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 decentralized inexact primal update. Section V analyzes the rate of convergence of our algorithm. Section VI presents simulation results to demonstrate convergence speed improvement of our algorithm to the existing methods. Section VII contains our concluding remarks.

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, and $[A]_i$ to denote the i^{th} column of the matrix A. 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 $f : X \to \mathbb{R}$, where X is a subset of $\mathbb{R}^n$, the gradient vector and the Hessian matrix of f at x in X are denoted by $\nabla f(x)$ and $\nabla^2 f(x)$ respectively.

A real-valued convex function $g : X \to \mathbb{R}$, where X is a subset of $\mathbb{R}$, is said to be *self-concordant* if $|g'''(x)| \leq 2g''(x)^{\frac{3}{2}}$ for all x in its domain[2]. For real-valued functions in $\mathbb{R}^n$, a convex function $g : X \to \mathbb{R}$, where X is a subset of $\mathbb{R}^n$, 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] Chapter 9 for more detail).

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*[3] of dimension $L \times S$, i.e.,

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

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

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

We adopt the following standard assumption.

Assumption 1: The utility functions $U_i : \mathbb{R}_+ \to \mathbb{R}$ are continuous, strictly concave, monotonically nondecreasing on $\mathbb{R}_+$ and twice continuously differentiable on the set of positive real numbers, i.e., $(0, \infty)$. The functions $-U_i : \mathbb{R}_+ \to \mathbb{R}$ are self-concordant on the set of positive real numbers.

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 $f : \mathbb{R}_+^S \to \mathbb{R}$ the objective function, i.e.,$f(x) = -\sum_{i=1}^{S} U_i(x_i) - \mu \sum_{i=1}^{S+L} \log(x_i)$, and by f^* the optimal objective value for the equality constrained problem (3). Notice that by Assumption 1 and the properties of logarithmic functions, the objective function $f(x)$ is separable, strictly convex, twice continuously differentiable, and has a positive definite diagonal Hessian matrix on the positive orthant. The function $f(x)$ is also self-concordant for $\mu \geq 1$, since it is a sum of self-concordant functions.

We denote the function $h : \mathbb{R}_+^S \to \mathbb{R}$ to be $h(x) = -\sum_{i=1}^{S} U_i(x_i)$, and let h^* be the negative of the optimal objective function value for problem (2). We write the optimal solution of problem (3) for a fixed barrier function coefficient μ as $x(\mu)$. One can show that as the barrier function coefficient μ approaches 0, the optimal solution of problem (3) approaches that of problem (2) [2], [7], and hence by continuity from Assumption 1, $h(x(\mu))$ approaches h^*. Based on results from [18], we can achieve an error bound in the form of $\frac{h(x(\mu))-h^*}{h^*} \leq a$ for any positive scalar a while keeping $\mu \geq 1$, which preserve the self-concordant property of the function f, which is in turn used to prove convergence of our distributed algorithm. Therefore, in the rest of this paper, unless clearly stated otherwise, our goal is to investigate *iterative distributed methods* for solving problem (3) for a given $\mu \geq 1$, when the function f is self-concordant.

[2]One alternative definition is a real-valued convex function $g : X \to \mathbb{R}$, where X is a subset of $\mathbb{R}$, is said to be *self-concordant*, if there exists a constant $a > 0$, such that $|g'''(x)| \leq 2a^{-\frac{1}{2}} g''(x)^{\frac{3}{2}}$ for all x in its domain [14], [10]. The definition we adopt is a special case of this one, in particular $a = 1$. We choose to not use this general definition, because it introduces unnecessary complications for the convergence analysis.

[3]This is also referred to as the *link-node incidence matrix* in the literature.

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 $x_i^0 = \frac{\min_k\{c_k\}}{S+1}$ for $i = 1, 2 \dots S$, and $x_{i+S}^0 = c_i - \sum_{j=1}^{S} R_{ij} \frac{\min_k\{c_k\}}{S+1}$ for $i = 1, 2 \dots L$, 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 $x^{k+1} = x^k + s^k \Delta x^k$, where s^k is a positive stepsize, Δx^k is the Newton direction given as the solution to the following system of linear equations[4]:

$$\begin{pmatrix} \nabla^2 f(x^k) & A' \\ A & 0 \end{pmatrix} \begin{pmatrix} \Delta x^k \\ w^k \end{pmatrix} = - \begin{pmatrix} \nabla f(x^k) \\ 0 \end{pmatrix}, \quad (4)$$

where the vector $[w_l^k]_{l \in \mathcal{L}}$ are the dual variables for the link capacity constraints. The dual variables associated with each link can be viewed as a price for using the link, we will use the terms "dual variable" and "price" interchangeably in the rest of the paper. We denote $H_k = \nabla^2 f(x^k)$ for notational convenience. Solving for x^k and w^k in the preceding system yields

$$\Delta x^k = -H_k^{-1}(\nabla f(x^k) + A' w^k), \text{ and} \quad (5)$$

$$(A H_k^{-1} A') w^k = -A H_k^{-1} \nabla f(x^k). \quad (6)$$

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_l^k for l in $L(i)$, the Newton direction Δx_i^k can be computed using local information by each source i. 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 decentralized scheme to compute the vector w^k.

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, a comprehensive review of which can be found in [6] and [18].

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

A. Distributed Computation of the Dual Variables

We use the matrix splitting scheme to compute the dual variables w^k in Eq. (6) in a distributed manner. Let D_k be a diagonal matrix, with diagonal entries $(D_k)_{ll} = (A H_k^{-1} A')_{ll}$, and matrix B_k be given by $B_k = A H_k^{-1} A' - D_k$. Let matrix $\bar{B}_k$ be a diagonal matrix, with diagonal entries $(\bar{B}_k)_{ii} = \sum_{j=1}^{L} (B_k)_{ij}$. By using the splitting scheme

$$(A H_k^{-1} A') = (D_k + \bar{B}_k) + (B_k - \bar{B}_k). \quad (7)$$

we obtain the following result.

Theorem 4.1: For a given $k > 0$, let D_k, B_k, $\bar{B}_k$ be the matrices defined as above. Let $w(0)$ be an arbitrary initial vector and consider the sequence $\{w(t)\}$ generated by the iteration

$$\begin{aligned} w(t+1) =& (D_k + \bar{B}_k)^{-1} (\bar{B}_k - B_k) w(t) \quad (8) \\ &+ (D_k + \bar{B}_k)^{-1} (-A H_k^{-1} \nabla f(x^k)), \end{aligned}$$

for all $t \geq 0$. Then the sequence $\{w(t)\}$ converges as $t \to \infty$, and its limit is the solution to Eq. (6).

There can be many ways to split the matrix $A H_k^{-1} A'$, the particular one in Eq. (7) is chosen here due to three desirable features. First it 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. The third feature is related to the concept of a dual (routing) graph and its graph Laplacian matrix (see [18] for more detail).

We next describe a distributed information exchange and computation procedure for the dual variables. In order to express the procedure concisely, we define the *price of the route* for source i, $\pi_i(t)$, as $\pi_i(t) = \sum_{l \in L(i)} w_l(t)$; and the *weighted price of the route* for source i, as $\Pi_i(t) = (H_k^{-1})_{ii} \sum_{l \in L(i)} w_l(t)$. Then at each primal iteration k, the dual variable can be computed as follows:

1. Initialization
   - 1.a Each source i sends its $(H_k)_{ii}$ and $\nabla_i f(x^k)$ (the i^{th} component of the gradient vector $\nabla f(x^k)$) to the links it is using, i.e., $l \in L(i)$. Each link l computes $(D_k)_{ll}$, $\sum_{i \in S(l)} (H_k)_{ii}^{-1}$ and $\sum_{i \in S(l)} (H_k^{-1})_{ii} \nabla_i f(x^k)$.
   - 1.b Each link sends a pilot price of 1, i.e., $w_l(0) = 1$, to the sources that use it, i.e., $i \in S(l)$. The sources aggregates the prices along its route to obtain $\pi_i(0) = \sum_{l \in L(i)} w_l(0)$ and computes $\Pi_i(0) = (H_k^{-1})_{ii} \sum_{l \in L(i)} w_l(0)$.
   - 1.c The weighted price of route of source i, $\Pi_i(0)$ is sent to all the links source i uses, i.e., $l \in L(i)$. Each link l aggregates the total prices and computes $(\bar{B}_k)_{ll} = \sum_{i \in S(l)} \Pi_i(0) - \sum_{i \in S(l)} (H_k)_{ii}^{-1}$.
   - 1.d Initialize an arbitrary value of dual variables at each link l as $w_l(1)$.
2. Iteration.
   - 2.a Each link sends $w_l(t)$ to the sources that use it, i.e., $i \in S(l)$. Each source i aggregates the prices

along its route to obtain $\pi_i(t) = \sum_{l \in L(i)} w_l(t)$ and computes $\Pi_i(t) = (H_k^{-1})_{ii} \sum_{l \in L(i)} w_l(t)$.

2.b The weighted price of route of source i, $\Pi_i(t)$ is sent to all the links source i uses, i.e., $l \in L(i)$, then each link l aggregates the total prices from all the sources and updates $w_l(t+1) = \frac{1}{(D_k)_{ll} + (\bar{B}_k)_{ll}}((\bar{B}_k)_{ll} w_l(t) - \sum_{i \in S(l)} \Pi_i(t) + \sum_{i \in S(l)} (H_k)_{ii}^{-1} w_l(t) - \sum_{i \in S(l)} (H_k^{-1})_{ii} \nabla_i f(x^k) - (H_k^{-1})_{(S+l)(S+l)} \nabla_{S+l} f(x^k))$.

It can be shown that the above procedure can be implemented in a distributed way and it coincides with iteration (8), and hence generates the desired solution to Eq. (6) for each primal iteration k (see [18] for more detail).

B. Distributed Computation of the Newton Primal Direction

Once the dual variables are obtained, the primal Newton direction can be solved according to (5), with $(\Delta x^k)_i = -(H_k)_{ii}^{-1}(\nabla_i f(x^k) + (A' w^k)_i) = -(H_k)_{ii}^{-1}(\nabla_i f(x^k) + \pi_i)$, where π_i is the last price of the route computed from the dual variable computation procedure, and hence the primal direction can be calculated in a distributed way also. However, because our distributed dual variable computation involves an iterative scheme, the exact value for w^k is not available. Hence, 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. (5) 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 computation can be easily performed due to the nature of slack variables.

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

$$x^{k+1} = x^k + s^k \Delta\tilde{x}^k, \tag{9}$$

where s^k is a positive stepsize, and $\Delta\tilde{x}^k$ is the inexact Newton direction at the k^{th} iteration. As we will show in Theorem 4.3, we can choose our stepsize to ensure the primal variables $x^k > 0$ for all k, and hence all the logarithmic barrier functions in the objective function of problem (3) are well defined.

We refer to the exact solution to the system of equations (4) 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

$$\lambda(x^k) = \sqrt{(\Delta x^k)' \nabla^2 f(x^k) \Delta x^k}. \tag{10}$$

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

$$\tilde{\lambda}(x^k) = \sqrt{(\Delta\tilde{x}^k)' \nabla^2 f(x^k) \Delta\tilde{x}^k}. \tag{11}$$

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 is based on the inexact Newton decrement $\tilde{\lambda}(x^k)$ to ensure the quadratic rate of convergence of our algorithm, as we will show in Section V. Therefore, we first need to compute $\tilde{\lambda}(x^k)$ in a distributed way, which is the norm of weighted inexact Newton direction $\Delta\tilde{x}^k$ and hence it 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], [8], [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

$$s^k = \begin{cases} \frac{c}{\theta^k + 1} & \text{if} \quad \theta^k \geq \frac{1}{4}, \\ 1 & \text{otherwise}, \end{cases} \tag{12}$$

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.

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. We quantify the bounds on the 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.

The constant ϵ is here to prevent the requirement of vanishing error when x^k is close to the optimal solution, 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, as we will show in Lemma 4.2 and Theorem 4.3. For the rest of the paper, we assume the conditions in Assumptions 1-3 hold.

We now show that the stepsize choice in (12) 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 4.2: Let $\tilde{\lambda}(x^k)$ be the inexact Newton decrement defined in (11), 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 (12). For $\theta^k \geq \frac{1}{4}$, the following relation holds $(2c-1)/(\tilde{\lambda}(x^k) + 1) \leq \frac{c}{\theta^k + 1} \leq 1/(\tilde{\lambda}(x^k) + 1)$.

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., $x^k > 0$, as in the following theorem.

Theorem 4.3: Let x^0 be a positive feasible primal variable, x^k be the sequence of primal variables updated using iteration (9), 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-B, and s^k is defined as in (12). Then for all k, the primal variable satisfies $x^k > 0$.

Therefore our algorithm guarantees the objective function of problem (3) is well defined throughout.

V. Convergence Analysis

We next present our analysis for convergence results for both primal and dual iterations. We first establish convergence for the dual iterations.

A. Convergence in Dual Iterations

In this section, we present an explicit rate of convergence bound for the iterations in our dual variable computation procedure described in Section IV-A.

Lemma 5.1: Let M be an $n \times n$ matrix, and assume that its spectral radius, denoted by $\rho(M)$, satisfies $\rho(M) < 1$. Let λ_i denote the set of eigenvalues of M, with $1 > |\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_n|$. Assume the matrix has n linearly independent eigenvectors. Then for the sequence $w(t)$ generated by the following iteration $w(t+1) = Mw(t)$, we have $||w(t) - w^*|| \leq |\lambda_1|^t \alpha$, for some positive scalar α, where w^* is the limit of the preceding iteration as $t \to \infty$.

Our dual variable computation algorithm implements iteration (8), using the above lemma it can be shown that $||w(t) - w^*|| = ||u(t) - u^*|| \leq |\lambda_1|^t \alpha$, where λ_1 is the eigenvalue of the matrix $(D_k + \bar{B}_k)^{-1}(\bar{B}_k - B_k)$ with largest magnitude, and α is a constant depending on the initial condition. Note that the matrix $(D_k + \bar{B}_k)^{-1}(\bar{B}_k - B_k)$ is the weighted Laplacian matrix of the dual graph, hence the rate of convergence depends on the spectral properties of the dual graph (see [18] for more detail).

B. Convergence in Primal Iterations

We next present our convergence analysis for the primal solution generated by the inexact Newton algorithm defined in Eq. (9). For the k^{th} iteration, we define the function $\tilde{f}_k : \mathbb{R} \to \mathbb{R}$ as

$$\tilde{f}_k(t) = f(x^k + t\Delta\tilde{x}^k), \tag{13}$$

which is self-concordant, because the objective function f is self-concordant. Note that $\tilde{f}_k(s^k) - \tilde{f}_k(0)$ measures the decrease in objective function value at the k^{th} iteration. For the rest of the analysis, we employ theories of self-concordant functions and properties of the Newton decrement, a comprehensive review of which can be found in [18]. Our analysis comprises of two parts, the first part is the damped convergent phase, in which improvement in the objective function value at each step is bounded below by a constant. The second part is the quadratically convergent phase, in which the suboptimality in the objective function value, i.e., $f(x^k) - f^*$, diminishes quadratically to an error neighborhood of 0.

1) 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. (12)]. We prove the improvement in the objective function value is lower bounded by a constant.

Theorem 5.2: Let $\tilde{f}_k$ be the function defined in Eq. (13), and $\tilde{\lambda}(x^k)$ be the inexact Newton decrement defined in Eq. (11). 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. (12)]. 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, $\tilde{f}_k(t) - \tilde{f}_k(0) \leq -\frac{\alpha(1+p)\left(\frac{6c-5}{4c}\right)^2}{\left(1+\frac{6c-5}{4c}\right)}$.

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 and the error between the two is bounded by Lemma 4.2. By using the convexity of the function f, Lemma 4.2 and the preceding theorem, we obtain $f(x^{k+1}) - f(x^k) \leq -\frac{(2c-1)\alpha(1+p)\left(\frac{6c-5}{4c}\right)^2}{\left(1+\frac{6c-5}{4c}\right)}$. Hence in the damped convergent phase we can guarantee a lower bound on the object function value improvement at each iteration.

2) 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 5.3: 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. (10) and (11) respectively. Then the following relation holds:

$$(1-p)\tilde{\lambda}(x^k) - \sqrt{\epsilon} \leq \lambda(x^k) \leq (1+p)\tilde{\lambda}(x^k) + \sqrt{\epsilon}, \tag{14}$$

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

We impose the following bound on the errors in our algorithm when $\theta^k < \frac{1}{4}$.

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,

$$(1+p)(\theta^k + \tau) + \sqrt{\epsilon} \leq \phi \tag{15}$$

$$p + \sqrt{\epsilon} \leq 1 - (4\phi^2)^{\frac{1}{4}} - \phi, \tag{16}$$

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 (16) can be satisfied by some positive scalars p and ϵ. By Assumption 4, the Newton decrement is sufficiently small, so that we can bound suboptimality in our algorithm, i.e., $f(x^k) - f^*$, using the exact Newton decrement (see [4], [18] for more detail).

Theorem 5.4: Let λ and $\tilde{\lambda}$ be the exact and inexact Newton decrement defined in Eqs. (10) and (11) respectively. Let $f(x^k)$ be the objective function value at k^{th} iteration for the algorithm defined in Section IV and f^* be the optimal

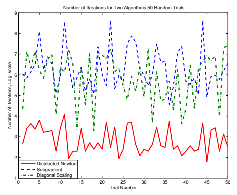

Fig. 1. Log scaled iteration count for the 3 methods implemented over 50 randomly generated networks .

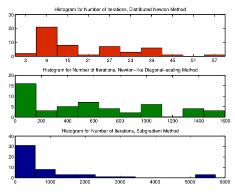

Fig. 2. Histogram of iteration counts for the 3 methods implemented over 50 randomly generated networks

objective function value for problem (3). Let Assumption 4 hold. Let $\xi = \frac{\phi p+\sqrt{\epsilon}}{1-p-\phi-\sqrt{\epsilon}} + \frac{2\phi\sqrt{\epsilon}+\epsilon}{(1-p-\phi-\sqrt{\epsilon})^2}$, and $v = \frac{1}{(1-p-\phi-\sqrt{\epsilon})^2}$. 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 $m > 0$,

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

and

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

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 f^*, with the neighborhood of size $\xi + \frac{\delta}{2v}$, where ξ, v, δ are defined as above. 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. For a comprehensive comparison, we count both the primal and dual iterations for our distributed Newton method. In the simulation results, we compare our distributed Newton method performance against both distributed subgradient method [12] and the Newton-type diagonal scaling dual method developed in [1].

To test the performances of the methods over general networks, we generated 50 random networks. The number of links L in the network is a random variable with mean 40, and number of sources S is a random variable with the mean 10. Each routing matrix consists of $L \times R$ Bernoulli random variables. All three methods are implemented over the 50 networks. We record the number of iterations upon termination for all 3 methods, and results are shown in Figure 1 and Figure 2. Figure 1 shows the number until termination on a log scale. We can see that overall distributed Newton method is about much faster than subgradient methods, and the diagonal-scaling method's performance lies between the other two, with a tendency to be closer to the first order subgradient method. Figure 2 shows the histogram for the same set of data. This figure shows on average our method is about 100 times faster than subgradient method for these relatively small networks. Diagonal scaling method has performance on the same order of magnitude as the subgradient method, but slightly faster.

VII. Conclusions

This paper develops a distributed Newton-type second order algorithm for network utility maximization problems that can achieve superlinear rate of convergence to some explicitly quantified error neighborhood. We show that the computation of the dual Newton step can be implemented in a decentralized manner using matrix splitting technique with very limited scalar information exchange. We show that even when the Newton direction and stepsize are computed with some error, the method achieves superlinear convergence rate to an error neighborhood. Simulation results also indicates significant improvement over traditional distributed algorithms. 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] S. Athuraliya and S. Low. Optimization flow control with Newton-like algorithm. *Journal of Telecommunication Systems*, 15:345–358, 2000.
[2] D. P. 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. V. Fiacco and G. P. McCormick. *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*. SIAM, 1990.
[8] 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.
[9] A. Jadbabaie, A. Ozdaglar, and M. Zargham. A Distributed Newton method for network optimization. *Proc. of CDC*, 2009.
[10] F. Jarre. Interior-point methods for convex programming. *Applied Mathematics and Optimization*, 26:287–311, 1992.
[11] F. Kelly. Charging and rate control for elastic traffic. *European Transactions on Telecommunications*, 8:33–37, 1997.
[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 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] E. Wei, A. Ozdaglar, and A. Jadbabaie. A Distributed Newton Method for Network Utility Maximization . *LIDS Report 2832*, 2010.