On Dual Convergence of the Distributed Newton Method for Network Utility Maximization*

Ermin Wei†, Michael Zargham‡, Asuman Ozdaglar†, and Ali Jadbabaie‡

***Abstract*— The existing distributed algorithms for Network Utility Maximization (NUM) problems mostly rely on dual decomposition and first-order (gradient or subgradient) methods, which suffer from slow rate of convergence. Recent works [17] and [18] proposed an alternative distributed Newton-type second-order algorithm for solving NUM problems with self-concordant utility functions. This algorithm is implemented in the primal space and involves for each primal iteration computing the dual variables using a finitely terminated iterative scheme obtained through novel matrix splitting techniques. These works presented a convergence rate analysis for the primal iterations and showed that if the error level in the Newton direction (resulting from finite termination of dual iterations) is below a certain threshold, then the algorithm achieves local quadratic convergence rate to an error neighborhood of the optimal solution. This paper builds on these works and presents a convergence rate analysis for the dual iterations that enables us to explicitly compute at each primal iteration the number of dual steps that can satisfy the error level. This yields for the first time a fully distributed second order method for NUM problems with local quadratic convergence guarantee. Simulation results demonstrate significant convergence rate improvement of our algorithm, even when only one dual update is implemented per primal iteration, relative to the existing first-order methods based on dual decomposition.**

I. Introduction

There has been much recent interest in developing distributed algorithms for solving convex optimization problems over networks. This is mainly motivated by resource allocation problems that arise in large-scale communication networks. This paper focuses on building a fully distributed fast converging algorithm for the rate allocation problem in wireline networks, also referred to as the *Network Utility Maximization (NUM)* problem in the literature (see [1], [3], [9], [10], [20], [15]). NUM problems are characterized by a fixed set of sources with predetermined routes over a network topology. Each source in the network has a local utility, which is a function of the rate at which it transmits information over the network. The objective is to determine the source rates such that collectively the sum of the utilities is maximized, without violating link capacity constraints. The standard approach for solving NUM problems in a distributed way relies on using dual decomposition and first-order (subgradient) methods, which through a dual price exchange mechanism enables each source to determine its transmission rate using only locally available information ([8], [10], [12]). However, the drawback of these methods is their slow rate of convergence.

In this paper, we study distributed second-order methods for solving NUM problems. Our development builds on [17] and [18], which proposed a Newton-type primal algorithm for solving NUM problems in a distributed manner. Computation of dual variables at each primal iteration is achieved using a distributed method constructed using novel matrix splitting techniques. This involves an iterative scheme, which needs to be truncated for practical implementation, leading to an error in the computation of the primal Newton direction. The works [17] and [18] assume an error level on the Newton direction and establish quadratic local convergence to a neighborhood of the optimal solution (under some assumptions on the error level). This paper provides a novel convergence rate analysis for the dual iterative scheme that yields an explicit bound on the number of dual iterations to be executed in order to satisfy the given error level. Our bounds are *primal-dependent*, i.e., they are a function of the current primal iterate. We also relate our bounds and the information exchange involved in dual iterations to the properties of the *dual-graph* defined through the given network and sources.

Our results enable us to compute the primal Newton direction and the dual variables using decentralized algorithms that involve limited scalar information exchange between sources and links, comparable to that of first-order methods. This yields for the first time a fully distributed second order method for the NUM problems with local quadratic convergence guarantee.

Other than the papers cited above, our paper is related to [6] and [21]. In [6], the authors developed a distributed Newton method to solve an equality constrained network optimization problem and showed that this method achieves local superlinear convergence rate under Lipschitz assumptions. [21] studied this algorithm under a uniform bound on the number of dual iterations. The NUM formulation involves an inequality-constrained problem, which is turned into an equality-constrained problem using barrier functions. Hence a Lipschitz-based analysis is not applicable. Instead we use a primal convergence analysis based on properties of self-concordant functions and develop a dual convergence analysis

*This research is supported by NSF Career grant DMI-0545910, AFOSR MURI R6756-G2, ONR MURI N000140810747, ARO MURI SWARMS, and 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.

based on the spectral properties of the matrices involved in the splitting scheme.

The rest of the paper is organized as follows: Section II defines the problem formulation and related transformations. Section III presents the exact constrained Newton method for this problem. Section IV presents a distributed method for computing the dual variables and the inexact primal Newton direction. Section V presents simulation results to demonstrate convergence speed improvement of our algorithm relative to the existing methods. Section VI contains our concluding remarks. Due to space constraints, some of the technical details in this paper are omitted. We refer the reader to [19] for the missing details.

Basic Notation and Notions:

We write $I(n)$ to denote the identity matrix of dimension $n \times n$. A real-valued convex function $g : X \to \mathbb{R}$, where X is a subset of $\mathbb{R}$, is *self-concordant* if $|g'''(x)| \leq 2g''(x)^{\frac{3}{2}}$ for all x in its domain.[1] 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. Operations that preserve self-concordance property include summing, scaling by a factor $\alpha \geq 1$, and composition with affine transformation (see [2] Chapter 9 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 nonzero capacity given by $c = [c_l]_{l\in\mathcal{L}}$ and a set $\mathcal{S} = \{1, ..., S\}$ of sources, each of which transmits information along a predetermined route.[2] 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. Let the nonnegative source rate vector be denoted by $s = [s_i]_{i\in\mathcal{S}}$. Let matrix R be the *routing matrix* of dimension $L \times S$, given by

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

For each i, we use $U_i : \mathbb{R}_+ \to \mathbb{R}$ to denote the utility function of source i. The *Network Utility Maximization (NUM)* problem involves choosing the source rates to maximize a global system function given by the sum of all utility functions and 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 assumptions on the utility function.

[1]Self-concordant functions are defined through the following more general definition: a real-valued convex function $g : X \to \mathbb{R}$, where X is a subset of $\mathbb{R}$, is *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 [13], [7]. Here we focus on the case $a = 1$ for notational simplification in the analysis.

[2]We assume that each source flow traverses at least one link and each link is used by at least one source.

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. The functions $-U_i : \mathbb{R}_+ \to \mathbb{R}$ are self-concordant on the set of positive real numbers.

The self-concordance assumption is satisfied by most utility functions considered in the literature, linear, quadratic and α-fair utility functions with $\alpha = 1$ for instance [11], and is adopted here to allow a self-concordant analysis in establishing local quadratic convergence.

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 $l = 1, 2 \dots L$, and using logarithmic barrier functions for the nonnegativity constraints. The new decision vector is $x = ([s_i]'_{i\in\mathcal{S}}, [y_l]'_{l\in\mathcal{L}})'$ and problem (2) 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 μ is a positive coefficient for the barrier functions and $A = [R \quad I(L)]$. We denote by $f : \mathbb{R}_+^{S+L} \to \mathbb{R}$ the objective function, i.e., $f(x) = -\sum_{i=1}^{S} U_i(x_i) - \mu \sum_{i=1}^{S+L} \log(x_i)$. Throughout the paper, we assume that $\mu \geq 1$, which guarantees that $f(x)$ is self-concordant. This is without loss of generality since it was shown in [18] that, under self-concordance assumptions, the problem with a general $\mu \geq 0$ can be addressed by solving two instances of problem (3) with different coefficients $\mu \geq 1$.

III. Exact Newton Method

We solve problem (3) using a (feasible start) equality-constrained Newton method (see [2] Chapter 10), which serves as a starting point in the development of a distributed algorithm. In our iterative method, we use x^k to denote the primal vector at the k^{th} iteration.

A. Feasible Initialization

To initialize the algorithm, we start with some feasible and strictly positive vector $x^0 > 0$. For example, one possible initialization is given by $x_i^0 = \frac{\min_k\{c_k\}}{S+1}$ for $i = 1, 2 \dots S$, and $x_{l+S}^0 = c_l - |S(l)|\frac{\min_k\{c_k\}}{S+1}$ for $l = 1, 2 \dots L$.

B. Iterative Update Rule

We denote $H_k = \nabla^2 f(x^k)$ for notational convenience. Given an initial feasible vector x^0, the algorithm generates the iterates by $x^{k+1} = x^k + d^k \Delta x^k$, where d^k is a positive stepsize, Δx^k is the (primal) Newton direction given as

$$\Delta x^k = -H_k^{-1}\left(\nabla f(x^k) + A'w^k\right), \text{ and} \tag{4}$$

$$(AH_k^{-1}A')w^k = -AH_k^{-1}\nabla f(x^k), \tag{5}$$

where $w^k = [w_l^k]_{l\in\mathcal{L}}$ is the dual vector and the w_l^k are the dual variables for the link capacity constraints at primal iteration k.

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, i.e., $[H_k^{-1}]_{ii} = \left(\frac{\partial^2 f}{(\partial x_i^k)^2}\right)^{-1} > 0$ for all k. Direct computation of w^k cannot be implemented in a decentralized manner, due to the fact that the evaluation of the matrix inverse $(AH_k^{-1}A')^{-1}$ requires global information. In the next section, we present an algorithm that can compute the dual vector w^k in a distributed manner over the network.

IV. Distributed Newton Method

This section describes the distributed Newton method. Our development will be based on [17] and [18], which introduced a distributed iterative scheme to compute the dual vector and a distributed method to determine an inexact primal Newton direction (given a dual vector) that maintains primal feasibility. Section IV-A summarizes this method and presents bounds on the error level in the computation of the inexact Newton direction. Section IV-B provides a convergence rate analysis for the computation of the dual vector and presents estimates on the number of iterations that can guarantee a given error level in the Newton direction. Section IV-C shows that these schemes can be computed using *distributed* and *scalar* information exchange comparable to that of first-order methods.

A. Distributed Computation of Iterates

The computation of the dual vector w^k at a given primal solution x^k requires solving a linear system of equations [cf. Eq. (5)]. The dual variables can be computed using a distributed iterative scheme relying on novel ideas from matrix splitting (see [4] for a comprehensive review). We let D_k be a diagonal matrix with diagonal entries $(D_k)_{ll} = (AH_k^{-1}A')_{ll}$, B_k be a symmetric matrix given by $B_k = (AH_k^{-1}A') - D_k$, and $\bar{B}_k$ be a diagonal matrix with diagonal entries $(\bar{B}_k)_{ii} = \sum_{j=1}^{L}(B_k)_{ij}$. Furthermore, we define a diagonal matrix $\bar{D}_k = D_k + \bar{B}_k$, which is used as a scaling matrix in the dual iterations. The dual iterations are given by

$$w^k(t+1) = \bar{D}_k^{-1}(\bar{B}_k - B_k)w^k(t) - \bar{D}_k^{-1}AH_k^{-1}\nabla f(x^k), \tag{6}$$

with initialization $w^k(1) = -\bar{D}_k^{-1}AH_k^{-1}\nabla f(x^k)$. It was shown in [17] that the matrix $\bar{D}^{-1}(\bar{B}_k - B_k)$ has subunit spectral radius and the sequence $\{w^k(t)\}_t$ converges to w^k as $t \to \infty$.

The distributed Newton method uses the same initialization as presented in Section III-A, however, it computes the primal Newton direction in two stages to maintain feasibility. In the first stage, the first S components of $\Delta\tilde{x}^k$ are computed via Eq. (4) using the dual vector obtained from iteration (6). 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. The feasibility correction is given by

$$\left(\Delta\tilde{x}^k\right)_{\{S+1\ldots S+L\}} = -R\left(\Delta\tilde{x}^k\right)_{\{1\ldots S\}}. \tag{7}$$

Starting from an initial feasible vector x^0, the distributed Newton algorithm generates the primal vectors x^k as follows:

$$x^{k+1} = x^k + d^k\Delta\tilde{x}^k, \tag{8}$$

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 of the system of equations (4) as the *exact Newton direction*, denoted by Δx^k. The inexact Newton direction $\Delta\tilde{x}^k$ computed by 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

$$\lambda(x^k) = \sqrt{(\Delta x^k)'\nabla^2 f(x^k)\Delta x^k}. \tag{9}$$

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

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

Note that both $\lambda(x^k)$ and $\tilde{\lambda}(x^k)$ are nonnegative and well defined because the matrix $\nabla^2 f(x^k)$ is positive definite.

The stepsize choice is based on $\tilde{\lambda}(x^k)$, which can be computed in a distributed way using the method in [17]. Once the inexact Newton decrement is computed, the stepsize is given by,

$$d^k = \begin{cases} \frac{b}{\tilde{\lambda}(x^k)+1} & \text{if} \quad \tilde{\lambda}(x^k) \geq \frac{1}{4}, \\ 1 & \text{otherwise,} \end{cases} \tag{11}$$

where b is some positive scalar that satisfies $\frac{5}{6} < b < 1$.

There are two sources of inexactness in this algorithm: finite precision achieved in the computation of the dual vector due to truncation of the iterative scheme (6); two-stage computation of an approximate primal direction to maintain feasibility. The following assumption presents bounds on the resulting error level.

Assumption 2: Let γ^k denote the error in the primal Newton direction, i.e., $\Delta x^k = \Delta\tilde{x}^k + \gamma^k$. For all k, γ^k satisfies $|(\gamma^k)'\nabla^2 f(x^k)\gamma^k| \leq p^2(\Delta\tilde{x}^k)'\nabla^2 f(x^k)\Delta\tilde{x}^k + \epsilon$ for some positive scalars $p < 1$ and ϵ.

Note that both the primal Newton direction can be computed with arbitrary precision (since the errors arise due to finite truncation of the dual iteration (6)). Hence, the error γ^k can be sufficiently small, thus satisfying the preceding assumptions for any value of the scalars p, and ϵ. Results from [18] show that if the error level in Assumption 2 is satisfied, then the inexact Newton algorithm is well-defined and the generated objective function value converges with quadratic rate to an error neighborhood of the optimal value, where the size of the neighborhood can be explicitly characterized by the parameters of the algorithm. In the following two sections, we develop a distributed computation procedure to guarantee

conditions in Assumption 2 is met.

B. Finite Termination for Dual Iteration

In this section, we derive a bound on the number of dual iterations given by (6) so that the Newton direction generated by our algorithm satisfies Assumption 2 for a given p and ϵ. For notational convenience, we denote $\beta_i^k = \sqrt{\frac{\epsilon}{(L+S)}}\left(|L(i)|(H_k^{-\frac{1}{2}})_{ii}\right)^{-1}$ for source i, $\beta_l^k = \sqrt{\frac{\epsilon}{(L+S)}}(H_k^{-\frac{1}{2}})_{(S+l)(S+l)}\left(\sum_{i\in S(l)}(H_k)_{ii}^{-1}|L(i)|\right)^{-1}$ for link l and $\psi^k = -AH_k^{-1}\nabla f(x^k)$. Our analysis uses the $\bar{D}_k$-induced norm of matrices and interested readers can see [19] for more details.

Theorem 4.1: Let $\{x^k\}$ be the primal sequence generated by the distributed Newton method (8). Let the scalar ϵ be the error level given in Assumption 2. Let $\bar{\rho}^k$, β^k and $\hat{d}^k$ be positive scalars defined by $\bar{\rho}^k = 1 - \frac{\min_{j\in\mathcal{L}\cup\mathcal{S}}(H_k^{-1})_{jj}}{\max_{l\in\mathcal{L}}(\bar{D}_k)_{ll}}$, $\beta^k = \min_{j\in\mathcal{L}\cup\mathcal{S}}\beta_j^k$, and $\hat{d}^k = \min_{l\in\mathcal{L}}\{(\bar{D}_k)_{ll}\}$ respectively. For any k, assume that the dual vector w^k is obtained by implementing the dual iterations (6) N^k steps, where

$$N^k \geq \frac{1}{\log(\bar{\rho}^k)}\log\left(\frac{(1-\bar{\rho}^k)\beta^k\hat{d}^k}{\sqrt{L}\left|\left|\bar{D}_k^{\frac{3}{2}}\psi^k\right|\right|_\infty}\right). \tag{12}$$

The inexact Newton direction obtained using w^k satisfies Assumption 2.

C. Distributed Computation of the Dual Variables

We next rewrite iteration (6), and then analyze the information exchange required among sources and links in the network both to implement the iteration and to calculate the number of iterations as obtained in the previous section. Based on the information exchange structure, we develop a fully distributed procedure to compute the dual variables.

We introduce some notations in order to express the dual iteration concisely for this section. We suppress k such that $w^k(t) = w(t)$ and 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 $\Pi_i(t) = (H_k^{-1})_{ii}\sum_{l\in L(i)} w_l(t)$. The following result from [17] enables us to develop a distributed implementation.

Lemma 4.2: For each primal iteration k, the dual iteration (6) can be written as

$$w_l(t+1) = \frac{1}{(H_k)_{(S+l)(S+l)}^{-1} + \sum_{i\in S(l)}\Pi_i(0)}\Big(\Big(\sum_{i\in S(l)}\Pi_i(0) - \sum_{i\in S(l)}(H_k)_{ii}^{-1}\Big)w_l(t) - \sum_{i\in S(l)}\Pi_i(t) + \sum_{i\in S(l)}(H_k)_{ii}^{-1}w_l(t) - \sum_{i\in S(l)}(H_k^{-1})_{ii}\nabla_i f(x^k) - (H_k^{-1})_{(S+l)(S+l)}\nabla_{S+l}f(x^k)\Big), \tag{13}$$

where $\Pi_i(0)$ is the weighted price of the route for source i when $w(0) = [1, 1\ldots, 1]'$.

We rewrite some terms in Theorem 4.1 using local information to analyze the structure of information exchange. By using the definition of matrices $\bar{D}_k$ and A, we have $(\bar{D}_k)_{ll} = \sum_{i\in S(l)}\Pi_i(0) + (H_k)_{(S+l)(S+l)}^{-1}$, $\psi_l^k = -\sum_{i\in S(l)}(H_k^{-1})_{ii}\nabla_i f(x^k) - (H_k^{-1})_{(S+l)(S+l)}\nabla_{S+l}f(x^k)$ and $(\bar{D}_k^{\frac{3}{2}}\psi^k)_l = (\bar{D}_k)_{ll}^{\frac{3}{2}}\psi_l^k$.

We next analyze the information exchange required among sources and links to implement (13) and compute the number N^k as given in Theorem 4.1. We first observe the local information available to sources and links. Each source i knows $|L(i)|$, $(H_k)_{ii}$ and $\nabla_i f(x^k)$; each link l knows L, $(H_k)_{S+l,S+l}$ and $\nabla_{S+l}f(x^k)$. Each link l also needs to compute the terms: $\sum_{i\in S(l)}(H_k)_{ii}^{-1}$, $\sum_{i\in S(l)}(H_k^{-1})_{ii}\nabla_i f(x^k)$, $\sum_{i\in S(l)}(H_k)_{ii}^{-1}|L(i)|$, $\sum_{i\in S(l)}\Pi_i(0)$, and $\sum_{i\in S(l)}\Pi_i(t)$. The first three terms can be computed by link l if each source sends its local information to the links along its route once in primal iteration k. The fourth term can be computed by link l again once for every k if the route price $\pi_i(0)$ are fed back by the destination to source i, which then sends the weighted price $\Pi_i(0)$ to the links along its route. The last term can be computed with a similar feedback mechanism for N^k times in primal iteration k. In addition to these terms, the sources and links need to calculate the following minima and maxima once per primal iteration: $\min_{j\in\mathcal{L}\cup\mathcal{S}}(H_k^{-1})_{jj}$, $\max_{l\in\mathcal{L}}(\bar{D}_k)_{ll}$, $\min_{j\in\mathcal{L}\cup\mathcal{S}}\beta_j^k$, $\min_{l\in\mathcal{L}}(\bar{D}_k)_{ll}$ and $\max_{l\in\mathcal{L}}|(\bar{D}_k^{\frac{3}{2}}\psi^k)_l|$. The objective function values of all of the above minimizations/maximizations are locally known to each links and/or sources, therefore we can use maximum consensus to obtain the minima/maxima in a distributed way.[3]

The preceding information exchange suggests the following fully distributed procedure to compute dual variables (at each primal iteration k) among the sources and the links:

Initialization.

1.a Each source i computes β_i^k and sends its local information $(H_k)_{ii}$, $|L(i)|$ and $\nabla_i f(x^k)$ to the links along its route, $l \in L(i)$. Each link l computes $(H_k)_{(S+l)(S+l)}^{-1}$, β_l^k, $(H_k^{-1})_{(S+l)(S+l)}\nabla_{S+l}f(x^k)$, $\sum_{i\in S(l)}(H_k)_{ii}^{-1}$ and $\sum_{i\in S(l)}(H_k^{-1})_{ii}\nabla_i f(x^k)$.

1.b Each link l starts with price $w_l(0) = 1$. The link prices $w_l(0)$ are aggregated along route i to compute $\pi(0) = \sum_{l\in L(i)} w_l(0)$ at the destination. This information is sent back to source i.

1.c Each source computes the weighted price $\Pi_i(0) =$

[3] In a maximum consensus algorithm, each node starts with some state and updates its current state with the maximum state value in its neighborhood (including itself). Therefore after one round of algorithm, the neighborhood of the node with maximal value has now the maximum value, after the diameter of the graph rounds of algorithm, the entire graph reaches a consensus on the maximum state value and the algorithm terminates. Interested readers should refer to [16], [5], [14] for further information about general consensus algorithms.

$(H_k^{-1})_{ii} \sum_{l \in L(i)} w_l(0)$ and sends it to the links along its route, $l \in L(i)$.

1.d Sources and links compute the values for $\beta^k = \min_{j \in \mathcal{L} \cup \mathcal{S}} \beta_j^k$, $\min_{j \in \mathcal{L} \cup \mathcal{S}} (H_k^{-1})_{jj}$, $\max_{l \in \mathcal{L}} (\bar{D}_k)_{ll}$, $\min_{l \in \mathcal{L}} (\bar{D}_k)_{ll}$ and $\max_{l \in \mathcal{L}} |(\bar{D}_k^{\frac{3}{2}} \psi^k)_l|$ using maximum consensus algorithm. Then the number of dual iterations N^k is determined by (12).

1.e Each link l then initializes with price $w_l(1) = [\bar{D}_k^{-1}]_{ll} \psi_l^k$.

Dual Iteration.

2.a The link prices $w_l(t)$ are updated using (13) and aggregated along route i to compute $\pi(t)$ at the destination. This information is sent back to source i.

2.b Each source computes the weighted price $\Pi_i(t)$ and sends it to the links along its route, $l \in L(i)$.

The result from the above distributed computation procedure is guaranteed to satisfy Assumption 2. In the remaining part of this section, we introduce the notion of a dual graph and relate the information exchange structure in the dual iterations to the topology of the dual graph.

Definition 1: Consider a network $\mathcal{G} = \{\mathcal{L}, \mathcal{S}\}$, represented by a set $\mathcal{L} = \{1, ..., L\}$ of (directed) links, and a set $\mathcal{S} = \{1, ..., S\}$ of sources. The links form a strongly connected graph, and each source sends information along a predetermined route. The *weighted dual (routing) graph* $\tilde{\mathcal{G}} = \{\tilde{\mathcal{N}}, \tilde{\mathcal{L}}\}$, where $\tilde{\mathcal{N}}$ is the set of nodes, and $\tilde{\mathcal{L}}$ is the set of (directed) links defined by: $\tilde{\mathcal{N}} = \mathcal{L}$; a link is present between node L_i to L_j in $\tilde{\mathcal{G}}$ if and only if there is some common flow between L_i and L_j in $\mathcal{G}$. The weight $\tilde{W}_{ij}$ on the link from node L_i to L_j is given by $\tilde{W}_{ij} = (D_k + \bar{B}_k)_{ii}^{-1} (B_k)_{ij} = (D_k + \bar{B}_k)_{ii}^{-1} (A H_k^{-1} A')_{ij} = [\bar{D}^{-1}]_{ii} \sum_{s \in S(i) \cap S(j)} H_{ss}^{-1}$.

One example of a network and its dual graph are presented in Figure 1. The out-degree of node i in the dual graph can be viewed as a *measure of centrality* of a link in the original network since the neighbors in the dual graph represent links that share flows in the original network. The matrix $M_k = \bar{D}_k^{-1}(\bar{B}_k - B_k)$ is the Laplacian matrix of the dual graph. Since the dual iteration (6) with initialization $w^k(1) = -\bar{D}_k^{-1} \psi_k$ can be written as in $w(t) = \sum_{r=0}^{t-1} M_k^r \bar{D}_k^{-1} \psi^k$, we conclude that N^k steps in the dual iterations require information from links (nodes in the dual graph) which are within $(N^k - 1)$ hops away. Thus the parameter N^k defines an explicit trade off between the exactness of the approximation and the information required to compute it. The spectral radius of the matrix M_k is subunit, which implies that information from nodes closer in the dual graph is more important than information from nodes far away. The relative importance is determined by the spectral radius of the matrix M_k .

V. Simulation Results

Our simulation results demonstrate that the distributed Newton method with locally computed dual iteration steps N^k, significantly outperforms the gradient descent method

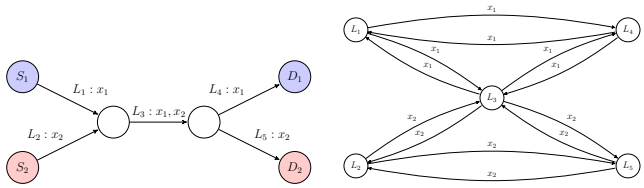

Fig. 1: A sample network and its dual graph. Each source-destination pair is displayed with the same color. We use x_i to denote the flow corresponding to the i^{th} source-destination pair and L_i to denote the i^{th} link.

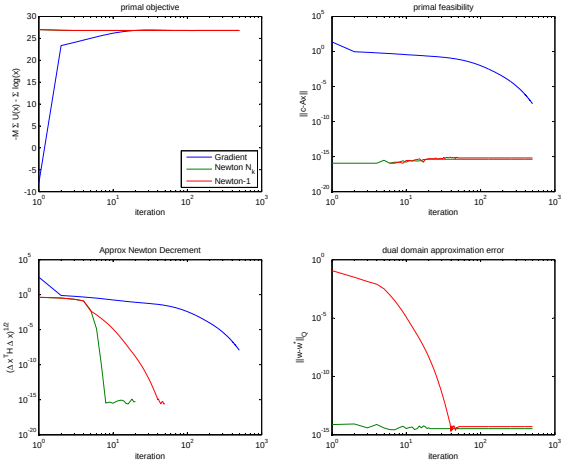

Fig. 2: Sample with 10 links and 7 sources. (Top Left) Convergence to the Primal Objective is shown. (Top Right) Feasibility of our algorithm is maintained. (Lower Left) The Newton Decrement exhibits quadratic convergence. (Lower Right) Dual variables are shown to be optimal for each iteration

in terms of primal iterations required to converge. Due to the conservative bounds on the number of dual iterations, our algorithm effectively recovers the true Newton method in practice. Recognizing that our algorithm recovers the Newton method at the cost of many dual updates, we also implement a truncated version of algorithm in the style of [21], with exactly 1 dual update per primal iteration. Our results also show that the truncated version of our algorithm yields an effective approximate method.

The key characteristics of our algorithm are demonstrated in Figure 2. This representative sample trial was generated randomly with 10 links and 7 sources and assuming that the probability a source uses a particular edge is a Bernoulli random variable. Figure 2 (top right) demonstrates that our algorithm, and its truncated counterpart (denoted Newton-1) remain feasible for the duration of the algorithm as guaranteed by the feasibility correction in (7). Figure 2 (bottom left) shows the Newton decrement of our algorithm exhibits quadratic convergence. It is clear the optimal dual is

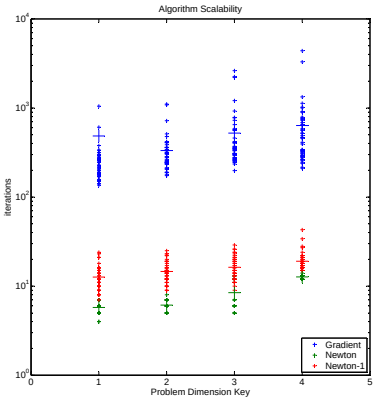

Fig. 3: Number of iterations for 50 randomly generated networks as a function of problem dimension defined as follows: 1= 10 links, 7 sources; 2= 20 links, 15 sources; 3= 40 links, 30 source; 4= 80 links, 50 sources. Wide lines denote means.

computed to machine precision. Therefore our approximate Newton algorithm, under conservative bound on the number of dual iterations (12), effectively achieves the true Newton algorithm. As shown in Figure 2 (bottom right), with the truncated method, the dual variable eventually converges to the true dual variable, after which the truncated algorithm also achieves quadratic convergence.

A more complete characterization of the convergence rate of our algorithm is shown in Figure 3. The data is collected from 200 random graphs, 50 each of 10 links with 7 sources, 20 links with 15 sources, 40 links with 30 sources and 80 links with 50 sources. Link usage is determined by Bernoulli random variable and convergence stopping rule is given by $\lambda(x) < 10^{-5}$. The markers denote the iterations required to converge for individual trials and the large markers denote the mean. The data clearly shows our inexact Newton algorithm converges in order 10 primal iterations which is consistent with the true Newtons method. The truncated version of our algorithm performs remarkably well requiring only marginally more iterations than the basic version. Both Newton based algorithms scale well with increasing problem dimension and outperform the gradient descent algorithm by several orders of magnitude in all cases.

VI. Conclusions

This paper continues to develop the distributed Newton-type family of second order algorithms for network utility maximization problem. In particular, the paper contains a novel convergence rate analysis for dual iterations that enable computation of the number of dual steps using local information, such that the error bounds sufficient for superlinear convergence to an error neighborhood around the optimal solution can be guaranteed. Simulations show that our bounds are conservative and computing the dual vector using the number of iterations suggested by our bounds essentially recovers the exact Newton direction at each step. Furthermore simulations show that an effective approximate version of this algorithm can be achieved by truncating the number of dual iterations to 1. Possible future work includes extending our convergence analysis to a "uniformly" truncated dual iterations (i.e., independent of primal iteration), analyzing asynchronous update schemes, and more broadly developing distributed second-order methods for other network resource allocation problems.

References

[1] S. Athuraliya and S. Low. Optimization flow control with Newton-like algorithm. *Journal of Telecommunication Systems*, 15:345–358, 2000.

[2] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[3] 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.

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

[5] 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.

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

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

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

[9] 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.

[10] 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.

[11] J. Mo and J Walrand. Fair end-to-end window-based congestion control. *IEEE/ACM Transactions on Networking*, 8(5), 2000.

[12] 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.

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

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

[15] J. Papandriopoulos, S. Dey, and J. Evans. Optimal and distributed protocols for cross-layer design of physical and transport layers in MANETs. *IEEE/ACM Transactions on Networking*, 16(6):1392–1405, 2008.

[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] E. Wei, A. Ozdaglar, and A. Jadbabaie. A distributed Newton method for Network Utility Maximization, I: Algorithm. *LIDS Report 2832*, 2011.

[18] E. Wei, A. Ozdaglar, and A. Jadbabaie. A distributed Newton method for Network Utility Maximization, II: Convergence. *LIDS Report 2870*, 2011.

[19] E. Wei, M. Zargham, A. Ozdaglar, and A. Jadbabaie. On dual convergence of the distributed Newton method for Network Utility Maximization. *LIDS Report 2868*, 2011.

[20] L. Xiao and S. Boyd. Optimal scaling of a gradient method for distributed resource allocation. *Journal of Optimization Theory and Applications*, 129(3):469–488, 2006.

[21] M. Zargham, A. Ribeiro, A. Ozdaglar, and A. Jadbabaie. Accelerated dual descent for network optimization. *Proceedings of the American Control Conference*, 2011.