Distributed Newton-type Algorithms for Network Resource Allocation

by

Ermin Wei

B.S., Computer Engineering
B.S., Mathematics
B.S., Finance
Minor in German Language, Literature, and Culture
University of Maryland, College Park (2008)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2010

Author .....
Department of Electrical Engineering and Computer Science
May 11, 2010

Certified by .....
Asuman Ozdaglar
Associate Professor
Thesis Supervisor

Accepted by .....
Professor Terry P. Orlando
Chair of the Committee on Graduate Students

Distributed Newton-type Algorithms for Network Resource Allocation

by

Ermin Wei

Submitted to the Department of Electrical Engineering and Computer Science
on May 11, 2010, in partial fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering

Abstract

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 motivated much interest in developing and studying distributed algorithms for network resource allocation problems, such as Internet routing, data collection and processing in sensor networks, and cross-layer communication network design. Existing works on network resource allocation problems rely on using dual decomposition and first-order (gradient or subgradient) methods, which involve simple computations and can be implemented in a distributed manner, yet suffer from slow rate of convergence. Second-order methods are faster, but their direct implementation requires computation intensive matrix inversion operations, which couple information across the network, hence cannot be implemented in a decentralized way. This thesis develops and analyzes Newton-type (second-order) distributed methods for network resource allocation problems. In particular, we focus on two general formulations: Network Utility Maximization (NUM), and network flow cost minimization problems.

For NUM problems, we develop a distributed Newton-type fast converging algorithm using the properties of self-concordant utility functions. Our algorithm utilizes novel matrix splitting techniques, which enable both primal and dual Newton steps to be computed using iterative schemes in a decentralized manner with limited information exchange. Moreover, the stepsize used in our method 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.

The second part of the thesis presents a distributed approach based on a Newton-type method for solving network flow cost minimization problems. The key component of our method is to represent the dual Newton direction as the limit of an

iterative procedure involving the graph Laplacian, which can be implemented based only on local information. Using standard Lipschitz conditions, we provide analysis for the convergence properties of our algorithm and show that the method converges superlinearly to an explicitly characterized error neighborhood, even when the iterative schemes used for computing the Newton direction and the stepsize are truncated. We also present some simulation results to illustrate the significant performance gains of this method over the subgradient methods currently used.

Thesis Supervisor: Asuman Ozdaglar
Title: Associate Professor

Acknowledgments

First and foremost, I would like to thank my dear advisor, Asu. Without her kind guidance, I wouldn't be able to come this far and accomplish the work presented here. I am grateful for her caring advice and fruitful discussions on both academic life and personal development. Her help is ubiquitous in my life at MIT: it is in the process of getting adjusted to the new environment, it is in the times when I could not move further in my research, and it is even in the small details such as whether I was happy with the office layout. Whenever it comes to interaction with her, she shines like a star, I can feel the passion and energy radiating into me and vitalize the challenging and sometimes lonesome journey of research. A special acknowledgement goes to her patience and dedication in reading my paper draft after draft, which, as we jokingly put it, can almost be modeled as an infinitely repeated game. In this process of continuing revision, I improved my writing tremendously, and there could simply be no better way in which I could have learned this much.

Another huge thank you goes to my loving parents. They share my happiness and experiences in my life, give me never-stop love and a warm family, and more importantly, when I am in the darkness, they shed light on my path, when I need someone to listen to me, they are always there, when I am scared of new walls in my life, they put me on their shoulders. Dad, your stories with eigenvalues brought laughter into the times when I was trapped to think inside a little box and bridged the jump to outside. I have many more stories to tell, and many more thanks to say. I do not know where to start, so instead I will quote a song: *You raise me up, so I can stand on mountains; You raise me up, to walk on stormy seas; I am strong, when I am on your shoulders; You raise me up: To more than I can be. — You raise me up* by Josh Groban.

To my collaborator, Professor Ali Jadbabaie from University of Pennsylvania, thanks for all the inspiring discussions. Also I would like to thank the entire Asu's group for their support and intriguing talks, special thank goes to Ozan Candogan, Kostas Bimpikis, Kimon Drakopoulos, Diego Feijer and Dr. Paul Njoroge, for being

wonderful officemates and Ishai Menache for engaging in far-reaching conversations. My work would have been impossible to accomplish without all the other friends, both those at MIT and otherwise. In particular, I would like thank Mitra Osqui, Rose Faghih, Yola Katsargyri, Mardavij Roozbehani, Fei Liang, Lei Zhang, Ying Yin, Bonnie Lam, Emi DiStefano, Chen Che and Jiarong Guo for being super supportive friends.

Last but not least, this work was partially supported by National Science Foundation under Career grant DMI-0545910 and the DARPA ITMANET program.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Recently there has been much interest in distributed control and coordination of networks comprised of multiple agents with local information and heterogeneous preferences, where the goal is to collectively optimize a global objective. This is motivated by the ubiquitous presence of large-scale networks and new networking applications, such as Internet and wireless sensor networks. One main characteristic of these networks is the lack of centralized access to information due to either security requirements or the size of the network. Therefore control and optimization algorithms deployed in such networks should be completely distributed, relying only on local information.

Standard approach for developing distributed algorithms is to use first-order methods which only rely on gradient (subgradient) information and involve simple steps. The simplicity in computation comes at the cost of slow rate of convergence, which limits application of such methods in dynamic networked environments. Second order methods are known to have much faster rate of convergence, however their implementation involves the inversion of a matrix whose size is the dimension of the problem, which is either computationally intensive or infeasible. Therefore for large scale optimization problems, even with full global knowledge, the requirement of large storage and high computation power restricts the usage of Newton-type algorithm, making

the once powerful algorithm powerless.

Motivated by the need for fast converging distributed Newton-type algorithms, this thesis focuses on two network resource allocation problems: Network Utility Maximization (NUM) problems, and network flow cost minimization problems. The objective in NUM problems is to allocate rates among sources in the network, such that the link capacity constraints are satisfied. In these problems, the network topology and routes are predetermined, each source in the network has a local utility function based on source rate, and the collective objective is to maximize overall utility, which is the sum of source utilities. Traditional distributed algorithm for this problem features a price exchange scheme and is first order in convergence speed [27], [30], [6], [14], [28]. In this thesis we propose and analyze a Newton-type second order distributed algorithm, which is significantly faster in convergence.

Similarly, the objective in (nonlinear) network flow cost minimization problems is to route the flows such that the flow conservation constraints are satisfied. In these problems, the network topology is predetermined and each link has a cost associated with it for carrying flows. Given the requirement of transporting fixed amount of flows from pre-selected sources to destinations, the overall objective is to minimize the incurred cost. Applications of this problem include but not limited to multi-commodity flow problems, supply chain management and Internet packet routing. The standard approach to this problem is to use dual decomposition and subgradient (or first-order) methods, which for some classes of problems yields iterative algorithms that operate on the basis of local information (see [30], [28], [39], and [14]). However, a major shortcoming of this approach, particularly relevant in today's large-scale networks, is the slow convergence rate of the resulting algorithms. In this thesis, we propose and analyze an alternative approach based on using Newton-type (or second-order) methods for network flow cost minimization problems.

1.2 Relevant Literature

In recent years, there has been much research in distributed control and coordination over networks. In this section, we present a brief overview of some relevant literature.

Most work in the area of multi-agent optimization, control, and learning in large-scale networked systems builds on the seminal work by Tsitsiklis [41] and Bertsekas and Tsitsiklis [10] (see also Tsitsiklis *et al.* [40]). The standard approach in this literature involves considering *consensus-based* schemes, in which agents exchange their local estimates (or states) with their neighbors with the goal of aggregating information over an exogenous (fixed or time-varying) network topology [22], [35], [24], [36], [42]. It has been shown that under some mild assumption on the connectivity of the graph and updating rules, consensus can be achieved. One application of consensus scheme is to compute the average of local estimates over a network, which will be extensively used in the development of our distributed Newton algorithms.

Pioneered by Kelly [27], followed by Low and Lapsley [30], distributed optimization algorithm over networks have been developed based on dual decomposition and subgradient (or first-order) method for network utility maximization problems. Despite the slow rate of convergence for these algorithm, these methods are appealing in view of their decentralized implementation. In these algorithms, each link charges the source traversing it a price, and then based on the aggregate price of a path, the source adjusts the flow rate accordingly. The information exchange in these works involves communication between the sources and the links. As we shall see, our algorithms will involve similar limited information exchange among the agents.

Berteskas and Gafni [3] and Athuraliya and Low in [6] have used diagonal scaling to approximate Newton steps to speed up the subgradient based distributed algorithm for multicommodity flows problem and NUM problems respectively. These algorithms incorporate some properties of the Newton-type algorithm, and feature improvements in speed over the first order methods. However, these improvements do not achieve the convergence rate behavior obtained from second order methods.

In a more recent work [11], 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 work differs from this by developing a standalone distributed Newton-type algorithm and providing analysis for the convergence properties thereof.

Our analysis for the convergence of the algorithms also relates to work on the convergence rate analysis of inexact Newton methods ([37], [26]). These works focus on providing conditions on the amount of error at each iteration relative to the norm of the gradient of the current iterate that ensures superlinear convergence to the exact optimal solution (essentially requiring the error to vanish in the limit). Even though these analyses can provide superlinear rate of convergence, the vanishing error requirement can be too restrictive for practical implementations. In our algorithms, we allow for a fixed error level to be maintained at each step of Newton direction computation and show superlinear convergence to an error neighborhood, whose size can be controlled by tuning the parameter of the algorithm. Hence our work also contributes to the literature on error analysis for inexact Newton methods.

1.3 Outline and Contributions

The rest of the thesis is organized as follows. In Chapter 2, we develop and analyze a distributed Newton-type fast converging algorithm for solving network utility maximization problems with self-concordant utility functions. Our main contribution lies in using novel matrix splitting techniques to compute both primal and dual Newton steps with iterative decentralized schemes. Moreover, the stepsize used in our method can be obtained via an iterative consensus-based averaging scheme. We show that the amount of information exchange required in this algorithm is similar to that of the methods currently used [27], [30], [6]. We also prove 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 rel-

ative to the existing subgradient methods based on dual decomposition or diagonal scaling. Hence our algorithm performs significantly faster, while it is scalable in the size of the network, and does not require significant increase in information exchange when compared with what is currently implemented.

Similarly, in Chapter 3, we develop and analyze a distributed algorithm based on Newton-type (or second-order) methods for network flow cost minimization problems. This chapter builds on the work [24] and provides a more detailed convergence rate analysis. Our main contribution lies in representing the dual Newton direction as the limit of an iterative procedure involving the graph Laplacian, which can be implemented based only on local information. Using standard Lipschitz conditions, we show that even when the iterative schemes used for computing the Newton direction and the stepsize in our method are truncated, the resulting iterates converge superlinearly within an explicitly characterized error neighborhood. Simulation results illustrate significant performance gains of this method relative to the existing methods.

Chapter 4 summarizes the thesis, and discusses possible future extensions.

Chapter 2

A Distributed Newton Method for Network Utility Maximization

2.1 Introduction

In this chapter, we study the *Network Utility Maximization (NUM) problems*, which is a general framework for formulating rate control problems in wireline networks. 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 [27], [30], [31]. One major shortcoming of this approach is the slow rate of convergence.

In this chapter 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 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 dual price vector exchange between sources and links. The second challenge is the global information required in the computation of the stepsize. We resolve this by using a consensus-based local averaging scheme.

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 versions 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. Due to inequality constraints, Lipschitz-based results cannot be applied, instead we use properties of self-concordant functions in our convergence analysis. 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.

The rest of the chapter is organized as follows: Section 2.2 defines the problem formulation and equivalent transformations thereof. Section 2.3 presents the exact constrained primal-dual Newton method for this problem. Section 2.4 presents a distributed iterative scheme for computing the dual Newton step and the distributed inexact Newton-type algorithm. Section 2.5 analyzes the rate of convergence of our algorithm. Section 2.6 presents simulation results to demonstrate convergence speed improvement of our algorithm to the existing methods with linear convergence rates. Section 2.7 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, and $[A]^j$ to denote the j^{th} row 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 $f : \mathbb{R}^n \rightarrow \mathbb{R}$, 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 $g : \mathbb{R} \rightarrow \mathbb{R}$ is said to be *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 $g : \mathbb{R}^n \rightarrow \mathbb{R}$ 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 [13] for more details).

2.2 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

$$Rs \leq c,$$

[1] One alternative definition for a real-valued convex function $g : \mathbb{R} \rightarrow \mathbb{R}$ to be *self-concordant* is that there exist a constant $a > 0$, such that $|g'''(x)| \leq 2a^{-\frac{1}{2}} g''(x)^{\frac{3}{2}}$ for all x in its domain [34], [25]. The definition we adopt is a special case of this one, where $a = 1$. Even though this alternative definition is more general, it introduces unnecessary complications for the convergence analysis.

where R is the *routing matrix* of dimension $L \times S$, i.e.,

$$R_{ij} = \begin{cases} 1 & \text{if link } i \text{ is on the route for source } j, \\ 0 & \text{otherwise.} \end{cases} \tag{2.1}$$

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

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{S} U_i(s_i) \\ \text{subject to} \quad & Rs \leq c, \\ & s \geq 0. \end{aligned} \tag{2.2}$$

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

$$\sum_{j=1}^{S} R_{lj} s_j + y_l = c_l \quad \text{for } l = 1, 2 \ldots L, \tag{2.3}$$

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 (2.2) then can be rewritten as

$$\begin{aligned} \text{minimize} \quad & -\sum_{i=1}^{S} U_i(x_i) - \mu \sum_{i=1}^{S+L} \log(x_i) \\ \text{subject to} \quad & Ax = c, \end{aligned} \tag{2.4}$$

where $A = [R\ I(L)]$, and μ is a nonnegative constant coefficient for the barrier functions. We denote by f^* the optimal objective value for the equality constrained problem (2.4). Notice that by Assumption 1 and the properties of logarithmic functions, the objective function for problem (2.4),

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

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 (2.4) approaches that of problem (2.2) [8], [20]. Therefore, in the rest of this chapter, unless clearly stated otherwise, our goal is to investigate *iterative distributed methods* for solving problem (2.4) 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 chapter.

2.3 Exact Newton Method

We consider solving problem (2.4) using a (feasible start) equality-constrained Newton method (see [13] Chapter 10). In our iterative method, we use x^k to denote the solution vector at the k^{th} step.

2.3.1 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. (2.1)].

2.3.2 Iterative Update Rule

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

$$x^{k+1} = x^k + s^k \Delta x^k, \tag{2.5}$$

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]

$$\begin{pmatrix} \nabla^2 f(x^k) & A' \\ A & 0 \end{pmatrix} \begin{pmatrix} \Delta x^k \\ w^k \end{pmatrix} = - \begin{pmatrix} \nabla f(x^k) \\ 0 \end{pmatrix}. \tag{2.6}$$

In the rest of the chapter, 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. The dual variables associated with each link capacity constraint 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 thesis. Solving for x^k and w^k in the preceding system yields

$$\Delta x^k = -H_k^{-1}(\nabla f(x^k) + A' w^k), \text{ and} \tag{2.7}$$

$$(A H_k^{-1} A') w^k = -A H_k^{-1} \nabla f(x^k). \tag{2.8}$$

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 $(A H_k^{-1} A')^{-1}$

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

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.

2.4 Distributed Inexact Newton Method

Our inexact Newton method uses the same initialization as presented in Section 2.3.1, 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. Before proceeding to present further details of the algorithm, we first introduce some preliminaries on the matrix splitting technique.

2.4.1 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.,

$$G = M + N. \tag{2.9}$$

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

$$y^{k+1} = -M^{-1}Ny^k + M^{-1}b. \tag{2.10}$$

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 [7] and [17] 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 [17]).

Theorem 2.4.1. *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 (2.10) to converge is that the matrix $M - N$ is positive definite. This can be guaranteed using Gershgorin Circle Theorem, which we introduce next (see [43] for more details).

Theorem 2.4.2. *(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 its diagonal entries strictly positive.

2.4.2 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. (2.8) in a distributed manner. Let D_k be a diagonal matrix, with diagonal entries

$$(D_k)_{ll} = (AH_k^{-1}A')_{ll}, \tag{2.11}$$

and matrix B_k be given by

$$B_k = AH_k^{-1}A' - D_k. \tag{2.12}$$

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

$$(\bar{B}_k)_{ii} = \sum_{j=1}^{L} (B_k)_{ij}. \tag{2.13}$$

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

Theorem 2.4.3. *For a given $k > 0$, let D_k, B_k, $\bar{B}_k$ be the matrices defined in Eqs. (2.11), (2.12) and (2.13). Let $w(0)$ be an arbitrary initial vector and consider the sequence $\{w(t)\}$ generated by the iteration*

$$w(t+1) = (D_k + \bar{B}_k)^{-1}(\bar{B}_k - B_k)w(t) + (D_k + \bar{B}_k)^{-1}(-AH_k^{-1}\nabla f(x^k)), \tag{2.14}$$

for all $t \geq 0$. Then the sequence $\{w(t)\}$ converges as $t \to \infty$, and its limit is the solution to Eq. (2.8).

Proof. We use a matrix splitting scheme given by

$$(AH_k^{-1}A') = (D_k + \bar{B}_k) + (B_k - \bar{B}_k) \tag{2.15}$$

and the iterative scheme presented in Eqs. (2.9) and (2.10) to solve Eq. (2.8). 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

$$Q_k = (D_k + \bar{B}_k) - (B_k - \bar{B}_k) = D_k + 2\bar{B}_k - B_k \tag{2.16}$$

denote the difference matrix. By definition of $\bar{B}_k$ [cf. Eq. (2.13)], 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 2.4.2, such matrices are positive definite. Therefore, by Theorem 2.4.1, 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 (2.15) guarantees the sequence $\{w(t)\}$ generated by iteration (2.14) to converge to the solution of Eq. (2.8). **Q.E.D.**

There can be many ways to split the matrix $AH_k^{-1}A'$, the particular one in Eq. (2.15) is chosen here due to two desirable features. First it guarantees that the difference matrix Q_k [cf. Eq. (2.16)] 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. In order to express the procedure concisely, we next define the *price of the route* for source i, $\pi_i(t)$, as the sum of the dual variables associated with the links traversed by source i at the t^{th} dual iteration, i.e., $\pi_i(t) = \sum_{l \in L(i)} w_l(t)$; and the *weighted price of the route* for source i, $\Pi_i(t)$ is defined as the price of the route for source i weighted by the element in the inverse Hessian matrix corresponding to source i, i.e., $\Pi_i(t) = (H_k^{-1})_{ii} \sum_{l \in L(i)} w_l(t)$. With this set of notation, at each primal iteration k, the dual variable can be computed as follows:

1. Initialization

   1.a Each source i sends its second derivative information $(H_k)_{ii}$ and first derivative $\nabla_i f(x^k)$ (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 uses it, i.e., $i \in S(l)$. The sources aggregates the prices along its path 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 along

i^{th} path. Each link l aggregate the total prices and computes

$$(\bar{B}_k)_{ll} = \sum_{i \in S(l)} \Pi_i(0) - \sum_{i \in S(l)} (H_k)_{ii}^{-1}. \tag{2.17}$$

1.d Initialize an arbitrary vector of dual variables as $w(1)$.

2. Iteration.

2.a Each link sends the current dual variable, i.e., $w_l(t)$ to the sources that uses it, i.e., $i \in S(l)$. Each source i aggregates the prices along its path 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 along i^{th} path, then each link l aggregate the total prices from all the sources and computes

$$\begin{aligned} w_l(t+1) =& \frac{1}{(D_k)_{ll} + (\bar{B}_k)_{ll}} ((\bar{B}_k)_{ll} w_l(t) - \sum_{i \in S(l)} \Pi_i(t) - \sum_{i \in S(l)} (H_k)_{ii}^{-1} w_l(t) \\ & - \sum_{i \in S(l)} (H_k^{-1})_{ii} \nabla_i f(x^k) - (H_k^{-1})_{(S+l)(S+l)} \nabla_{S+l} f(x^k)). \end{aligned} \tag{2.18}$$

The direction of information flow can be seen in Figures 2-1 and 2-2, with Figure 2-1 representing the direction of information flow for the steps 1.a, 1.c and 2.b in the above procedure, and Figure 2-2 representing the direction of flow for the steps 1.b and 2.a.

We now show the above procedure can be implemented in a distributed way. We first observe that due to the separable nature of the objective function, the Hessian matrix H_k is diagonal, with dimension $(S+L) \times (S+L)$, whose entries are of the form

$$(H_k)_{ii} = \begin{cases} -\frac{\partial^2 U_i(x_i^k)}{\partial x_i^2} + \frac{\mu}{(x_i^k)^2} & 1 \leq i \leq S, \\ \frac{\mu}{(x_i^k)^2} & S+1 \leq i \leq S+L. \end{cases}$$

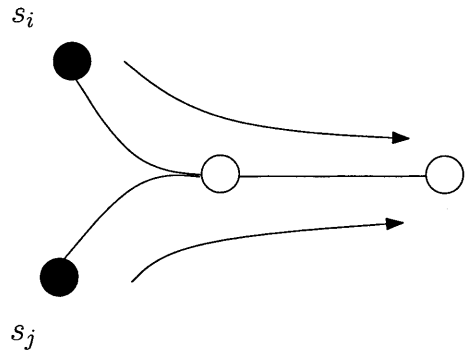

Figure 2-1: Direction of information flow for the steps 1.a, 1.c and 2.b, from sources to the links they use.

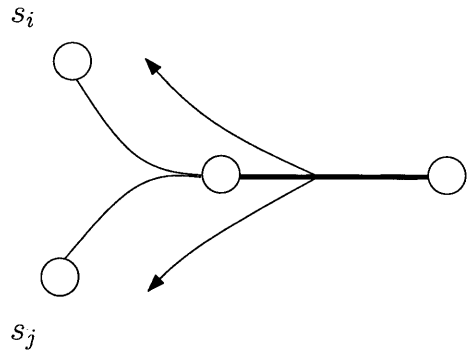

Figure 2-2: Direction of flow for the steps 1.b and 2.a, from links to the sources using them.

Each source knows its own utility function, barrier function and source rate, therefore the first S entries of the form $(H_k)_{ii}$ can be computed by the sources in a distributed manner. Similarly, since each link has information about its corresponding slack variable and barrier function, the last L entries of the form $(H_k)_{ii}$ can be computed by the links in a distributed way. The similar argument holds for the gradient vector. Therefore, with information received from step 1.a, the l^{th} diagonal entry of the matrix D_k, i.e.,

$$(D_k)_{ll} = (AH_k^{-1}A')_{ll} = \sum_{i \in S(l)} (H_k)_{ii}^{-1} + (H_k)_{(S+l)(S+l)}^{-1}, \tag{2.19}$$

[cf. Eq. (2.11)], can be calculated by link l with only information about its slack variable and the Hessian information from the sources $i \in S(l)$. Also each link l also has the gradient $\nabla_i f(x^k)$ for $i \in S(l)$, and therefore each link l can compute $\sum_{i \in S(l)} (H_k^{-1})_{ii} \nabla_i f(x^k)$ individually. Hence all information in step 1.a can be obtained in a decentralized manner.

Using the definition of the price of the routes, $\pi_i(t)$, once the links send out their associated price, each source i can compute $\pi_i(t)$ in a distributed way. We use the fact that the Hessian information $(H_k)_{ii}$ can be computed at each source one more time, and conclude the weighted prices $\Pi_i(t)$ can be calculated in a distributed manner, and therefore steps 1.b and 2.a can be completed using only local information.

In step 1.c, each link aggregates the weighted prices $\Pi_i(0)$ sent out via the sources that use it, and uses Hessian information from step 1.a to calculate $(\bar{B}_k)_{ll}$. It can

clearly be done in a distributed way. The similar argument hold for step 2.b, by noting that the term $(H_k^{-1})_{(S+l)(S+l)}\nabla_{S+l}f(x^k)$ only depends on the slack variable and the barrier function at the l^{th} link.

Hence all the steps can be implemented in a decentralized way. We next show the limit point of $w(t)$ generated by the dual variable computation procedure above solves Eq. (2.8) by establishing the following lemma.

Lemma 2.4.4. *The iterative dual variable computation procedure defined above coincides with the iteration (2.14).*

Proof. Recall the definition of the matrix A, i.e., $A_{ji} = 1$ for $i = 1, 2 \ldots S$ if link j is on the route for source i, and $A_{ji} = 0$ otherwise. Therefore, we can write the price of the route for source i as, $\pi_i(t) = \sum_{j=1}^{L} A_{ji}w(t)_j = [A']^i w(t)$. Similarly since the Hessian matrix H is diagonal, the weighted price can be written as

$$\Pi_i(t) = (H_k)_{ii}^{-1}[A']^i w(t) = [H_k^{-1}A']^i w(t). \tag{2.20}$$

On the other hand, from basic linear algebra, and the fact that $A = [R\ I(L)]$, where R is the routing matrix, we have the following relation hold,

$$\begin{aligned}(AH_k^{-1}A'w(t))_l &= \sum_{i=1}^{S}([A]_i[H_k^{-1}A']^i w(t))_l + (H_k^{-1})_{(S+l)(S+l)}w_l(t) \\ &= \sum_{i=1}^{S} A_{li}([H_k^{-1}A']^i w(t)) + (H_k^{-1})_{(S+l)(S+l)}w_l(t),\end{aligned}$$

where $[A]_i$ denotes the i^{th} column of the matrix A, and $[A]^j$ to denotes the j^{th} row of the matrix A. Using the definition of the matrix A one more time, the above relation implies,

$$\begin{aligned}(AH_k^{-1}A'w(t))_l &= \sum_{i\in S(l)} [H_k^{-1}A']^i w(t) + (H_k^{-1})_{(S+l)(S+l)}w_l(t) \\ &= \sum_{i\in S(l)} \Pi_i(t) + (H_k^{-1})_{(S+l)(S+l)}w_l(t),\end{aligned} \tag{2.21}$$

where the last equality follows from relation (2.20).

Using Eq. (2.12), the above relation implies that $((B_k+D_k)w(t))_l = \sum_{i\in S(l)} \Pi_i(t)$. We next show that the value $(\bar{B}_k)_{ll}$ can be computed at link l in a decentralized way. Using the fact that $w(0) = [1, 1\ldots, 1]'$, we have

$$(AH_k^{-1}A'w(0))_l = ((B_k + D_k)w(0))_l = \sum_{j=1}^{L}(B_k)_{lj} + (D_k)_{ll}.$$

Since $(D_k)_{ll}$ is known to the l^{th} link, it can compute the value $(\bar{B}_k)_{ll}$, by definition of $\bar{B}_k$ [cf. Eq. (2.13)] and relation (2.21) we have

$$\begin{aligned}(\bar{B}_k)_{ll} &= \sum_{j=1}^{L}(B_k)_{lj} = (AH_k^{-1}A'w(0))_l - (D_k)_{ll} \\ &= \sum_{i\in S(l)} \Pi_i(0) + (H_k^{-1})_{(S+l)(S+l)}w_l(t) - (D_k)_{ll},\end{aligned}$$

where the last equality follows from relation (2.21). This calculation can further be simplified using relation (2.19), and we obtain Eq. (2.17) in step 1.c, i.e.,

$$(\bar{B}_k)_{ll} = \sum_{i\in S(l)} \Pi_i(t) - \sum_{i\in S(l)} (H_k)_{ii}^{-1}.$$

Following the same procedure, the value $(B_kw(t))_l$ can be written as

$$\begin{aligned}(B_kw(t))_l &= (AH_k^{-1}A'w(t))_l - (D_kw(t))_l \\ &= \sum_{i=1}^{S} \Pi_i(t) + (H_k^{-1})_{(S+l)(S+l)}w_l(t) - (D_k)_{ll}w_l(t) \\ &= \sum_{i=1}^{S} \Pi_i(t) - \sum_{i\in S(l)} (H_k)_{ii}^{-1}w_l(t),\end{aligned}$$

where the first equality follows from Eq. (2.13), the second equality follows from Eq. (2.21), and the last equality follows from Eq. (2.19).

We can write $(AH_k^{-1}\nabla f(x^k))_l$ as,

$$(AH_k^{-1}\nabla f(x^k))_l = \sum_{i\in S(l)} (H_k^{-1})_{ii}\nabla_i f(x^k) + (H_k^{-1})_{(S+l)(S+l)}\nabla_{S+l} f(x^k).$$

By substituting the previous two relations into iteration (2.14) we obtain the desired iteration, i.e., iteration (2.18). **Q.E.D.**

The above lemma, together with Theorem 2.4.3 guarantees the dual variable computation procedure generates the desired solution to Eq. (2.8).

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. This can be seen from the fact that the computations done in the initialization phase only needs to be executed once.

2. This algorithm has comparable level of information exchange with the existing subgradient based algorithms applied to the NUM problem (2.2) (see [6], [27], [30], [31] for more details). In both types of algorithms, only the sum of dual variables of links along a source path is fed back to the source, and the link updates the price based on some aggregate information from all the sources that use the particular link. The above procedure is designed for the Newton-type algorithm, admittedly the computation here is more complicated than simply summing up, since it involves multiplication and division. However since all operations are scalar-based, the computation complexity should not impose degradation on the performance of the algorithm given today's technology.

3. The initialization phase was not present in the existing first order algorithm, however this extra 3 rounds of information exchange enables the implementation of Newton-type method, and speeds up the algorithm significantly, as we show in Section 2.5.

4. The dual iterations can use the pilot signal $w_l(0) = 1$ as initialization in step

1.d, however if the links can each store the value of the dual variable from the previous primal iteration w_l^{k-1}, then simulation results suggest the dual iteration converges faster.

2.4.3 Distributed Computation of the Newton Primal Direction

Once the dual variables are obtained, the primal Newton direction can be solved according to (2.7), 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 (2.4). 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. (2.7) 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,

$$x^{k+1} = x^k + s^k \Delta\tilde{x}^k, \tag{2.22}$$

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 2.4.6, 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 (2.4) are well defined.

We refer to the exact solution to the system of equations (2.6) 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

$$\lambda(x^k) = \sqrt{(\Delta x^k)' \nabla^2 f(x^k) \Delta x^k}. \tag{2.23}$$

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

$$\tilde{\lambda}(x^k) = \sqrt{(\Delta\tilde{x}^k)' \nabla^2 f(x^k) \Delta\tilde{x}^k}. \tag{2.24}$$

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 2.5, 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 [41], [22], [36] 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

$$s^k = \begin{cases} \frac{c}{\theta^k+1} & \text{if } \theta^k \geq \frac{1}{4}, \\ 1 & \text{otherwise,} \end{cases} \tag{2.25}$$

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 2.4.6 and Section 2.5.2 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 estimate for $\tilde{\lambda}(x^k)$ obtained via an iterative scheme, which also introduces inexactness to the algorithm. 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*

$$\Delta x^k = \Delta\tilde{x}^k + \gamma, \tag{2.26}$$

where γ is bounded by

$$|\gamma' \nabla^2 f(x^k)\gamma| \leq p^2(\Delta\tilde{x}^k)'\nabla^2 f(x^k)\Delta\tilde{x}^k + \epsilon. \tag{2.27}$$

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.,*

$$\tau^k = \tilde{\lambda}(x^k) - \theta^k, \tag{2.28}$$

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 2.4.5 and Theorem 2.4.6. For the rest of the chapter, we assume the conditions in Assumptions 1-3 hold.

We now show that the stepsize choice in (2.25) 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 (2.4) are well defined. We proceed by first establishing a bound on the error in the stepsize calculation.

Lemma 2.4.5. *Let $\tilde{\lambda}(x^k)$ be the inexact Newton decrement defined in (2.24), 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 (2.25). For $\theta^k \geq \frac{1}{4}$, the following relation holds*

$$(2c-1)/(\tilde{\lambda}(x^k)+1) \leq \frac{c}{\theta^k + 1} \leq 1/(\tilde{\lambda}(x^k)+1). \tag{2.29}$$

Proof. By Assumption 3 and the fact that $\theta^k \geq \frac{1}{4}$ we have

$$|\tilde{\lambda}(x^k) - \theta^k| \leq \left(\frac{1}{c} - 1\right)\frac{5}{4} \leq \left(\frac{1}{c} - 1\right)(1 + \theta^k). \tag{2.30}$$

By multiplying both sides by the positive scalar c, the above relation implies

$$c\theta^k - c\tilde{\lambda}(x^k) \leq (1-c)(1+\theta^k),$$

which further implies,

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

By dividing both sides of the above relation by the positive scalar $(\theta^k+1)(\tilde{\lambda}(x^k)+1)$, we obtain the first inequality in relation (2.29).

Similarly, using relation (2.30) we can also establish

$$c\tilde{\lambda}(x^k) - c\theta^k \leq (1-c)(1+\theta^k),$$

which can be rewritten as,

$$c\tilde{\lambda}(x^k) + c \leq \theta^k + 1.$$

After dividing both sides of the preceding relation by the positive scalar $(\theta^k+1)(\tilde{\lambda}(x^k)+1)$, we obtain the second inequality in relation (2.29). **Q.E.D.**

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 2.4.6. *Let x^0 be a positive feasible primal variable, x^k be the sequence of primal variables updated using iteration (2.22), 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 2.4.3, and s^k is defined as in (2.25). Then for all k, the primal variable satisfies $x^k > 0$.*

Proof. We will prove this claim by induction. The base case of $x^0 > 0$ holds by the assumption of the theorem. At a primal solution x^k, by Assumption 1, the utility functions U_i are strictly concave, hence $-\frac{\partial^2 U_i}{\partial x_i^2}(x_i^k) \geq 0$. Given the form of the Hessian matrix,

$$(H_k)_{ii} = \begin{cases} -\frac{\partial^2 U_i}{\partial x_i^2}(x_i^k) + \frac{\mu}{(x_i^k)^2} \geq \frac{\mu}{(x_i^k)^2} & 1 \leq i \leq S, \\ \frac{\mu}{(x_i^k)^2} & S+1 \leq i \leq S+L, \end{cases}$$

we have

$$\tilde{\lambda}(x^k) = \left(\sum_{i=1}^{S+L} (\Delta\tilde{x}_i^k)^2 (H_k)_{ii}\right)^{\frac{1}{2}} \geq \left(\sum_{i=1}^{S+L} \mu \left(\frac{\Delta\tilde{x}_i^k}{x_i^k}\right)^2\right)^{\frac{1}{2}} \geq \max_i \left|\frac{\sqrt{\mu}\Delta\tilde{x}_i^k}{x_i^k}\right|,$$

where the last inequality follows from the nonnegativity of the terms $\mu\left(\frac{\tilde{\Delta}x_i^k}{x_i^k}\right)^2$. By taking the reciprocal on both sides, the above relation implies

$$\frac{1}{\tilde{\lambda}(x^k)} \leq \frac{1}{\max_i \left|\frac{\sqrt{\mu}\Delta\tilde{x}_i^k}{x_i^k}\right|} = \frac{1}{\sqrt{\mu}} \min_i \left|\frac{x_i^k}{\Delta\tilde{x}_i^k}\right| \leq \min_i \left|\frac{x_i^k}{\Delta\tilde{x}_i^k}\right|, \tag{2.31}$$

where the last inequality follows from the fact that $\mu \geq 1$.

We show the inductive step by considering two cases.

- Case i: $\theta^k \geq \frac{1}{4}$

  By Lemma 2.4.5, we have the stepsize s^k satisfies, $s^k \leq 1/(1+\tilde{\lambda}(x^k)) < 1/\tilde{\lambda}(x^k)$. Thus using relation (2.31), we obtain $s^k < \min_i \left|\frac{x_i^k}{\Delta\tilde{x}_i^k}\right|$, and hence if $x^k > 0$, then $x^{k+1} = x^k + s^k\Delta\tilde{x}^k > 0$.

- Case ii: $\theta^k < \frac{1}{4}$

  By Assumption 3, we have $\tilde{\lambda}(x^k) < \frac{1}{4} + \left(\frac{1}{c} - 1\right)\frac{5}{4}$. Using the fact that $c > \frac{5}{6}$, we obtain

  $$\tilde{\lambda}(x^k) < \frac{1}{4} + \left(\frac{1}{c} - 1\right)\frac{5}{4} < \frac{1}{4} + \left(\frac{6}{5} - 1\right)\frac{5}{4} < 1.$$

  Hence we have $s^k = 1 < \frac{1}{\tilde{\lambda}(x^k)} \leq \min_i |\frac{x_i^k}{\tilde{\Delta}x_i^k}|$, where the last inequality follows from relation (2.31). Once again, if $x^k > 0$, then $x^{k+1} = x^k + s^k\Delta\tilde{x}^k > 0$.

Therefore in both cases we have $x^{k+1} = x^k + s^k\Delta\tilde{x}^k > 0$, which completes the induction proof. **Q.E.D.**

Therefore our algorithm guarantees the objective function of problem (2.4) is well defined throughout.

2.5 Convergence Analysis

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

2.5.1 Convergence in Dual Iterations

We present an explicit rate of convergence for the iterations in our dual variable computation procedure described in Section 2.4.2. We need the following lemma.

Lemma 2.5.1. *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 $1 > |\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_n|$ and let v_i denote the set of corresponding unit length right eigenvectors. Assume the matrix has n linearly independent eigenvectors* [3] *. Then for the sequence $w(t)$ generated by the following iteration*

$$w(t+q) = M^q w(t), \tag{2.32}$$

we have

$$||w(t) - w^*|| \leq \lambda_1^t \alpha, \tag{2.33}$$

for some positive scalar α, where w^* is the limit of iteration (2.32) as $t \to \infty$.

Proof. We let V denote the $n \times n$ matrix, whose columns are the n eigenvectors of matrix M, i.e., $[V]_i = v_i$. Let D be the diagonal matrix, with $D_{ii} = \lambda_i$. Since the matrix M has n linearly independent eigenvectors, which span $\mathbb{R}^n$. Then using basic linear algebra [4], we can show that

$$MV = VD. \tag{2.34}$$

Using the columns in V as a basis, we perform a change of basis and $w(t)$ can be

[3] An alternative assumption is that the algebraic multiplicity of each λ_i is equal to its corresponding geometric multiplicity, since eigenvectors associated with different eigenvalues are independent [29].

[4] A review of such material can be found in [29], [21], [5] and [18].

written as $w(t) = V\bar{w}(t)$ for some $\bar{w}(t)$. Hence the iteration (2.32) implies

$$w(t+1) = MV\bar{w}(t) = VD\bar{w}(t).$$

By applying relation (2.34) to the above relation iteratively, we obtain

$$w(t+q) = VD^q\bar{w}(t). \tag{2.35}$$

Since the matrix D is diagonal, and $D_{ii} < 1$ for all i, we have $\lim_{q\to\infty} D^q = 0$, and therefore

$$w^* = \lim_{q\to\infty} VD^q\bar{w}(t) = 0.$$

The above relation, combined with relation (2.35), implies with some arbitrary initial vector $w(0)$, we have

$$||w(t) - w^*|| = \left|\left|VD^t\bar{w}(0)\right|\right| = \left|\left|\sum_{i=1}^{n} D_{ii}^t\bar{w}_i(0)v_i\right|\right|.$$

We next apply triangle inequality and obtain

$$||w(t) - w^*|| \leq \sum_{i=1}^{n}\left|\left|D_{ii}^t\bar{w}_i(0)v_i\right|\right| = \sum_{i=1}^{n}|D_{ii}^t\bar{w}_i(0)| \leq \lambda_1^t\sum_{i=1}^{n}|\bar{w}_i(0)|,$$

where the equality follows from the fact that the eigenvectors, v_i, are all unit length, and the last inequality follows by the fact that λ_1 is the eigenvalue with largest magnitude. By setting $\alpha = \sum_{i=1}^{n}|\bar{w}_i(0)|$ we have shown the desired relation (2.33). **Q.E.D.**

As Lemma 2.4.4 shows, our dual variable computation algorithm implements iteration (2.14). For notational simplicity, let M by the $L \times L$ matrix with $M = (D_k + \bar{B}_k)^{-1}(\bar{B}_k - B_k)$, and $z = (D_k + \bar{B}_k)^{-1}(-AH_k^{-1}\nabla f(x^k))$, then iteration (2.14)

can be written as $w(t+1) = Mw(t) + z$, which implies

$$w(t+q) = M^q w(t) + \sum_{i=0}^{q-1} M^i z = M^q w(t) + (I - M^{q+1})(I - M)^{-1} z.$$

This representation is possible due to the fact that $\rho(M) < 1$, which is immediate from the proof of Theorem 2.4.3. After rearranging the terms, we obtain

$$w(t+q) = M^q (w(t) - M(I - M)^{-1} z) + (I - M)^{-1} z.$$

Therefore starting from some arbitrary initial vector $w(0)$, the convergence properties of the sequence $w(t)$ coincides with a sequence $u(t)$, generated by $u(t+q) = M^q u(0)$, where $u(0) = w(0) - M(I - M)^{-1} z$. We first assume the matrix M has L linearly independent eigenvectors, then by applying the preceding lemma, we have

$$||w(t) - w^*|| = ||u(t) - u^*|| \leq \lambda_1^t \alpha,$$

where λ_1 is the eigenvalue of M with largest magnitude, and α is a constant depending on the initial vector $u(0) = w(0) - M(I - M)^{-1} z$. Hence number of iteration our procedure takes to reach desired level of precision depends on the eigenvalue with the largest magnitude of the matrix $M = (D_k + \bar{B}_k)^{-1}(\bar{B}_k - B_k)$. The smaller the magnitude is, the faster the iterations converge. The rate of convergence for the dual iteration is linear.

For completeness, we next examine the case when the matrix M has less than L linearly independent eigenvectors, even though this set of matrices M has measure zero. By Jordan normal form, we have for some invertible matrix V and block diagonal matrix J, $MV = VJ$. The similar arguments applies as in Lemma 2.5.1, we have the

$$w(t+q) = VJ^q \bar{w}(t),$$

for some $\bar{w}(t)$, which is the representation for the vector $w(t)$ under the basis formed by the columns of V. Hence the rate of convergence of $w(t)$ depends on how fast the

matrix J^t diminishes to 0. Let $J(\lambda)$ denote the Jordan block corresponding to the eigenvalue λ. Then $J(\lambda) = \lambda I(k+1) + N(k+1)$, where k is the difference between algebraic and geometric multiplicity of eigenvalue λ, and $N(k+1)$ is a matrix of dimension $(k+1) \times (k+1)$, with

$$N(k+1) = \begin{pmatrix} 0 & 1 & \dots & 0 \\ 0 & 0 & 1 & \\ & & \ddots & \ddots \\ 0 & \dots & 0 & 1 \\ 0 & \dots & 0 & 0 \end{pmatrix},$$

which is nilpotent. For each Jordan block, by binomial formula, we have

$$(J(\lambda))^q = (\lambda I(k+1) + N(k+1))^q = \lambda^q I(k+1) + \sum_{i=1}^{q} \begin{pmatrix} q \\ i \end{pmatrix} \lambda^{q-i} (N(k+1))^i.$$

Due to the fact that $N^j(k+1) = 0$ for $j > k+1$, the above relation can be simplified to

$$\begin{aligned} (J(\lambda))^q &= \lambda^q I(k+1) + \sum_{i=1}^{q+1} \begin{pmatrix} q \\ i \end{pmatrix} \lambda^{q-i} (N(k+1))^i \\ &\leq \lambda^q I(k+1) + \sum_{i=1}^{k+1} \frac{q^i}{i!} \lambda^{q-i} (N(k+1))^i, \end{aligned}$$

where the inequality follows from expansion of the binomial coefficient. Recall that by Theorem 2.4.3, we have the spectral radius of M satisfying $\rho(M) < 1$, and hence $|\lambda| < 1$, the above matrix $(J(\lambda))^q$ converges to 0. The rate of convergence is, however, slower than before, due to the effect of the second term. We can observe that the larger k is, the slower the convergence is. Also the smaller $|\lambda|$ is, the faster the exponential λ^{q-i} in the second term dominates the polynomial term q^i, and hence the faster it converges to 0.

2.5.2 Convergence in Primal Iterations

We next present our convergence analysis for the primal solution generated by the inexact Newton algorithm defined in Eq. (2.22). For the k^{th} iteration, we define the function $\tilde{f}_k : \mathbb{R} \to \mathbb{R}$ as

$$\tilde{f}_k(t) = f(x^k + t\Delta\tilde{x}^k), \tag{2.36}$$

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.

Preliminaries

Using the definition of a self-concordant function, we can obtain the following result (see [13] for the proof).

Lemma 2.5.2. *Let $\tilde{f} : \mathbb{R} \to \mathbb{R}$ be a self-concordant function. Then for all $t \geq 0$ in the domain of the function $\tilde{f}$ with $t\tilde{f}''(0)^{\frac{1}{2}} < 1$, the following inequality holds:*

$$\tilde{f}(t) \leq \tilde{f}(0) + t\tilde{f}'(0) - t\tilde{f}''(0)^{\frac{1}{2}} - \log(1 - t\tilde{f}''(0)^{\frac{1}{2}}). \tag{2.37}$$

We will use the preceding lemma to prove a key relation in analyzing convergence properties of our algorithm [cf. Lemma 2.5.7]. The next lemma will be used to relate the weighted norms of a vector z, with weights $\nabla^2 f(x)$ and $\nabla^2 f(y)$ for some x and y. This lemma plays an essential role in establishing properties for the Newton decrement (see [25], [34] for more details).

Lemma 2.5.3. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be self-concordant. Suppose vectors x and y are in the domain of f and $\tilde{\lambda} = ((x-y)'\nabla^2 f(x)(x-y))^{\frac{1}{2}} < 1$, then for any $z \in \mathbb{R}^n$, the*

following inequality holds:

$$(1-\tilde{\lambda})^2 z'\nabla^2 f(x)z \leq z'\nabla^2 f(y)z \leq \frac{1}{(1-\tilde{\lambda})^2} z'\nabla^2 f(x)z. \tag{2.38}$$

Using the above lemma we 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 [25] and [34] 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 2.5.4. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a self-concordant convex function. Consider the unconstrained optimization problem*

$$minimize_{x\in\mathbb{R}^n} f(x).$$

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,

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

Proof. For any $t < 1$, $(x - x(t))'\nabla^2 f(x)(x - x(t)) = t^2\tilde{\lambda}^2 < 1$ and by Lemma 2.5.3 for any z, we have

$$(1-t\tilde{\lambda})^2 z'\nabla^2 f(x)z \leq z'\nabla^2 f(x(t))z \leq \frac{1}{(1-t\tilde{\lambda})^2} z'\nabla^2 f(x)z$$

which implies

$$z'(\nabla^2 f(x(t)) - \nabla^2 f(x))z \leq \left(\frac{1}{(1-t\tilde{\lambda})^2} - 1\right) z'\nabla^2 f(x)z, \tag{2.39}$$

and

$$z'(\nabla^2 f(x) - \nabla^2 f(x(t)))z \leq \left(1 - (1-t\tilde{\lambda})^2\right) z'\nabla^2 f(x)z.$$

Using the fact that $1-(1-t\tilde{\lambda})^2 \leq \frac{1}{(1-t\tilde{\lambda})^2} - 1$, the preceding relation can be rewritten as

$$z'(\nabla^2 f(x) - \nabla^2 f(x(t)))z \leq \left(\frac{1}{(1-t\tilde{\lambda})^2} - 1\right) z'\nabla^2 f(x)z. \tag{2.40}$$

Combining relations (2.39) and (2.40) yields

$$\left|z'(\nabla^2 f(x) - \nabla^2 f(x(t)))z\right| \leq \left(\frac{1}{(1-t\tilde{\lambda})^2} - 1\right) z'\nabla^2 f(x)z. \tag{2.41}$$

Since the function f is convex, the Hessian matrix $\nabla^2 f(x)$ is positive semidefinite. We can therefore apply the generalized Cauchy-Schwarz inequality and obtain

$$\begin{aligned} &\left|(\Delta\tilde{x})'(\nabla^2 f(x(t)) - \nabla^2 f(x))z\right| \\ &\leq \sqrt{(\Delta\tilde{x})'(\nabla^2 f(x(t)) - \nabla^2 f(x))\Delta\tilde{x}'}\sqrt{z'(\nabla^2 f(x(t)) - \nabla^2 f(x))z} \\ &\leq \left(\frac{1}{(1-t\tilde{\lambda})^2} - 1\right)\sqrt{(\Delta\tilde{x})'\nabla^2 f(x)\Delta\tilde{x}}\sqrt{z'\nabla^2 f(x)z} \\ &= \left(\frac{1}{(1-t\tilde{\lambda})^2} - 1\right)\tilde{\lambda}\sqrt{z'\nabla^2 f(x)z}, \end{aligned} \tag{2.42}$$

where the second inequality follows from relation (2.41), and the equality follows from definition of $\tilde{\lambda}$.

Define the function $\kappa : \mathbb{R} \to \mathbb{R}$, as $\kappa(t) = \nabla f(x(t))'z + (1-t)(\Delta\tilde{x})'\nabla^2 f(x)'z$, then

$$\left|\frac{d}{dt}\kappa(t)\right| = \left|(\Delta\tilde{x})'\nabla^2 f(x(t))'z - (\Delta\tilde{x})'\nabla^2 f(x)z\right| = \left|(\Delta\tilde{x})'(\nabla^2 f(x(t)) - \nabla^2 f(x))z\right|,$$

which is the left hand side of (2.42).

The exact Newton direction Δx satisfies $\Delta x = -\nabla^2 f(x)^{-1}\nabla f(x)$ and hence $\Delta x'\nabla^2 f(x) = -\nabla f(x)'$. Using the fact that $\Delta x = \Delta\tilde{x} + \gamma$, we obtain

$$\kappa(0) = \nabla f(x)'z + (\Delta\tilde{x})'\nabla^2 f(x)'z = \nabla f(x)'z - \nabla f(x)'z + \gamma'\nabla^2 f(x)'z = \gamma'\nabla^2 f(x)'z.$$

Hence by integration we obtain the bound

$$|\kappa(t)| \leq \tilde{\lambda}\sqrt{z'\nabla^2 f(x)z}\int_0^t \left(\frac{1}{(1-s\tilde{\lambda})^2} - 1\right) ds + |\gamma'\nabla^2 f(x)'z|$$
$$= \frac{\tilde{\lambda}^2 t^2}{1-\tilde{\lambda}t}\sqrt{z'\nabla^2 f(x)z} + |\gamma'\nabla^2 f(x)'z|.$$

For $t = 1$, $x(t) = x + \Delta\tilde{x}$, above equation implies

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

Finally since z is the exact Newton direction at $x+\Delta\tilde{x}$, z satisfies $z\nabla^2 f(x+\Delta\tilde{x})'z = |\nabla f(x+\Delta\tilde{x})'z|$, which proves the desired relation. **Q.E.D.**

The above lemma will be used to guarantee quadratic rate of convergence for our distributed inexact Newton method [cf. Section 2.5.2]. The next lemma plays a central role in relating the suboptimality gap in the objective function value and the exact Newton decrement (see [13] for more details).

Lemma 2.5.5. *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 f^* denote the optimal objective function value. If $\lambda(x^k) \leq 0.68$, then the following relation holds,*

$$f^* \geq f(x^k) - \lambda(x^k)^2. \tag{2.43}$$

The number 0.68 is obtained based on numerical simulation by [13]. The above lemmas are established for the unconstrained Newton method. However as shown in [13], for each equality constrained optimization problem, via elimination technique we can construct an unconstrained problem, such that both the Newton decrement and the Newton primal direction coincide in both problems. Hence the above lemmas can be applied for the constrained Newton method also. We will use extensively these lemmas in the subsequent sections for rate of convergence analysis for our algorithm,

which 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.

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 2.5.6. *Let $\tilde{\lambda}(x^k)$ be the inexact Newton decrement defined in Eq. (2.24). 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.

Proof. By Assumption 1, the Hessian matrix $\nabla^2 f(x^k)$ is positive definite for all x^k. We therefore can apply the generalized Cauchy-Schwarz inequality and obtain

$$\begin{aligned} |\gamma' \nabla^2 f(x^k) \Delta \tilde{x}^k| &\leq \sqrt{(\gamma' \nabla^2 f(x^k)\gamma)((\Delta \tilde{x}^k)' \nabla^2 f(x^k) \Delta \tilde{x}^k)} \\ &\leq \sqrt{(p^2 \tilde{\lambda}(x^k)^2 + \epsilon)\tilde{\lambda}(x^k)^2} \\ &\leq \sqrt{(p^2 \tilde{\lambda}(x^k)^2 + \epsilon + 2p\tilde{\lambda}(x^k)\sqrt{\epsilon})\tilde{\lambda}(x^k)^2}, \end{aligned} \tag{2.44}$$

where the second inequality follows from Assumption 2 and definition of $\tilde{\lambda}(x^k)$, and the third inequality follows by adding the nonnegative term $2p\sqrt{\epsilon}\tilde{\lambda}(x^k)^3$ to the right hand side. By the nonnegativity of the inexact Newton decrement $\tilde{\lambda}(x^k)$, it can be seen that relation (2.44) implies

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

which proves the desired relation. **Q.E.D.**

Using the preceding lemma, the following basic relation can be established, which will be used to measure the improvement in the objective function value.

Lemma 2.5.7. *Let $\tilde{f}_k(t)$ and $\tilde{\lambda}(x^k)$ be the functions defined in Eqs. (2.36) and (2.24) respectively. Then the following relation holds for all k with $0 \leq t < 1/\tilde{\lambda}(x^k)$,*

$$\tilde{f}_k(t) \leq \tilde{f}_k(0) - t(1-p)\tilde{\lambda}(x^k)^2 - (1-\sqrt{\epsilon})t\tilde{\lambda}(x^k) - \log(1-t\tilde{\lambda}(x^k)), \tag{2.45}$$

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

Proof. Recall that Δx^k is the exact Newton direction, which solves the system (2.6). Therefore for some w^k, the following equation is satisfied,

$$\nabla^2 f(x^k)\Delta x^k + A'w^k = -\nabla f(x^k).$$

By left multiplying the above relation by $(\Delta\tilde{x}^k)'$, we obtain

$$(\Delta\tilde{x}^k)'\nabla^2 f(x^k)\Delta x^k + (\Delta\tilde{x}^k)'A'w^k = -(\Delta\tilde{x}^k)'\nabla f(x^k).$$

Using the facts that $\Delta x^k = \Delta\tilde{x}^k + \gamma$ from Assumption 2 and $A\Delta\tilde{x}^k = 0$ by the design of our algorithm, the above relation yields

$$(\Delta\tilde{x}^k)'\nabla^2 f(x)\Delta\tilde{x}^k + (\Delta\tilde{x}^k)'\nabla^2 f(x^k)\gamma = -(\Delta\tilde{x}^k)'\nabla f(x^k).$$

By Lemma 2.5.6, we can bound $(\Delta\tilde{x}^k)'\nabla^2 f(x^k)\gamma$ by,

$$p\tilde{\lambda}(x^k)^2 + \tilde{\lambda}(x^k)\sqrt{\epsilon} \geq (\Delta\tilde{x}^k)'\nabla^2 f(x^k)\gamma \geq -p\tilde{\lambda}(x^k)^2 - \tilde{\lambda}(x^k)\sqrt{\epsilon}.$$

Using the definition of $\tilde{\lambda}(x^k)$ [cf. Eq. (2.24)] and the preceding two relations, we obtain the following bounds on $(\Delta\tilde{x}^k)'\nabla f(x^k)$:

$$-(1+p)\tilde{\lambda}(x^k)^2 - \tilde{\lambda}(x^k)\sqrt{\epsilon} \leq (\Delta\tilde{x}^k)'\nabla f(x^k) \leq -(1-p)\tilde{\lambda}(x^k)^2 + \tilde{\lambda}(x^k)\sqrt{\epsilon}.$$

By differentiating the function $\tilde{f}_k(t)$, and using the preceding relation, this yields,

$$\begin{aligned} \tilde{f}_k'(0) &= \nabla f(x^k)' \Delta \tilde{x}^k \\ &\leq -(1-p)\tilde{\lambda}(x^k)^2 + \tilde{\lambda}(x^k)\sqrt{\epsilon}. \end{aligned} \tag{2.46}$$

Moreover, we have

$$\begin{aligned} \tilde{f}_k''(0) &= (\Delta \tilde{x}^k)' \nabla^2 f(x^k) \Delta \tilde{x}^k \\ &= \tilde{\lambda}(x^k)^2. \end{aligned} \tag{2.47}$$

The function $\tilde{f}_k(t)$ is self-concordant for all k, therefore by Lemma 2.5.2, for $0 \leq t < 1/\tilde{\lambda}(x^k)$, the following relations hold:

$$\begin{aligned} \tilde{f}_k(t) &\leq \tilde{f}_k(0) + t\tilde{f}_k'(0) - t\tilde{f}_k''(0)^{\frac{1}{2}} - \log(1 - t\tilde{f}_k''(0)^{\frac{1}{2}}) \\ &\leq \tilde{f}_k(0) - t(1-p)\tilde{\lambda}(x^k)^2 + t\tilde{\lambda}(x^k)\sqrt{\epsilon} - t\tilde{\lambda}(x^k) - \log(1 - t\tilde{\lambda}(x^k)) \\ &= \tilde{f}_k(0) - t(1-p)\tilde{\lambda}(x^k)^2 - (1-\sqrt{\epsilon})t\tilde{\lambda}(x^k) - \log(1 - t\tilde{\lambda}(x^k)), \end{aligned}$$

where the second inequality follows by Eqs. (2.46) and (2.47). This proves relation (2.45). **Q.E.D.**

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.

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. (2.25)]. 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 2.5.8. *Let $\tilde{f}_k$ be the function defined in Eq. (2.36), and $\tilde{\lambda}(x^k)$ be the inexact Newton decrement defined in Eq. (2.24). 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. (2.25)]. 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,*

$$\tilde{f}_k(t) - \tilde{f}_k(0) \leq -\alpha(1+p)\left(\frac{6c-5}{4c}\right)^2 / \left(1 + \frac{6c-5}{4c}\right) \tag{2.48}$$

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 (2.48) 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 following inequality is satisfied

$$\sqrt{\epsilon} \leq \frac{6c-5}{4c}(\frac{1}{2} - p - \alpha(1+p)).$$

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

$$y \geq \theta^k - (\frac{1}{c} - 1)\frac{5}{4} \geq \frac{1}{4} - (\frac{1}{c} - 1)\frac{5}{4} = \frac{6c-5}{5c} > 0, \tag{2.49}$$

where the strict inequality follows from the fact that $c > \frac{5}{6}$. Hence the preceding two relations imply

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

Using algebraic manipulation, this yields,

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

From relation (2.49), we have $y > 0$. We can therefore multiply by y and divide by $1+y$ both sides of the above inequality to obtain

$$-\frac{1-p}{1+y}y^2 - \frac{1-\sqrt{\epsilon}}{1+y}y + y - \frac{y^2}{2(1+y)} \leq -\alpha\frac{(1+p)y^2}{1+y} \tag{2.50}$$

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. (2.50) 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 (2.45) from Lemma 2.5.7 and definition of y, the preceding relation implies

$$\tilde{f}_k(t) - \tilde{f}_k(0) \leq -\alpha(1+p)\frac{y^2}{y+1}.$$

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 (2.49) 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, Eq. (2.48). **Q.E.D.**

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 (2.29) as shown in Lemma 2.4.5. 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 $t = 1/(\tilde{\lambda}(x^k) + 1)$, then we have the following relation,

$$f(x^k + \beta t \Delta x^k) = f(\beta(x^k + t\Delta x^k) + (1-\beta)(x^k)) \leq \beta f(x^k + t\Delta x^k) + (1-\beta) f(x^k),$$

where the last inequality follows from convexity of the function f. Therefore the objective function value improvement is bounded by

$$\begin{aligned} f(x + \beta t \Delta x^k) - f(x^k) &\leq \beta f(x^k + t\Delta x^k) + (1-\beta) f(x^k) - f(x^k) \\ &= \beta(f(x^k + t\Delta x^k) - f(x^k)) \\ &= \beta(\tilde{f}_k(t) - \tilde{f}_k(0)), \end{aligned}$$

where the inequality follows from the relation above, and the last equality follows from the definition of $\tilde{f}_k(t)$. Using Lemma 2.4.5, we obtain bounds on β as $2c-1 \leq \beta \leq 1$. Hence combining this bound with Theorem 2.5.8, we obtain

$$f(x^{k+1}) - f(x^k) \leq -(2c-1)\alpha(1+p)(\frac{6c-5}{4c})^2/(1+\frac{6c-5}{4c}). \tag{2.51}$$

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].

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 2.5.9. *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. (2.23)*

and (2.24) respectively. Then the following relation holds:

$$(1-p)\tilde{\lambda}(x^k) - \sqrt{\epsilon} \leq \lambda(x^k) \leq (1+p)\tilde{\lambda}(x^k) + \sqrt{\epsilon}, \tag{2.52}$$

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

Proof. By Assumption 1, for all k, $\nabla^2 f(x^k)$ is positive definite. We therefore can apply the generalized Cauchy-Schwarz inequality and obtain

$$\begin{aligned} |(\Delta x^k)' \nabla^2 f(x^k) \Delta \tilde{x}^k| &\leq \sqrt{((\Delta x^k)' \nabla^2 f(x^k) \Delta x^k)((\Delta \tilde{x}^k)' \nabla^2 f(x^k) \Delta \tilde{x}^k)} \\ &= \lambda(x^k)\tilde{\lambda}(x^k), \end{aligned} \tag{2.53}$$

where the equality follows from definition of $\lambda(x^k)$ and $\tilde{\lambda}(x^k)$ [cf. Eqs. (2.23) and (2.24)]. Note that by Assumption 2, we have $\Delta x^k = \Delta \tilde{x}^k + \gamma$, and hence

$$\begin{aligned} |(\Delta x^k)' \nabla^2 f(x^k) \Delta \tilde{x}^k| &= |(\Delta \tilde{x}^k + \gamma)' \nabla^2 f(x^k) \Delta \tilde{x}^k| \\ &\geq (\Delta \tilde{x}^k)' \nabla^2 f(x^k) \Delta \tilde{x}^k - |\gamma' \nabla^2 f(x^k) \Delta \tilde{x}^k| \\ &\geq \tilde{\lambda}(x^k)^2 - p\tilde{\lambda}(x^k)^2 - \tilde{\lambda}(x^k)\sqrt{\epsilon}, \end{aligned} \tag{2.54}$$

where the first inequality follows from a variation of triangle inequality, and the last inequality follows from Lemma 2.5.7. Combining the two inequalities (2.53) and (2.54), we obtain

$$\lambda(x^k)\tilde{\lambda}(x^k) \geq \tilde{\lambda}(x^k)^2 - p\tilde{\lambda}(x^k)^2 - \sqrt{\epsilon}\tilde{\lambda}(x^k),$$

By canceling a nonnegative term $\tilde{\lambda}(x^k)$ on both sides, we have

$$\lambda(x^k) \geq \tilde{\lambda}(x^k) - p\tilde{\lambda}(x^k) - \sqrt{\epsilon}.$$

This shows the first half of the relation (2.52). For the second half, using the definition

of $\lambda(x^k)$, we have

$$\begin{aligned}\lambda(x^k)^2 &= (\Delta x^k)'\nabla^2 f(x^k)\Delta x^k \\ &= (\Delta\tilde{x}^k + \gamma)'\nabla^2 f(x^k)(\Delta\tilde{x}^k + \gamma) \\ &= (\Delta\tilde{x}^k)'\nabla^2 f(x^k)\Delta\tilde{x}^k + \gamma'\nabla^2 f(x^k)\gamma + 2(\Delta\tilde{x}^k)'\nabla^2 f(x^k)\gamma,\end{aligned}$$

where the second equality follows from the definition of γ [cf. Eq. (2.26)]. By using the definition of $\tilde{\lambda}(x^k)$, Assumption 2 and Lemma 2.5.6, the preceding relation implies,

$$\begin{aligned}\lambda(x^k)^2 &\leq \tilde{\lambda}(x^k)^2 + p^2\tilde{\lambda}(x^k)^2 + \epsilon + 2p\tilde{\lambda}(x^k)^2 + 2\sqrt{\epsilon}\tilde{\lambda}(x^k) \\ &\leq \tilde{\lambda}(x^k)^2 + p^2\tilde{\lambda}(x^k)^2 + 2p\tilde{\lambda}(x^k)^2 + 2\sqrt{\epsilon}(1+p)\tilde{\lambda}(x^k) + \epsilon \\ &= ((1+p)\tilde{\lambda}(x^k) + \sqrt{\epsilon})^2,\end{aligned}$$

where the second inequality follows by adding a nonnegative term of $2\sqrt{\epsilon}p\tilde{\lambda}(x^k)$ to the right hand side. By nonnegativity of p, ϵ, λ and $\tilde{\lambda}(x^k)$, we can take the square root of both sides and this completes the proof for relation (2.52). **Q.E.D.**

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,*

$$(1+p)(\theta^k + \tau) + \sqrt{\epsilon} \leq \phi \tag{2.55}$$

$$p + \sqrt{\epsilon} \leq 1 - (4\phi^2)^{\frac{1}{4}} - \phi, \tag{2.56}$$

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 (2.56) can be satisfied by some positive scalars p and ϵ. Relation (2.55) will be used to guarantee the condition $\lambda(x^k) \leq 0.68$ is satisfied, so that we can use Lemma 2.5.5 to relate the suboptimality bound with the Newton decrement. Relation (2.56) will be used for

establishing the quadratic rate of convergence of the objective function value, as we will show in Theorem 2.5.11.

By Assumption 4, we have $\tilde{\lambda}(x^k) = \theta^k + \tau^k \leq \theta^k + \tau$. Therefore, by relation (2.52), we have

$$\lambda(x^k) \leq (1+p)\tilde{\lambda}(x^k) + \sqrt{\epsilon} \leq \phi \leq 0.267. \tag{2.57}$$

Thus the condition $\lambda(x^k) \leq 0.68$ for Lemma 2.5.5 is satisfied. We can therefore apply relation (2.43) to bound suboptimality in our algorithm, i.e., $f(x^k) - f^*$, 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 f^*, we will also characterize explicitly the size of the neighborhood. We will need the following lemma, which relates the exact Newton decrement at the current and the next step.

Lemma 2.5.10. *Let x^k be the iterates generated by the inexact Newton method [cf. Section 2.4]. Let λ and $\tilde{\lambda}$ be the exact and inexact Newton decrement defined in Eqs. (2.23) and (2.24) 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*

$$\lambda(x^{k+1}) \leq v\lambda(x^k)^2 + \xi, \tag{2.58}$$

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.

Proof. Due to the fact that $\theta^k < \frac{1}{4}$, by stepsize rule in Eq. (2.25) we have stepsize $s^k = 1$, and therefore $x^{k+1} = x^k + \Delta\tilde{x}$. Let Δx^{k+1} denote the exact Newton direction

at primal solution x^{k+1}, then in view of Lemma 2.5.4 by letting $z = \Delta x^{k+1}$, we have

$$\begin{aligned}\lambda(x^{k+1})^2 &= (\Delta x^{k+1})'\nabla f^2(x+\Delta\tilde{x})\Delta x^{k+1} \\ &\leq \frac{\tilde{\lambda}(x^k)^2}{1-\tilde{\lambda}(x^k)}\sqrt{(\Delta x^{k+1})'\nabla^2 f(x)\Delta x^{k+1}} + \left|\gamma'\nabla^2 f(x)'\Delta x^{k+1}\right| \\ &\leq \frac{\tilde{\lambda}(x^k)^2}{1-\tilde{\lambda}(x^k)}\sqrt{(\Delta x^{k+1})'\nabla^2 f(x)\Delta x^{k+1}} + \sqrt{\gamma'\nabla^2 f(x)\gamma}\sqrt{(\Delta x^{k+1})'\nabla^2 f(x)\Delta x^{k+1}},\end{aligned}$$

where the last inequality follows from the generalized Cauchy-Schwarz inequality. Using Assumption 2, the above relation implies,

$$\lambda(x^{k+1})^2 \leq \left(\frac{\tilde{\lambda}(x^k)^2}{1-\tilde{\lambda}(x^k)} + \sqrt{p^2\tilde{\lambda}(x^k)^2+\epsilon}\right)\sqrt{(\Delta x^{k+1})'\nabla^2 f(x)\Delta x^{k+1}}.$$

By the fact that $\tilde{\lambda}(x^k) \leq \theta^k + \tau \leq \phi < 1$, we can apply Lemma 2.5.3 and obtain,

$$\begin{aligned}\lambda(x^{k+1})^2 &\leq \frac{1}{1-\tilde{\lambda}(x^k)}\left(\frac{\tilde{\lambda}(x^k)^2}{1-\tilde{\lambda}(x^k)} + \sqrt{p^2\tilde{\lambda}(x^k)^2+\epsilon}\right)\sqrt{(\Delta x^{k+1})'\nabla^2 f(x+\Delta\tilde{x})\Delta x^{k+1}} \\ &= \left(\frac{\tilde{\lambda}(x^k)^2}{(1-\tilde{\lambda}(x^k))^2} + \frac{\sqrt{p^2\tilde{\lambda}(x^k)^2+\epsilon}}{1-\tilde{\lambda}(x^k)}\right)\lambda(x^{k+1}).\end{aligned}$$

By dividing the last line by $\lambda(x^{k+1})$, we obtain

$$\lambda(x^{k+1}) \leq \frac{\tilde{\lambda}(x^k)^2}{(1-\tilde{\lambda}(x^k))^2} + \frac{\sqrt{p^2\tilde{\lambda}(x^k)^2+\epsilon}}{1-\tilde{\lambda}(x^k)} \leq \frac{\tilde{\lambda}(x^k)^2}{(1-\tilde{\lambda}(x^k))^2} + \frac{p\tilde{\lambda}(x^k)+\sqrt{\epsilon}}{1-\tilde{\lambda}(x^k)}.$$

From Eq. (2.52), we have $\tilde{\lambda}(x^k) \leq \frac{\lambda(x^k)+\sqrt{\epsilon}}{1-p}$. Therefore the above relation implies,

$$\lambda(x^{k+1}) \leq \left(\frac{\lambda(x^k)+\sqrt{\epsilon}}{1-p-\lambda(x^k)-\sqrt{\epsilon}}\right)^2 + \frac{p\lambda(x^k)+\sqrt{\epsilon}}{1-p-\lambda(x^k)-\sqrt{\epsilon}}.$$

By relation (2.57), we have $\lambda(x^k) \leq \phi$, the above relation can be relaxed to be

$$\lambda(x^{k+1}) \leq \left(\frac{\lambda(x^k)}{1-p-\phi-\sqrt{\epsilon}}\right)^2 + \frac{\phi p+\sqrt{\epsilon}}{1-p-\phi-\sqrt{\epsilon}} + \frac{2\phi\sqrt{\epsilon}+\epsilon}{(1-p-\phi-\sqrt{\epsilon})^2}$$

Hence, by definition of ξ and v, we have

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

Q.E.D.

The above lemma can then 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 2.5.11. *Let λ and $\tilde{\lambda}$ be the exact and inexact Newton decrement defined in Eqs. (2.23) and (2.24) respectively. Let $f(x^k)$ be the objective function value at k^{th} iteration for the algorithm defined in Section 2.4 and f^* be the optimal objective function value for problem (2.4). 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 $m > 0$,

$$\lambda(x^{k+m}) \leq \frac{1}{2^{2^m}v} + \xi + \frac{\delta}{v}\frac{2^{2^m-1}-1}{2^{2^m}}, \tag{2.59}$$

and

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

Proof. We prove relation (2.59) by induction. First for $m = 1$, by relation (2.56), 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 (2.58) and (2.57), 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 (2.59) for $m = 1$.

We next assume that relation (2.59) holds for some some $m > 0$, and show that it also holds for $m + 1$. By relation (2.58), we have

$$\begin{aligned}
\lambda(x^{k+m+1}) &\leq v\lambda(x^{k+m})^2 + \xi \\
&\leq v\left(\frac{1}{2^{2^m}v} + \xi + \frac{\delta}{v}\frac{2^{2^m-1}-1}{2^{2^m}}\right)^2 + \xi \\
&= \frac{1}{2^{2^{m+1}}v} + \frac{\xi}{2^{2^m-1}} + \frac{\delta}{v}\frac{2^{2^m-1}-1}{2^{2^{m+1}-1}} + v\left(\xi + \frac{\delta}{v}\frac{2^{2^m-1}-1}{2^{2^m}}\right)^2 + \xi,
\end{aligned}$$

where the second inequality follows by the assumption that relation (2.59) holds for m. Using algebraic manipulations and the assumption that $\xi + v\xi \leq \frac{\delta}{4v}$, this yields

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

completing the induction and therefore the proof of relation (2.59).

Using relation (2.57), we have $\lambda(x^k) \leq \phi \leq 0.68$, we can therefore apply Lemma 2.5.5, and obtain an upper bound on suboptimality as follows,

$$f(x^{k+m}) - f^* \leq \left(\lambda(x^{k+m})\right)^2 \leq \lambda(x^{k+m}).$$

Combine this with relation (2.59), we obtain

$$f(x^{k+m}) - f^* \leq \frac{1}{2^{2^m}v} + \xi + \frac{\delta}{v}\frac{2^{2^m-1}-1}{2^{2^m}}.$$

Taking limit superior on both sides of the preceding relation establishes the final result. **Q.E.D.**

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 $\xi = \frac{\phi p + \sqrt{\epsilon}}{1-p-\phi-\sqrt{\epsilon}} + \frac{2\phi\sqrt{\epsilon}+\epsilon}{(1-p-\phi-\sqrt{\epsilon})^2}$, $v = \frac{1}{(1-p-\phi-\sqrt{\epsilon})^2}$, and 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.

2.5.3 Convergence with respect to Design Parameter μ

So far, we restricted attention to develop an algorithm for a fixed logarithm barrier coefficient μ. We next study the convergence property of the optimal object function value as a function of μ. We utilize the following result from [34].

Lemma 2.5.12. *Let G be a closed convex domain, and function g be a self-concordant barrier function for G, then for any x, $y \in$ int G, we have $(y-x)'\nabla g(x) \leq 1$.*

Using this lemma, we can establish the following result, which bounds the sub-optimality as a function of μ.

Theorem 2.5.13. *For the constrained optimization problem (2.4) and a given μ, denote the optimal solution as $x(\mu)$, and $h(\mu) = \sum_{i=1}^{S} -U_i(x_i(\mu))$. Similarly, denote the optimal solution for the inequality constrained problem (2.2) together with corresponding slack variables as defined in Eq. (2.3) as x^*, and $h^* = \sum_{i=1}^{S} -U_i(x_i^*)$. Then the following relation holds,*

$$h(\mu) - h^* \leq \mu.$$

Proof. For notational simplicity, we write $h(x) = \sum_{i=1}^{S} -U_i(x_i(\mu))$, and $g(x) = -\sum_{i=1}^{S+L} \log(x_i)$. Therefore the objective function for problem (2.4) can be written as $h(x) + \mu g(x)$.

By Assumption 1, we have that the utility functions are concave, therefore the negative objective functions in the minimization problems are convex. From convexity,

we obtain

$$h(x^*) \geq h(x(\mu)) + (x^* - x(\mu))'\nabla h(x). \tag{2.60}$$

By optimality condition for $x(\mu)$, we have,

$$(\nabla h(x(\mu)) + \mu \nabla g(x(\mu)))'(x - x(\mu)) \geq 0,$$

for any feasible x. Since x^* is feasible, we have

$$(\nabla h(x(\mu)) + \mu \nabla g(x(\mu)))'(x^* - x(\mu)) \geq 0,$$

which implies

$$\nabla h(x(\mu))'(x^* - x(\mu)) \geq -\mu \nabla g(x(\mu))'(x^* - x(\mu)).$$

For any μ, we have $x(\mu)$ belong to the interior of the feasible set, and by Lemma 2.5.12, we have for all $\tilde{\mu}$, $g(x(\mu))'(x(\tilde{\mu}) - x(\mu)) \leq 1$. By continuity of $x(\mu)$ and closedness of the convex set $Ax \leq c$, for A and c defined in problem (2.4), we have $x^* = \lim_{\mu \to 0} x(\mu)$, and hence

$$g(x(\mu))'(x^* - x(\mu)) = \lim_{\tilde{\mu} \to 0} g(x(\mu))'(x(\tilde{\mu}) - x(\mu)) \leq 1.$$

The preceding two relations imply

$$\nabla h(x(\mu))'(x^* - x(\mu)) \geq -\mu.$$

In view of relation (2.60), this establishes the desired result, i.e.,

$$h(\mu) - h^* \leq \mu.$$

Q.E.D.

This theorem suggests that as the barrier function coefficient μ approaches 1,

the sub-optimality between the objective function value our algorithm provides for problem (2.4) and the exact optimal objective function value for problem (2.2) can be bounded by 1, i.e., $h(x(1)) - h^* \leq 1$ following the notation in the preceding theorem. The effect of this constant 1 vanishes when we scale the original problem objective function by a large constant, implying the solution of the two problems coincide.

2.6 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 what follows, when we refer to the number of iterations of our Newton method, we are referring to the number iterations for the dual variable computation procedure, i.e., the inner loop. In the simulation results, we compare our distributed Newton method performance against both distributed subgradient method [30] and also the Newton-type diagonal scaling dual method developed in [6].

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 the mean of 40, and number of sources S is a random varible with the mean of 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 2-3 and Figure 2-4. Figure Figure 2-3 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 2-4 shows the histogram for the same set of data. This figure shows on average our method is about 100 times faster for these relatively small networks, for larger networks the performance is even more dramatic. Diagonal scaling method has performance on the

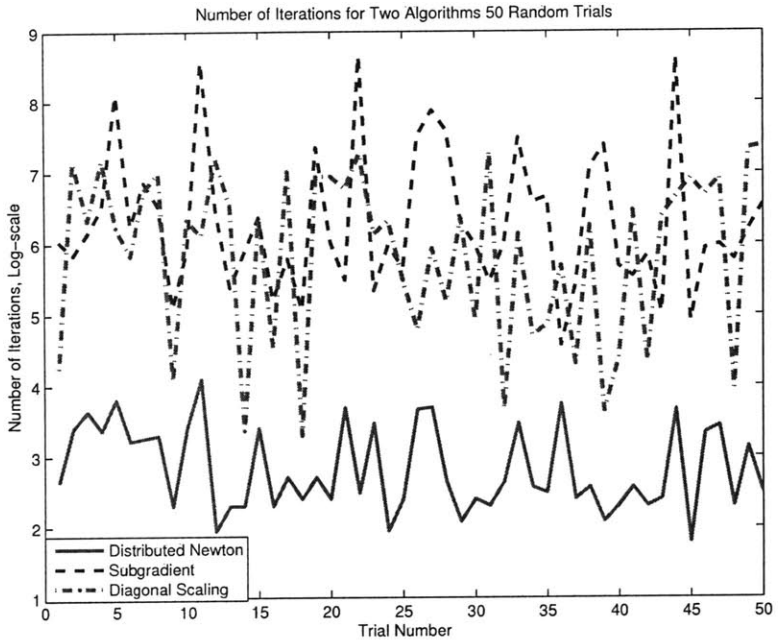

Figure 2-3: Log scaled iteration count for the 3 methods implemented over 50 randomly generated networks .

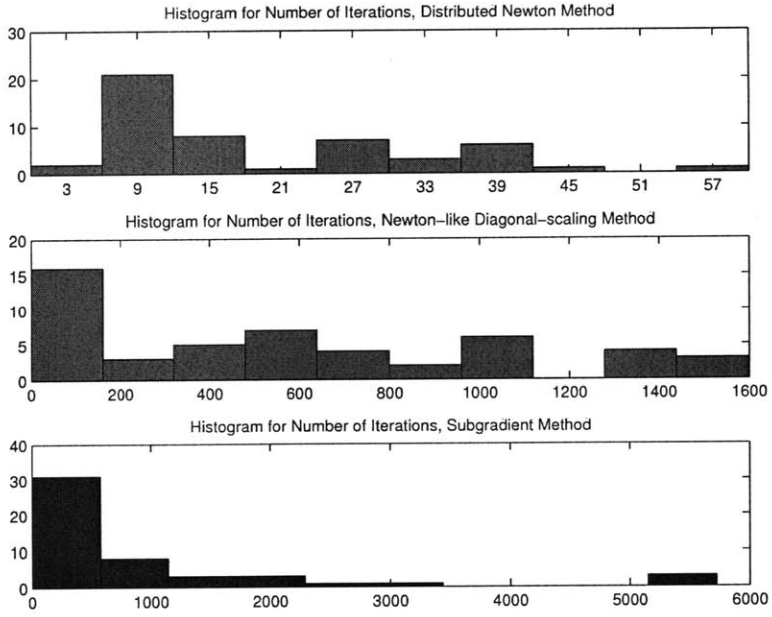

Figure 2-4: Histogram of iteration counts for the 3 methods implemented over 50 randomly generated networks

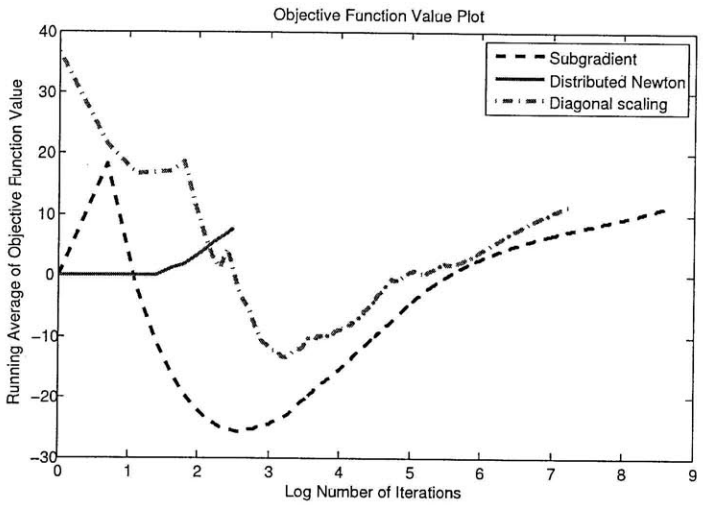

Figure 2-5: One sample simulation of the 3 methods, running average of objective function value after each iteration against log scaled iteration count

same order of magnitude as the subgradient method, but slightly faster.

A typical output for running average of objective function value after each iteration against log scaled iteration count is presented in Figure 2-5. The results shows newly developed method exhibits significant advantage over the traditional first order ones. In this particular example, our distributed Newton algorithm terminated after 12 iterations with a objective function value of 15.27, the Newton-like diagonal scaling method took 1435 iterations with a objective function value of 15.77, and the subgradient method finished after 5725 iterations with a objective function value of 15.43.

2.7 Conclusions

This chapter develops a distributed Newton-type second order algorithm for network utility maximization problems that can achieve the superlinear convergence rate within some error neighborhood. We show that the computation of the dual Newton

step can be implemented in a decentralized manner using matrix splitting technique. 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 for network utility maximization problems. Possible future directions include to analyze the relationship between the rate of converge and the underlying topology and analyze the rate of convergence for the consensus-based schemes when computing the errors.

Chapter 3

A Distributed Newton Method for Network Flow Cost Minimization Problems

3.1 Introduction

In this chapter, we propose an alternative approach based on using Newton-type (or second-order) methods for network flow cost minimization problems. We show that the proposed method can be implemented in a distributed manner and has faster convergence properties.

Consider a network represented by a directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$. Each edge e in the network has a convex cost function $\phi_e(x^e)$, which captures the cost due to congestion effects as a function of the flow x^e on this edge. The total cost of a flow vector $x = [x^e]_{e\in\mathcal{E}}$ is given by the sum of the edge costs, i.e., $\sum_{e\in\mathcal{E}} \phi_e(x^e)$. Given an external supply b_i for each node $i \in \mathcal{N}$, the *network flow cost minimization problem* is to find a minimum cost flow allocation vector that satisfies the flow conservation constraint at each node.[1]

This problem can be formulated as a convex optimization problem with linear

[1]We focus on feasible problems, i.e., we assume that the total in-flow to the network is equal to the total out-flow, $\sum_{i\in\mathcal{N}} b_i = 0$.

equality constraints. The application of dual decomposition together with a dual subgradient algorithm then yields a distributed iterative solution method. Instead, we propose a distributed primal-dual Newton-type method that achieves a superlinear convergence rate (to an error neighborhood).

The challenges in using Newton-type methods in this context are twofold. First, the superlinear convergence rate of this type of methods is achieved by using a backtracking stepsize rule, which relies on global information in the computation of the norm of a residual function (used in defining the stepsize). We solve this problem by using a consensus-based local averaging scheme for estimating the norm of the residual function. Second, the computation of the dual Newton step involves a matrix inversion, which requires global information. Our main contribution in this regard is to develop a distributed iterative scheme for the computation of the dual Newton step. The key idea is to recognize that the dual Newton step can be solved via an iterative scheme, which involves the Laplacian of the graph, and therefore can be solved using a consensus-based scheme in a distributed manner. We show that the convergence rate of this scheme is governed by the spectral properties of the underlying graph. Hence, for fast-mixing graphs (i.e., those with large spectral gaps), the dual Newton step can be computed efficiently using only local information.

Since our method uses consensus-based schemes to compute the stepsize and the Newton direction in each iteration, exact computation is not feasible. Another major contribution of this thesis is to consider truncated versions of these consensus-schemes at each iteration and present convergence rate analysis of the constrained Newton method when the stepsize and the direction are estimated with some error. We show that when these errors are sufficiently small, the value of the residual function converges superlinearly to a neighborhood of the origin, whose size is explicitly quantified as a function of the errors and the parameters of the objective function and the constraints of the network flow cost minimization problem.

The rest of the chapter is organized as follows: Section 3.2 defines the network flow cost minimization problem and shows that the dual decomposition and subgradient method can be implemented in a distributed manner. This section also presents

the constrained primal-dual Newton method for this problem and introduces a distributed iterative scheme for computing the dual Newton step. Section 3.3 presents a convergence rate analysis for an inexact Newton method, for which there are errors associated with computation of the step and the stepsize. In section 3.4, simulations demonstrate that the Newton's method outperforms the subgradient method with respect to runtime. Section 3.5 contains our concluding remarks.

Basic Notation and Notions:

A vector is viewed as a column vector, unless clearly stated otherwise. 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} or $[A]_{ij}$ to denote the matrix entry in the i-th row and j-th column. We write x' to denote the transpose of a vector x. The scalar product of two vectors $x, y \in \mathbb{R}^m$ is denoted by $x'y$. We use $\|x\|$ to denote the standard Euclidean norm, $\|x\| = \sqrt{x'x}$. For a vector-valued function $f : \mathbb{R}^n \to \mathbb{R}^m$, the gradient matrix of f at $x \in \mathbb{R}^n$ is denoted by $\nabla f(x)$.

A vector $a \in \mathbb{R}^m$ is said to be a *stochastic vector* when its components a_i, $i = 1, \ldots, m$, are nonnegative and their sum is equal to 1, i.e., $\sum_{i=1}^m a_i = 1$. A square $m \times m$ matrix A is said to be a *stochastic matrix* when each row of A is a stochastic vector. A stochastic matrix is called irreducible and aperiodic (also known as primitive) if all eigenvalues (except the trivial eigenvalue at 1) are subunit.

One can associate a discrete-time Markov chain with a stochastic matrix and a graph $\mathcal{G}$ as follows: The state of the chain at time $k \in \{1, 2, \cdots\}$, denoted by $X(k)$, is a node in $\mathcal{N}$ (the node set of the graph) and the weight associated to each edge in the graph is the probability with which X makes a transition between two adjacent nodes. In other words, the transition from state i to state j happens with probability p_{ij}, the weight of edge (i, j). If $\pi(k)$ with elements defined as $\pi_i(k) = \mathbb{P}(X(k) = i)$ is the probability distribution of the state at time k, the state distribution satisfies the recursion $\pi(k+1)^T = \pi(k)^T P$. If the chain is irreducible and aperiodic then for all initial distributions, π converges to the unique stationary distribution π^* [7].

3.2 Network Flow Cost Minimization Problem

We consider a network represented by a directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ with node set $\mathcal{N} = \{1, \ldots, N\}$, and edge set $\mathcal{E} = \{1, \ldots, E\}$. We denote the flow vector by $x = [x^e]_{e \in \mathcal{E}}$, where x^e denotes the flow on edge e. The flow conservation conditions at the nodes can be compactly expressed as

$$Ax = b,$$

where A is the $N \times E$ *node-edge incidence matrix* of the graph, i.e.,

$$A_{ij} = \begin{cases} 1 & \text{if edge } j \text{ leaves node } i \\ -1 & \text{if edge } j \text{ enters node } i \\ 0 & \text{otherwise,} \end{cases}$$

and the vector b denotes the external sources, i.e., $b_i > 0$ (or $b_i < 0$) indicates b_i units of external flow enters (or exits) node i.

We associate a cost function $\phi_e : \mathbb{R} \to \mathbb{R}$ with each edge e, i.e., $\phi_e(x^e)$ denotes the cost on edge e as a function of the edge flow x^e. We assume that the cost functions ϕ_e are strictly convex and twice continuously differentiable.

The network flow cost miminzation problem can be written as

$$\begin{array}{ll} \text{minimize} & \displaystyle\sum_{e=1}^{E} \phi_e(x^e) \\ \text{subject to} & Ax = b. \end{array} \tag{3.1}$$

In this chapter, our goal is to investigate *iterative distributed methods* for solving problem (3.1). In particular, we focus on two methods: first relies on solving the dual of problem (3.1) using a subgradient method; second uses a constrained Newton method, where, at each iteration, the Newton direction is computed iteratively using an averaging method.

3.2.1 Dual Subgradient Method

We first consider solving problem (3.1) using a dual subgradient method. To define the dual problem, we form the Lagrangian function of problem (3.1) $\mathcal{L} : \mathbb{R}^E \times \mathbb{R}^N \to \mathbb{R}$ given by

$$\mathcal{L}(x, \lambda) = \sum_{e=1}^{E} \phi_e(x^e) - \lambda'(Ax - b).$$

The dual function $q(\lambda)$ is then given by

$$\begin{aligned} q(\lambda) &= \inf_{x \in \mathbb{R}^E} \mathcal{L}(x, \lambda) \\ &= \inf_{x \in \mathbb{R}^E} \left(\sum_{e=1}^{E} \phi_e(x^e) - \lambda' A x \right) + \lambda' b \\ &= \sum_{e=1}^{E} \inf_{x^e \in \mathbb{R}} \left(\phi_e(x^e) - (\lambda' A)^e x^e \right) + \lambda' b. \end{aligned}$$

Hence, in view of the fact that the objective function and the constraints of problem (3.1) are separable in the decision variables x^e, the evaluation of the dual function decomposes into one-dimensional optimization problems. We assume that each of these optimization problems has an optimal solution, which is unique by the strict convexity of the functions ϕ_e and is denoted by $x^e(\lambda)$. Using the first order optimality conditions, it can be seen that for each e, $x^e(\lambda)$ is given by

$$x^e(\lambda) = (\phi_e')^{-1}(\lambda^i - \lambda^j), \tag{3.2}$$

where $i, j \in \mathcal{N}$ denote the end nodes of edge e. Thus, for each edge e, the evaluation of $x^e(\lambda)$ can be done based on local information about the edge cost function ϕ^e and the dual variables of the incident nodes i and j.

We can write the dual problem as

$$\text{maximize}_{\lambda \in \mathbb{R}^N} \; q(\lambda).$$

The dual problem can be solved by using a subgradient method: given an initial

vector λ_0, the iterates are generated by

$$\lambda_{k+1} = \lambda_k - \alpha_k g_k \qquad \text{for all } k \geq 0,$$

where g_k is a subgradient of the dual function $q(\lambda)$ at $\lambda = \lambda_k$ given by

$$g_k = Ax(\lambda_k) - b, \qquad x(\lambda_k) = \operatorname*{argmin}_{x \in \mathbb{R}^E} \mathcal{L}(x, \lambda_k),$$

i.e., for all $e \in \mathcal{E}$, $x^e(\lambda_k)$ is given by Eq. (3.2) with $\lambda = \lambda_k$.

This method naturally lends itself to a distributed implementation: each node i updates its dual variable λ^i using local (subgradient) information g^i obtained from edges e incident to that node, which in turn updates its primal variables $x^e(\lambda)$ using dual variables of the incident nodes. Despite its simplicity and distributed nature, however, it is well-known that the dual subgradient method suffers from slow rate of convergence (see [32] and [33] for rate analysis and construction of primal solutions for dual subgradient methods), which motivates us to consider a Newton method for solving problem (3.1).

3.2.2 Equality-Constrained Newton Method

We next consider solving problem (3.1) using an (infeasible start) equality-constrained Newton method (see [13], Chapter 10). We let $f(x) = \sum_{e=1}^{E} \phi_e(x^e)$ for notational simplicity. Given an initial primal vector x_0, the iterates are generated by

$$x_{k+1} = x_k + \alpha_k v_k$$

where v_k is the Newton step given as the solution to the following system of linear equations:[2]

$$\begin{pmatrix} \nabla^2 f(x_k) & A' \\ A & 0 \end{pmatrix} \begin{pmatrix} v_k \\ w_k \end{pmatrix} = - \begin{pmatrix} \nabla f(x_k) \\ Ax_k - b \end{pmatrix}.$$

[2]This is essentially a primal-dual method with the vectors v_k and w_k acting as primal and dual steps; see Section 3.3.

We let $H_k = \nabla^2 f(x_k)$ and $h_k = Ax_k - b$ for notational convenience. Solving for v_k and w_k in the preceding yields

$$v_k = -H_k^{-1}(\nabla f(x_k) + A'w_k), \text{ and}$$

$$(AH_k^{-1}A')w_k = h_k - AH_k^{-1}\nabla f(x_k). \tag{3.3}$$

Since the matrix H_k^{-1} is a diagonal matrix with entries $[H_k^{-1}]_{ee} = (\frac{\partial^2 \phi_e}{(\partial x^e)^2})^{-1}$, given the vector w_k, the Newton step v_k can be computed using local information. However, the computation of the vector w_k at a given primal vector x_k cannot be implemented in a decentralized manner in view of the fact that solving equation (3.3) $(AH_k^{-1}A')^{-1}$ requires global information. The following section provides an iterative scheme to compute the vector w_k using local information.

Distributed Computation of the Newton Direction

Consider the vector w_k defined in Eq. (3.3). The key step in developing a decentralized iterative scheme for the computation of the vector w_k is to recognize that the matrix $AH_k^{-1}A'$ is the weighted Laplacian of the underlying graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, denoted by L_k. Hence, L_k can be written as

$$L_k = AH_k^{-1}A' = D_k - B_k.$$

Here B_k is an $N \times N$ matrix with entries

$$(B_k)_{ij} = \begin{cases} \left(\frac{\partial^2 \phi_e}{(\partial x^e)^2}\right)^{-1} & \text{if } e = (i,j) \in \mathcal{E}, \\ 0 & \text{otherwise,} \end{cases}$$

and D_k is an $N \times N$ diagonal matrix with entries

$$(D_k)_{ii} = \sum_{j \in \mathcal{N}_i} (B_k)_{ij},$$

where $\mathcal{N}_i$ denotes the set of neighbors of node i, i.e., $\mathcal{N}_i = \{j \in \mathcal{N} \mid (i,j) \in \mathcal{E}\}$. Letting $s_k = h_k - AH_k^{-1}\nabla f(x_k)$ for notational convenience, Eq. (3.3) can be then rewritten as

$$(I - (D_k + I)^{-1}(B_k + I))w_k = (D_k + I)^{-1}s_k.$$

This motivates the following iterative scheme (known as splitting) to solve for w_k. For any $t \geq 0$, the iterates are generated by

$$w(t+1) = (D_k + I)^{-1}(B_k + I)w(t) + (D_k + I)^{-1}s. \tag{3.4}$$

The i^{th} row of both matrices B_k and D_k can be computed at the node i using the local neighborhood information, therefore the above iteration shows that the dual Newton step can be computed in a decentralized fashion. Since our distributed solution involves an iterative scheme, exact computation is not feasible. In what follows, we show that the Newton method has desirable convergence properties even when the Newton direction is computed with some error provided that the errors are sufficiently small.

3.3 Inexact Newton Method

In this section, we consider the following convex optimization problem with equality constraints:

$$\begin{aligned} \text{minimize} \quad & f(x) \\ \text{subject to} \quad & Ax = b, \end{aligned} \tag{3.5}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a twice continuously differentiable convex function, and A is an $m \times n$ matrix. The network flow cost minimization problem (3.1) is a special case of this problem with $f(x) = \sum_{e=1}^{E} \phi_e(x^e)$ and A is the $N \times E$ node-edge incidence matrix. We denote the optimal value of this problem by f^*. Throughout this section, we assume that the value f^* is finite and problem (3.5) has an optimal solution, which

we denote by x^*.

We consider an *inexact (infeasible start) Newton method* for solving problem (3.5) (see [13]). In particular, we let $y = (x, \nu) \in \mathbb{R}^n \times \mathbb{R}^m$, where x is the primal variable and ν is the dual variable, and study a primal-dual method which updates the vector y at iteration k as follows:

$$y_{k+1} = y_k + \alpha_k d_k, \tag{3.6}$$

where α_k is a positive stepsize, and the vector d_k is an *approximate constrained Newton direction* given by

$$Dr(y_k)d_k = -r(y_k) + \epsilon_k. \tag{3.7}$$

Here, the residual function $r : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n \times \mathbb{R}^m$ is defined as

$$r(x, \nu) = (r_{dual}(x, \nu), r_{pri}(x, \nu)), \tag{3.8}$$

where

$$r_{dual}(x, \nu) = \nabla f(x) + A'\nu, \tag{3.9}$$

and

$$r_{pri}(x, \nu)) = Ax - b. \tag{3.10}$$

Moreover, $Dr(y) \in \mathbb{R}^{(n+m)\times(n+m)}$ is the gradient matrix of r evaluated at y, and the vector ϵ_k is an error vector at iteration k. We assume that the error sequence $\{\epsilon_k\}$ is uniformly bounded from above, i.e., there exists a scalar $\epsilon \geq 0$ such that

$$\|\epsilon_k\| \leq \epsilon \qquad \text{for all } k \geq 0. \tag{3.11}$$

We adopt the following standard assumption:

Assumption 5. Let $r : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n \times \mathbb{R}^m$ be the residual function defined in Eqs. (3.8)-(3.10). Then, we have:

(a) (*Lipschitz Condition*) There exists some constant $L > 0$ such that

$$\|Dr(y) - Dr(\bar{y})\| \leq L\|y - \bar{y}\| \, \forall y, \bar{y} \in \mathbb{R}^n \times \mathbb{R}^m.$$

(b) There exists some constant $M > 0$ such that

$$\|Dr(y)^{-1}\| \leq M \qquad \forall y \in \mathbb{R}^n \times \mathbb{R}^m.$$

3.3.1 Basic Relation

We use the norm of the residual vector $\|r(y)\|$ to measure the progress of the algorithm. In the next proposition, we present a relation between the iterates $\|r(y_k)\|$, which holds for any stepsize rule. The proposition follows from a multi-dimensional extension of the descent lemma (see [9]).

Proposition 1. Let Assumption 5 hold. Let $\{y_k\}$ be a sequence generated by the method (3.6). For any stepsize rule α_k, we have

$$\begin{aligned}\|r(y_{k+1})\| \leq \; & (1-\alpha_k)\|r(y_k)\| + M^2 L \alpha_k^2 \|r(y_k)\|^2 \\ & + \alpha_k \|\epsilon_k\| + M^2 L \alpha_k^2 \|\epsilon_k\|^2.\end{aligned}$$

Proof. We consider two vectors $w \in \mathbb{R}^n \times \mathbb{R}^m$ and $z \in \mathbb{R}^n \times \mathbb{R}^m$. We let ξ be a scalar parameter and define the function $g(\xi) = r(w + \xi z)$. From the chain rule, it follows that $\nabla g(\xi) = Dr(w + \xi z)z$. Using the Lipschitz continuity of the residual function gradient [cf. Assumption 5(a)], we obtain:

$$
\begin{aligned}
r(w+z) - r(w) = g(1) - g(0) & \\
&= \int_0^1 \nabla g(\xi) d\xi \\
&= \int_0^1 Dr(w+\xi z) z d\xi \\
&\leq \left| \int_0^1 (Dr(w+\xi z) - Dr(w)) z d\xi \right| + \int_0^1 Dr(w) z d\xi \\
&\leq \int_0^1 \|Dr(w+\xi z) - Dr(w)\| \, \|z\| d\xi + Dr(w) z \\
&\leq \|z\| \int_0^1 L\xi \|z\| d\xi + Dr(w) z \\
&= \frac{L}{2} \|z\|^2 + Dr(w) z.
\end{aligned}
$$

We apply the preceding relation with $w = y_k$ and $z = \alpha_k d_k$ and obtain

$$
r(y_k + \alpha_k d_k) - r(y_k) \leq \alpha_k Dr(y_k) d_k + \frac{L}{2} \alpha_k^2 \|d_k\|^2.
$$

By Eq. (3.7), we have $Dr(y_k) d_k = -r(y_k) + \epsilon_k$. Substituting this in the previous relation, this yields

$$
r(y_k + \alpha_k d_k) \leq (1 - \alpha_k) r(y_k) + \alpha_k \epsilon_k + \frac{L}{2} \alpha_k^2 \|d_k\|^2.
$$

Moreover, using Assumption 5(b), we have

$$
\begin{aligned}
\|d_k\|^2 &= \|Dr(y_k)^{-1}(-r(y_k) + \epsilon_k)\|^2 \\
&\leq \|Dr(y_k)^{-1}\|^2 \, \| - r(y_k) + \epsilon_k\|^2 \\
&\leq M^2 \Big(2\|r(y_k)\|^2 + 2\|\epsilon_k\|^2 \Big).
\end{aligned}
$$

Combining the above relations, we obtain $\|r(y_{k+1})\| \leq (1-\alpha_k)\|r(y_k)\| + M^2 L \alpha_k^2 \|r(y_k)\|^2 + \alpha_k \|\epsilon_k\| + M^2 L \alpha_k^2 \|\epsilon_k\|^2$, establishing the desired relation. **Q.E.D.**

3.3.2 Inexact Backtracking Stepsize Rule

We use a backtracking stepsize rule in our method to achieve the superlinear local convergence properties of the Newton method. However, this requires computation of the norm of the residual function $\|r(y)\|$. In view of the distributed nature of the residual vector $r(y)$ [cf. Eq. (3.8)], this norm can be computed using a distributed consensus-based scheme. Since this scheme is iterative, in practice the residual norm can only be estimated with some error.

Let $\{y_k\}$ be a sequence generated by the inexact Newton method (3.6). At each iteration k, we assume that we can compute the norm $\|r(y_k)\|$ with some error, i.e., we can compute a scalar $n_k \geq 0$ that satisfies

$$\Big| n_k - \|r(y_k)\| \Big| \leq \gamma/2, \tag{3.12}$$

for some constant $\gamma \geq 0$. Hence, n_k is an approximate version of $\|r(y_k)\|$, which can be computed for example using distributed iterative methods. For fixed scalars $\sigma \in (0, 1/2)$ and $\beta \in (0, 1)$, we set the stepsize α_k equal to $\alpha_k = \beta^{m_k}$, where m_k is the smallest nonnegative integer that satisfies

$$n_{k+1} \leq (1 - \sigma\beta^m)n_k + B + \gamma. \tag{3.13}$$

Here, γ is the maximum error in the residual function norm [cf. Eq. (3.12)], and B is a constant given by

$$B = \epsilon + M^2 L \epsilon^2, \tag{3.14}$$

where ϵ is the upper bound on the error sequence in the constrained Newton direction [cf. Eq. (3.11)] and M and L are the constants in Assumption 5.

Convergence Properties for Consensus Algorithms

In this section, we analyze the rate of convergence for the consensus algorithm, because it is used to calculate the norm of the residual function, $\|r(y)\|$. One possible way to implement the consensus scheme is to have each node simply iteratively taking

average over their local neighborhoods including the value itself has (for more details on consensus, we refer the readers to [22], [24], [36], [41], [42], [2], [1]). Let $z(0)$ denote the vector of initial values the nodes have, then the consensus update rule can be written as

$$z(t+1) = Pz(t) = P^{t+1}z(0),$$

where the matrix P is nonnegative and row stochastic, with positive diagonal elements. The same iteration can be also obtained as a result of random walk Markov chain process over a graph, where each node has a self-arc. Hence the time it takes to reach consensus coincides with the mixing time for the Markov chain with transition matrix P. This Markov chain is known to be a reversible one [12], therefore we resort to Markov chain theories to characterize the speed of convergence.

It is well known that the rate of convergence to the stationary distribution of a Markov chain is governed by the second largest eigenvalue magnitude of matrix P defined as $\mu(P) = \max_{i=2,\cdots,n}\{|\lambda_i(P)|\}$ [19]. To make this statement more precise, let i be the initial state and define the *total variation distance* between the distribution at time k and the stationary distribution π^* as

$$\Delta_i(k) = \frac{1}{2}\sum_{j\in\mathcal{V}} |P_{ij}^k - \pi_j^*|.$$

The rate of convergence to the stationary distribution is measured using the following quantity known as the *mixing time*:

$$T_{mix} = \max_i \min\{k : \Delta_i(k') < e^{-1} \text{ for all } k' \geq k\}.$$

The following theorem indicates the relationship between the mixing time of a Markov chain and the second largest eigenvalue magnitude of its probability transition matrix [38], [4].

Theorem 3.3.1. The mixing time of a reversible Markov chain with transition prob-

ability matrix W and second largest eigenvalue magnitude μ satisfies

$$\frac{\mu}{2(1-\mu)}(1-\ln 2) \leq T_{mix} \leq \frac{1+\log n}{1-\mu}.$$

Therefore, the speed of convergence of the Markov chain to its stationary distribution is determined by the value of $1-\mu$ known as the *spectral gap*; the larger the spectral gap, the faster the convergence. This suggests that the consensus can be reached faster on graphs with large spectral gap.

3.3.3 Global Convergence of the Inexact Newton Method

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

Convergence Properties for Dual Iteration

We will show the sequence $w(t)$ generated by the iteration (3.4), i.e.

$$w(t+1) = (D_k + I)^{-1}(B_k + I)w(t) + (D_k + I)^{-1}s,$$

converges as t goes to infinity and analyze its rate of convergence.

Recall that the matrix A is the $N \times E$ node-edge incidence matrix, and the matrix H_k is the Hessian matrix at k^{th} iteration. The graph Laplacian at k^{th} step can be written as,

$$L_k = D_k - B_k,$$

where B_k is an $N \times N$ symmetric nonegative matrix and D_k is an $N \times N$ diagonal matrix with entries

$$(D_k)_{ii} = \sum_{j \in \mathcal{N}_i} (B_k)_{ij}, \tag{3.15}$$

where $\mathcal{N}_i$ denotes the set of neighbors of node i.

We prove convergence using change of basis technique. Let the matrix $V =$

$[v_1 v_2 \ldots v_n]$, where v_i denotes the i^{th} column of V, be a $N \times N$ orthonormal matrix, whose first column is the unit vector in the direction of $(D_k+I)e$, i.e. $v_1 = \frac{(D_k+I)e}{\|(D_k+I)e\|} = c(D_k + I)e$, for some scalar $c \neq 0$. Since V is orthonormal, we have $VV' = I$. Using columns of V as the new basis, $w(t)$ can be written as $\bar{w}(t) = V'w(t)$ and hence $w(t) = V\bar{w}(t)$. Under this bijection change of basis, the sequence $w(t)$ converges if and only if the sequence $\bar{w}(t)$ converges, where the iteration on $\bar{w}(t)$ is given as

$$\bar{w}(t+1) = V'(D_k + I)^{-1}(B_k + I)V\bar{w}(t) + V'(D_k + I)^{-1}s_k. \tag{3.16}$$

We will next show that the sequence $\bar{w}(t+1)$ converges as t goes to infinity, by showing that the first component of $\bar{w}(t)$ stays constant throughout the iterations and the rest of the component converges to a vector.

For notational simplicity, we denote $P_k = (D_k + I)^{-1}(B_k + I)$. Let U denote the $N \times (N-1)$ matrix, obtained by removing the first column of V, i.e. $U = [v_2 \ldots v_n]$. Let $a(t)$ denote the first component of $\bar{w}(t)$ and $y(t)$ denote the rest, i.e. $y(t) = [\bar{w}(t)]_{2\ldots N}$. Then iteration (3.16) can be written as

$$\begin{aligned}
\begin{pmatrix} a(t+1) \\ y(t+1) \end{pmatrix} &= \begin{pmatrix} v_1' \\ U' \end{pmatrix} P_k \begin{pmatrix} v_1 & U \end{pmatrix} \begin{pmatrix} a(t) \\ y(t) \end{pmatrix} + \begin{pmatrix} v_1' \\ U' \end{pmatrix} (D_k + I)^{-1}s_k \qquad (3.17) \\
&= \begin{pmatrix} v_1'(a(t)P_k v_1 + P_k U y(t)) \\ U'(a(t)P_k v_1 + P_k U y(t)) \end{pmatrix} + \begin{pmatrix} v_1'(D_k + I)^{-1}s_k \\ U'(D_k + I)^{-1}s_k \end{pmatrix} \\
&= \begin{pmatrix} a(t)v_1'P_k v_1 + v_1'P_k U y(t) + v_1'(D_k + I)^{-1}s_k \\ U'P_k U y(t) + a(t)U'P_k v_1 + U'(D_k + I)^{-1}s_k \end{pmatrix}
\end{aligned}$$

To show $a(t)$ is constant, we first establish the following two simple lemmas. The first one explores the structure of the node-edge incidence matrix, and the second one characterizes one left eigenvalue of the matrix P_k.

Lemma 3.3.2. *Let the matrices A and H_k be the node-edge incidence matrix and the Hessian matrix at k^{th} step respectively. Let vectors $\nabla f(x_k)$ and b be the gradient vector of the cost function and the external sources respectively. Assume the problem*

$Ax = b$ has a feasible solution, then for all k, the vector $s_k = Ax_k - b - AH_k^{-1}\nabla f(x_k)$, satisfies

$$e's_k = 0.$$

Proof. Each column of the node-edge incidence matrix A consists of exactly two nonzero elements 1 and -1, hence the column sum is 0, i.e. $(e'A)_i = 0$ for all i. By the feasibility assumption, we have for some vector x, $e'b = e'Ax = 0$. Therefore

$$e's_k = e'(Ax_k - b - AH_k^{-1}\nabla f(x_k)) = (e'A)x_k - e'b - (e'A)H_k^{-1}\nabla f(x_k) = 0.$$

Q.E.D.

Lemma 3.3.3. *Let the matrix P_k be $P_k = (D_k + I)^{-1}(B_k + I)$, where the matrices B_k and D_k are defined as above. Then for all k, the vector $v_1 = \frac{(D_k+I)e}{\|(D_k+I)e\|} = c(D_k + I)e$ for some scalar $c \neq 0$ is a left eigenvector of P_k with corresponding eigenvalue 1.*

Proof. Applying definition of v_1 and P_k, we obtain

$$v_1'P_k = ce'(D_k + I)'(D_k + I)^{-1}(B_k + I) = ce'(B_k + I) = ce'(D_k + I),$$

where the second equality follows because the matrix D_k is diagonal, and the last equality follows from the fact that the matrix B_k is symmetric, and relation (3.15). Using the definition of v_1 again, we conclude the desired relation

$$v_1'P_k = v_1'. \tag{3.18}$$

Q.E.D.

We will show now that $a(t) = a$ for some constant a, for all t. From iteration (3.17), we have $a(t + 1) = a(t)v_1'P_kv_1 + v_1'P_kUy(t) + v_1'(D_k + I)^{-1}s_k$, using Lemma

3.3.3 and definition of v_1 we have

$$\begin{aligned} a(t+1) &= a(t)v_1'v_1 + v_1'Uy(t) + ce'(D_k+I)'(D_k+I)^{-1}s_k \\ &= a(t) + ce's_k \\ &= a(t), \end{aligned} \tag{3.19}$$

where the second equality follows from the fact that v_1 is a unit vector, orthogonal to the column space of the matrix U and the symmetry of the matrix D_k, and the last equality follows from Lemma 3.3.2. Hence the first component of $\bar{w}(t)$ stays constant throughout the iterations, we can rewrite the relation (3.17) as

$$\begin{pmatrix} a(t+1) \\ y(t+1) \end{pmatrix} = \begin{pmatrix} a \\ U'P_kUy(t) + aU'P_kv_1 + U'(D_k+I)^{-1}s_k \end{pmatrix}$$

Hence the iteration on $y(t)$ can be written as

$$y(t) = U'P_kUy(t) + z, \tag{3.20}$$

where $z = aU'P_kv_1 + U'(D_k+I)^{-1}s_k$ is a constant vector independent of t. To establish convergence of the sequence $y(t)$ we will use the following theorem.

Theorem 3.3.4. *Let the matrix $U'P_kU$ be as defined above, where U is the last $N-1$ orthonormal columns of the $N \times N$ square matrix V, whose first column is $v_1 = \frac{(D_k+I)e}{\|(D_k+I)e\|} = c(D_k+I)e$ for some scalar $c \neq 0$. Let the underlying graph of the network be strongly connected, then all the eigenvalues of the matrix $U'P_kU$ are subunit.*

Proof. We will prove this theorem by associating the eigenvalues of the matrix $U'P_kU$ with those of the matrix P, then provide a range for the eigenvalues of the matrix P, and lastly show all the eigenvalues of the matrix $U'P_kU$ are subunit.

(1) The set of eigenvalues of the matrix $U'P_kU$ is a subset of the eigenvalues of the matrix P.

Let x be an eigenvector associated with eigenvalue λ for the matrix $U'P_kU$, then we have

$$U'P_kUx = \lambda x. \tag{3.21}$$

Since U has orthornomal columns, we have $U'U = I$. Also for the orthonormal square matrix V, we have $VV' = \begin{pmatrix} v_1 & U \end{pmatrix} \begin{pmatrix} v_1' \\ U' \end{pmatrix} = v_1v_1' + UU' = I$, and hence

$$UU' = I - v_1v_1'. \tag{3.22}$$

Therefore, let $y = Ux$, then $x = U'y$ and the relation (3.21) can be written as

$$U'P_ky = \lambda U'y.$$

By multiplying both sides by the matrix U and relation (3.22), we obtain

$$P_ky - v_1v_1'P_ky = \lambda(y - v_1v_1'y).$$

From Lemma 3.3.3, we have $v_1'P = v_1'$. Hence the proceeding relation can be written as

$$P_ky - v_1v_1'y = \lambda y - \lambda v_1v_1'y.$$

Using the definition of y, it is in the columns space of U, which is orthogonal to v_1, i.e. $v_1'y = v_1'Ux = 0$. Therefore the above relation reduces $P_ky = \lambda y$, and hence λ is an eigenvalue of the matrix P.

(2) The spectral radius of the matrix P, $\rho(P)$, satisfies $\rho(P) = 1$, with exactly one eigenvalue whose magnitude is 1.

The matrix P_k is nonnegative and row stochastic, and when the underlying graph of the network is strongly connected, the matrix P_k is irreducible [7]. By Perron-Frobenius theorem, the spectral radius of P_k is 1, and its eigenvalues satisfy

$$1 = \lambda_1(P) > \lambda_2(P) \geq \ldots \geq \lambda_n(P) > -1. \tag{3.23}$$

Since the positive vector consisting of all 1, i.e. $e = (1, 1...1)'$, is an eigenvector of P_k with corresponding eigenvalue 1, i.e. $P_k e = e$, and the eigenvalue 1 has multiplicity of 1, all eigenvectors of P_k corresponding to the eigenvalue 1 are along the direction of e.

(3) Eigenvalue of the matrix $U'P_kU$ cannot be 1.

We show this by contradiction. Assume there exist a vector $x \neq 0$, such that $U'P_kUx = x$. Then we have

$$\begin{aligned} V'P_kV \begin{pmatrix} 0 \\ x \end{pmatrix} &= \begin{pmatrix} v_1' \\ U' \end{pmatrix} P_k \begin{pmatrix} v_1 & U \end{pmatrix} \begin{pmatrix} 0 \\ x \end{pmatrix} \\ &= \begin{pmatrix} v_1'P_kUx \\ U'P_kUx \end{pmatrix} \\ &= \begin{pmatrix} 0 \\ x \end{pmatrix}, \end{aligned}$$

where the last equality follows from Lemma 3.3.3 the fact the v_1 is orthogonal to the columns of the matrix U and the assumption that x satisfies $U'P_kUx = x$. Therefore $\begin{pmatrix} 0 \\ x \end{pmatrix}$ is an eigenvector to the matrix $V'P_kV$ with associated eigenvalue 1.

By multiplying the proceeding eigenvalue relation by V and use the relation $VV' = I$, we obtain

$$P_kV \begin{pmatrix} 0 \\ x \end{pmatrix} = V \begin{pmatrix} 0 \\ x \end{pmatrix}, \tag{3.24}$$

which implies $V \begin{pmatrix} 0 \\ x \end{pmatrix}$ is an eigenvector of the matrix P_k, with associated eigenvalue 1. In step (2), we showed that any eigenvector for P_k associated with eigenvalue 1 is along the direction of the vector e, therefore there exists a scalar $d \neq 0$, such that $V \begin{pmatrix} 0 \\ x \end{pmatrix} = de$, by multiplying both sides by the matrix V',

we have $\begin{pmatrix} 0 \\ x \end{pmatrix} = cV'e$. However $0 = c(V'e)_1 = dv_1'e = dce'(I+D)e \neq 0$, since D is a nonnegative, which leads to a contradiction. Therefore 1 cannot be an eigenvalue of the matrix $U'P_kU$.

Combining the 3 steps above, we conclude all eigenvalues of the matrix $U'P_kU$ are subunit.

Q.E.D.

Using the same idea of the proof, one can show the next lemma.

Lemma 3.3.5. *Let the matrix $U'P_kU$ be as defined above, where U is the last $N-1$ orthonormal columns of the $N \times N$ square matrix V, whose first column is $v_1 = \frac{(D_k+I)e}{\|(D_k+I)e\|} = c(D_k+I)e$ for some scalar $c \neq 0$. Let the underlying graph of the network be strongly connected, then any eigenvalue other than 1 for the matrix P_k is also an eigenvalue of the matrix $U'P_kU$.*

Proof. If a nonzero scalar $\lambda \neq 1$ is an eigenvalue of the matrix P_k with corresponding eigenvector x, then we have $P_kx = \lambda x$, and since $I = UU' + v_1v_1'$, we obtain

$$U'P_kUU'x + U'P_kv_1v_1'x = \lambda U'x. \tag{3.25}$$

Next we show that $v_1'x = 0$. Because v_1 is the left eigenvector corresponding to the eigenvalue, we have $v_1'x = v_1'P_kx = \lambda v_1'x$, where the last equality follows from the assumption that x is a right eigenvector of P_k with corresponding eigenvalue λ, since $\lambda \neq 1$, we have $v_1'x = 0$. Hence relation (3.25) implies $U'P_kUy = \lambda y$, where $y = U'x$, and therefore λ is an eigenvalue for the matrix $U'P_kU$. **Q.E.D.**

By using Theorem 3.3.4, we have the infinite sum of $\sum_{i=0}^{\infty}(U'P_kU)^i = (I - U'P_kU)$ is well defined, and $(U'P_kU)^q$ goes to 0 as q goes to infinity. Therefore the iteration

(3.20) can be solved iteratively as

$$\begin{aligned} y(t+q) &= (U'P_kU)^q y(t) + \sum_{i=0}^{q-1} (U'P_kU)^i z \\ &= (U'P_kU)^q y(t) + (I - (U'P_kU)^{q+1})(I - U'P_kU)^{-1} z \\ &= (U'P_kU)^q (y(t) - U'P_kU(I - U'P_kU)^{-1} z) + (I - U'P_kU)^{-1} z, \end{aligned}$$

Taking limit over q we have shown the convergence of the sequence $y(t)$,

$$\lim_{q \to \infty} y(t+q) = (I - U'P_kU)^{-1} z.$$

We next analyze the speed of convergence for the iteration (3.4). Using the fact that orthonormal change of basis preserve norms, we have the following relation hold,

$$||w(t) - w^*|| = ||\bar{w}(t) - \bar{w}^*|| = ||y(t) - y^*||,$$

where the first equality holds due to the fact that $w(t) = V\bar{w}(t)$ and columns of the matrix V are orthonormal, the second equality follow from the established fact that the first component of $\bar{w}(t)$ stays constant [cf. Eq. (3.19)], so that the norm of the difference only depends on the other $N-1$ components.

For the rest of the convergence rate analysis, we restrict our attention to the case when the $(N-1) \times (N-1)$ matrix $U'P_kU$ has $N-1$ linearly independent eigenvectors (for the other case, see Section 2.5.1 for more details). By applying Lemma 2.5.1, we have that the rate of convergence of $y(t)$ depends on the largest eigenvalue magnitude of the matrix $U'P_kU$, which by Lemma 3.3.5 is the same as the second largest eigenvalue magnitude of the matrix P. Let λ denote the eigenvalue of the matrix P, which has the second largest magnitude, then from Lemma 2.5.1, we have

$$||y(t) - y^*|| = \left|\left|(U'P_kU)^t (y(0) - U'P_kU(I - U'P_kU)^{-1} z)\right|\right| \leq \lambda^t \alpha,$$

where α is a positive scalar depends on $y(0) - U'P_kU(I - U'P_kU)^{-1}z$. Therefore

$$||w(t) - w^*|| \leq \lambda^t \alpha,$$

where α depends on the initial vector $w(0)$. Hence the larger the spectral gap, i.e. $1 - |\lambda|$, the matrix P has, the faster the dual iteration converges, and the convergence rate is linear.

The next two sections provide convergence rate estimates for the damped Newton phase and the local convergence phase of the inexact Newton method in the primal domain when there are errors in the Newton direction and the backtracking stepsize.

Convergence Rate for Damped Newton Phase

We first show a strict decrease in the norm of the residual function if the errors ϵ and γ are sufficiently small, as quantified in the following assumption.

Assumption 6. The errors B and e [cf. Eqs. (3.14) and (3.12)] satisfy

$$B + 2\gamma \leq \frac{\beta}{16M^2L},$$

where β is the constant used in the inexact backtracking stepsize rule, and M and L are the constants defined in Assumption 5.

Under this assumption, the next proposition establishes a strict decrease in the norm of the residual function as long as $\|r(y)\| > \frac{1}{2M^2L}$.

Proposition 2. Let Assumptions 5 and 6 hold. Let $\{y_k\}$ be a sequence generated by the method (3.6) when the stepsize sequence $\{\alpha_k\}$ is selected using the inexact backtracking stepsize rule [cf. Eq. (3.13)]. Assume that $\|r(y_k)\| > \frac{1}{2M^2L}$. Then, we have

$$\|r(y_{k+1})\| \leq \|r(y_k)\| - \frac{\beta}{16M^2L}.$$

Proof. For any $k \geq 0$, we define

$$\bar{\alpha}_k = \frac{1}{2M^2L(n_k + \gamma/2)}.$$

In view of the condition on n_k [cf. Eq. (3.12)], we have

$$\frac{1}{2M^2L(\|r(y_k)\|+\gamma)} \leq \bar{\alpha}_k \leq \frac{1}{2M^2L\|r(y_k)\|} < 1, \tag{3.26}$$

where the last inequality follows by the assumption $\|r(y_k)\| > \frac{1}{2M^2L}$. Using the preceding relation and substituting $\alpha_k = \bar{\alpha}_k$ in the basic relation in Proposition 3.12, we obtain:

$$\begin{aligned}
\|r(y_{k+1})\| \;\leq\; & \|r(y_k)\| + \bar{\alpha}_k\|\epsilon_k\| + M^2L\bar{\alpha}_k^2\|\epsilon_k\|^2 \\
& -\bar{\alpha}_k\|r(y_k)\|\Big(1 - M^2L\bar{\alpha}_k\|r(y_k)\|\Big) \\
\leq\; & \|r(y_k)\| + \bar{\alpha}_k\|\epsilon_k\| + M^2L\bar{\alpha}_k^2\|\epsilon_k\|^2 \\
& -\bar{\alpha}_k\|r(y_k)\|\Big(1 - M^2L\frac{\|r(y_k)\|}{2M^2L\|r(y_k)\|}\Big) \\
\leq\; & \bar{\alpha}_k\|\epsilon_k\| + M^2L\bar{\alpha}_k^2\|\epsilon_k\|^2 + \Big(1 - \frac{\bar{\alpha}_k}{2}\Big)\|r(y_k)\| \\
\leq\; & B + \Big(1 - \frac{\bar{\alpha}_k}{2}\Big)\|r(y_k)\|,
\end{aligned}$$

where the second inequality follows from the definition of $\bar{\alpha}_k$ and the third inequality follows by combining the facts $\bar{\alpha}_k < 1$, $\|\epsilon_k\| \leq \epsilon$ for all k, and the definition of B. The constant σ used in the definition of the inexact backtracking line search satisfies $\sigma \in (0, 1/2)$, therefore, it follows from the preceding relation that

$$\|r(y_{k+1})\| \leq (1 - \sigma\bar{\alpha}_k)\|r(y_k)\| + B.$$

Using condition (3.12) once again, this implies

$$n_{k+1} \leq (1 - \sigma\bar{\alpha}_k)n_k + B + \gamma,$$

showing that the steplength α_k selected by the inexact backtracking line search sat-

isfies $\alpha_k \geq \beta\bar{\alpha}_k$. From condition (3.13), we have

$$n_{k+1} \leq (1 - \sigma\alpha_k)n_k + B + \gamma,$$

which implies

$$\|r(y_{k+1})\| \leq (1 - \sigma\beta\bar{\alpha}_k)\|r(y_k)\| + B + 2\gamma.$$

Combined with Eq. (3.26), this yields

$$\|r(y_{k+1})\| \leq \Big(1 - \frac{\sigma\beta}{2M^2L(\|r(y_k)\| + \gamma)}\Big)\|r(y_k)\| + B + 2\gamma.$$

By Assumption 6, we also have

$$\gamma \leq B + 2\gamma \leq \frac{\beta}{16M^2L},$$

which in view of the assumption $\|r(y_k)\| > \frac{1}{2M^2L}$ implies that $\gamma \leq \|r(y_k)\|$. Substituting this in the preceding relation and using the fact $\alpha \in (0, 1/2)$, we obtain

$$\|r(y_{k+1})\| \leq \|r(y_k)\| - \frac{\beta}{8M^2L} + B + 2\gamma.$$

Combined with Assumption 6, this yields the desired result. **Q.E.D.**

The preceding proposition shows that, under the assumptions on the size of the errors, at each iteration, we obtain a minimum decrease (in the norm of the residual function) of $\frac{\beta}{16M^2L}$, as long as $\|r(y_k)\| > 1/2M^2L$. This establishes that we need at most

$$\frac{16\|r(y_0)\|M^2L}{\beta},$$

iterations until we obtain $\|r(y_k)\| \leq 1/2M^2L$.

Convergence Rate for Local Convergence Phase

In this section, we show that when $\|r(y_k)\| \leq 1/2M^2L$, the inexact backtracking stepsize rule selects a full step $\alpha_k = 1$ and the norm of the residual function $\|r(y_k)\|$

converges quadratically within an error neighborhood, which is a function of the parameters of the problem (as given in Assumption 5) and the error level in the constrained Newton direction.

Proposition 3. Let Assumption 5 hold. Let $\{y_k\}$ be a sequence generated by the method (3.6) when the stepsize sequence $\{\alpha_k\}$ is selected using the inexact backtracking stepsize rule [cf. Eq. (3.13)]. Assume that there exists some k such that

$$\|r(y_k)\| \leq \frac{1}{2M^2L}.$$

Then, the inexact backtracking stepsize rule selects $\alpha_k = 1$. We further assume that for some $\delta \in (0, 1/2)$,

$$B + M^2LB^2 \leq \frac{\delta}{4M^2L},$$

where B is the constant defined in Eq. (3.14). Then, we have

$$\|r(y_{k+m})\| \leq \frac{1}{2^{2^m}M^2L} + B + \frac{\delta}{M^2L}\frac{(2^{2^m-1}-1)}{2^{2^m}} \tag{3.27}$$

for all $m > 0$. As a particular consequence, we obtain

$$\limsup_{m\to\infty} \|r(y_m)\| \leq B + \frac{\delta}{2M^2L}.$$

Proof. We first show that if $\|r(y_k)\| \leq \frac{1}{2M^2L}$ for some $k > 0$, then the inexact backtracking stepsize rule selects $\alpha_k = 1$. Replacing $\alpha_k = 1$ in the basic relation of Proposition 3.12 and using the definition of the constant B, we obtain

$$\begin{aligned}
\|r(y_{k+1})\| &\leq M^2L\|r(y_k)\|^2 + B \\
&\leq \frac{1}{2}\|r(y_k)\| + B \\
&\leq (1-\sigma)\|r(y_k)\| + B,
\end{aligned}$$

where to get the last inequality, we used the fact that the constant σ used in the inexact backtracking stepsize rule satisfies $\sigma \in (0, 1/2)$. Using the condition on n_k

[cf. Eq. (3.12)], this yields

$$n_{k+1} \leq (1-\sigma)n_k + B + \gamma,$$

showing that the steplength $\alpha_k = 1$ satisfies condition (3.13) in the inexact backtracking stepsize rule.

We next show Eq. (3.27) using induction on the iteration m. Using $\alpha_k = 1$ in the basic relation of Proposition 3.12, we obtain

$$\|r(y_{k+1})\| \leq \frac{1}{2}\|r(y_k)\| + B \leq \frac{1}{4M^2L} + B,$$

where the second inequality follows from the assumption $\|r(y_k)\| \leq \frac{1}{2M^2L}$. This establishes relation (3.27) for $m = 1$.

We next assume that (3.27) holds for some $m > 0$, and show that it also holds for $m+1$. Eq. (3.27) implies that

$$\|r(y_{k+m})\| \leq \frac{1}{4M^2L} + B + \frac{\delta}{4M^2L}.$$

Using the assumption $B + M^2LB^2 \leq \frac{\delta}{4M^2L}$, this yields

$$\|r(y_{k+m})\| \leq \frac{1+2\delta}{4M^2L} < \frac{1}{2M^2L},$$

where the strict inequality follows from $\delta \in (0, 1/2)$. Hence, the inexact backtracking stepsize rule selects $\alpha_{k+m} = 1$. Using $\alpha_{k+m} = 1$ in the basic relation, we obtain

$$M^2L\|r(y_{k+m+1})\| \leq \Big(M^2L\|r(y_{k+m})\|\Big)^2 + B.$$

Using Eq. (3.27), this implies that

$$
\begin{aligned}
& M^2L\|r(y_{k+m+1})\| \\
& \leq \left(\frac{1}{2^{2^m}} + M^2LB + \frac{\delta(2^{2^m-1}-1)}{2^{2^m}}\right)^2 + M^2LB \\
& = \frac{1}{2^{2^{m+1}}} + \frac{M^2LB}{2^{2^m-1}} + \delta\frac{2^{2^m-1}-1}{2^{2^{m+1}-1}} \\
& \quad + M^2L\left(B + \frac{\delta}{M^2L}\frac{(2^{2^m-1}-1)}{2^{2^m}}\right)^2 + M^2LB.
\end{aligned}
$$

Using algebraic manipulations and the assumption $B + M^2LB^2 \leq \frac{\delta}{4M^2L}$, this yields

$$
\|r(y_{k+m+1})\| \leq \frac{1}{2^{2^{m+1}}M^2L} + B + \frac{\delta}{M^2L}\frac{(2^{2^{m+1}-1}-1)}{2^{2^{m+1}}},
$$

completing the induction and therefore the proof of relation (3.27). Taking the limit superior in Eq. (3.27) establishes the final result. **Q.E.D.**

3.4 Simulation Results

Our simulation results demonstrate that the decentralized Newton significantly outperforms the dual subgradient method algorithm in terms of runtime. Simulations were conducted as follows: Network flow problems with conservation constraints and the cost function $\Phi(x) = \sum_{e=1}^{E} \phi_e(x^e)$ where $\phi_e(x^e) = 1 - \sqrt{1-(x^e)^2}$ [3] were generated on Erdös-Rényi random graphs with $n = 10$, 20, 80, and 160 nodes and an expected node degree, $np = 5$. We limit the simulations to optimization problems which are well behaved in the sense that the Hessian matrix remains well conditioned, defined as $\frac{\lambda_{\max}}{\lambda_{\min}} \leq 200$. For this subclass of problem, the runtime of the Newton's method algorithm is significantly less than the subgradient method for all trials. Note that the stopping criterion is also tested in a distributed manner for both algorithms.

In any particular experiment the Newton's method algorithm discovers the feasible direction in one iteration and proceeds to find the optimal solution in two to four additional iterations. One such experiment is shown in Figure 3-1. A sample runtime distribution for 150 trials is presented in Figure 3-2. The fact that of course Newton's

[3] the cost function is motivated by the Kuramoto model of coupled nonlinear oscillators [23].

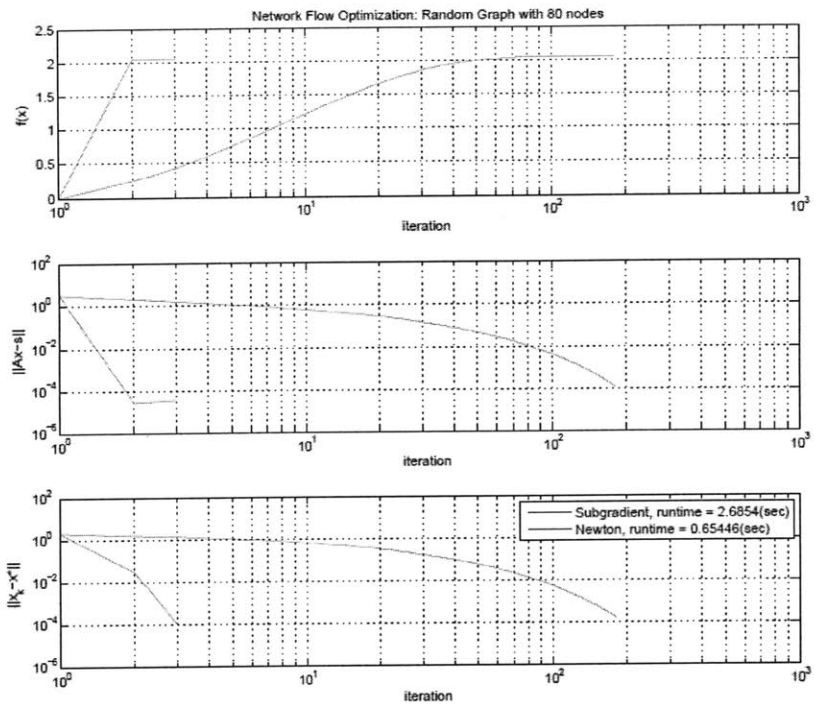

Figure 3-1: Sample convergence trajectory for a random graph with 80 nodes and mean node degree 5.

method outperform the subgradient scheme is not surprising. Perhaps the surprising fact is that Newton's method outperforms subgradient scheme, despite the fact the computation of the dual Newton step is based on an iterative scheme.

On average the Newton's method terminates in less than half of the subgradient runtime and exhibits a tighter variance. This is a representative sample with respect to varying the number of nodes. As shown in Figure 3-3, the Newton's method algorithm completes in significantly less time on average for all of the graphs evaluated.

We also tested the performance of the proposed method on graphs with different connectivity properties. In particular, we considered a complete (fully connected) graph and a sparse graph. Figures 3-4 and 3-5 compare the performance of the subgradient scheme with the Newton method. The Newton method outperforms the subgradient scheme in both cases. As expected, the performance gains are more significant for the complete graph since the dual Newton step can be computed efficiently on graphs with large spectral gap.

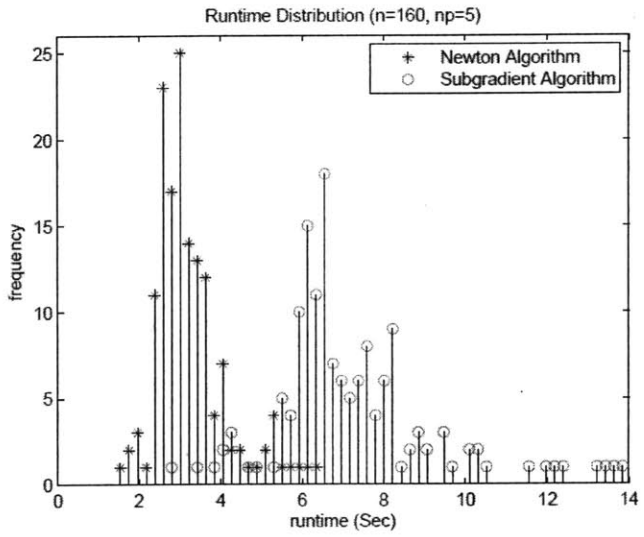

Figure 3-2: Runtime Histogram, 160 node graphs with mean node degree 5

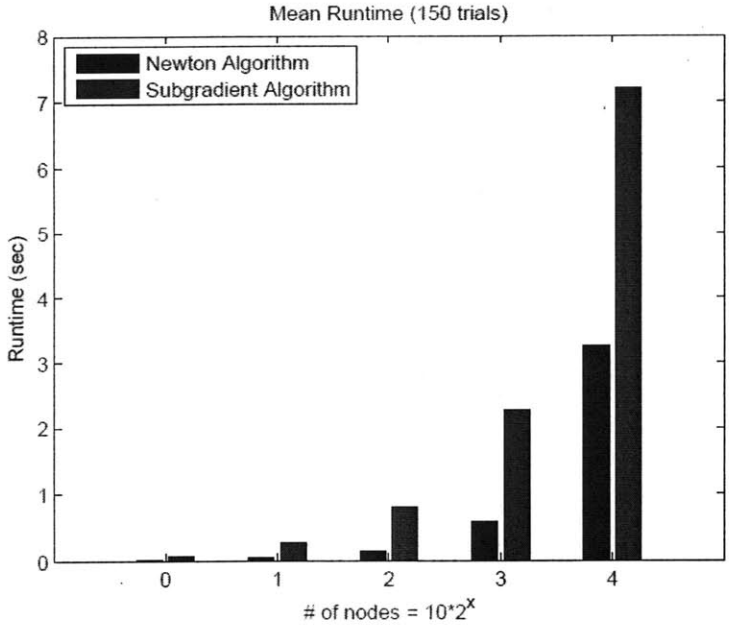

Figure 3-3: Average Runtime, 150 samples each

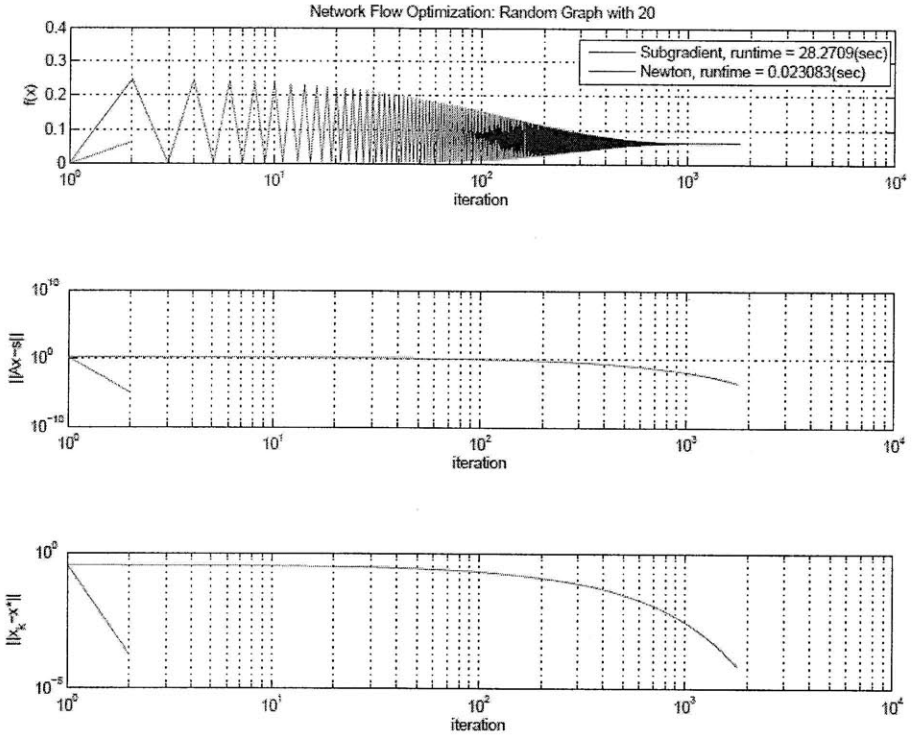

Figure 3-4: Sample convergence trajectory for a complete graph

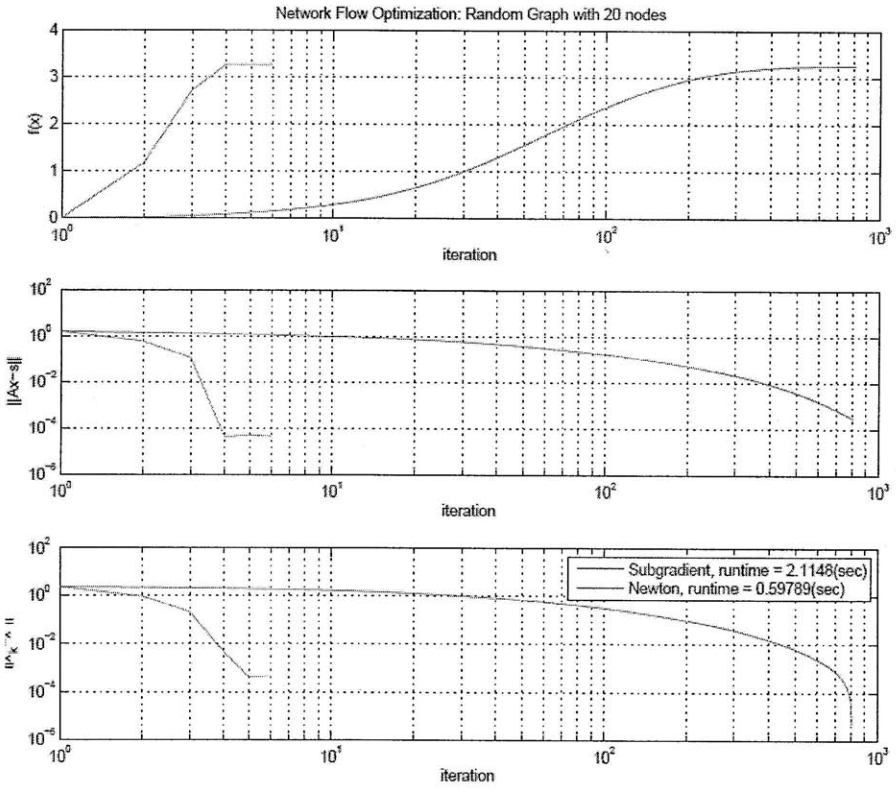

Figure 3-5: Sample convergence trajectory for a sparse graph

3.5 Conclusions

This chapter develops a distributed Newton-type method for solving network flow cost minimization problems that can achieve the superlinear convergence rate within some error neighborhood. We show that due to the sparsity structure of the incidence matrix of a network, the computation of the dual Newton step can be performed using iterative scheme in a distributed way. This enables using distributed consensus schemes to compute the dual Newton direction. 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. Our simulation experiments on different graphs suggested the superiority of the proposed Newton scheme to standard dual subgradient methods. For future works, we believe new results in [16] will enable us to solve the dual Newton step using modified PageRank algorithms in a more scalable fashion [15].

Chapter 4

Conclusions

This thesis develops and analyzes Newton-type distributed algorithms for two particular network resource allocation problems: Network Utility Maximization (NUM) problems, and network flow cost minimization problems. The existing methods for both problems rely on dual decomposition and subgradient methods, which is scalable but slow in convergence. Our algorithms uses consensus-based schemes, are both fast and can be implemented in a distributed way. For both problems we show even when the Newton direction and stepsize 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. We provide simulation results for both problems to reflect significant convergence speed improvements over the existing methods. In particular, even for relatively small networks, our algorithms improve dramatically in the runtime over existing distributed ones, without significant increase in the information exchange overhead. For larger networks, this improvement is expected to scale with the size and be more striking.

Separately, for the NUM problem, we use novel matrix splitting techniques, so that both primal and dual updates for the Newton step can be computed using iterative schemes in a decentralized manner with limited information exchange. Specifically, the information exchange in the new algorithm is comparable with that of the existing algorithms. We utilize properties of self-concordant utility functions, because due to the inequality constraint structure and usage of barrier functions, Lipschitz-based

results cannot be applied.

For the network flow optimization problem, the key component of our method is to represent the dual Newton direction as the limit of an iterative procedure involving the graph Laplacian, which can be implemented based only on local information. We are able to use Liptschitz-based analysis in this problem, due to the favorable equality constraint structure.

Future work includes the analysis of the relationship between the rate of converge and the underlying topology. Our simulation result from Section 2.6 suggests some correlation in the performance across the 3 methods, i.e. subgradient method, diagonal scaled Newton-type method and our distributed Newton algorithm, which illuminates the possible relation between the algorithms and the network topology. This relation can potentially be universal for a broader class of algorithms.

Bibliography

[1] A. Ozdaglar A. Nedic. Convergence Rate for Consensus with Delays. *to appear in Journal of Global Optimization*, LIDS report 2774, 2008.

[2] D. Acemoglu, A. Ozdaglar, and A. ParandehGheibi. Spread of (Mis)information in Social Networks. *to appear in Games and Economic Behavior*, LIDS report 2812, 2009.

[3] D. P. Bertsekas ad E. M. Gafni. Projected newton Methods and Optimization of Multicommodity Flows. *IEEE Transaction on Automatic Control*, AC-28(12):1090–1096, 1983.

[4] D. Aldous. Some Inequalities for Reversible Markov Chains. *Journal of the London Mathematical Society.*

[5] G. D. Allen. *Lectures on Linear Algebra and Matrices.* Department of Mathematics, Texas AM University, 2003.

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

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

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

[9] D. P. Bertsekas, A. Nedic, and A. E. Ozdaglar. *Convex Analysis and Optimization.* Athena Scientific, Cambridge, MA, 2003.

[10] D. P. Bertsekas and J. Tsitsiklis. *Parallel and distributed computation: Numerical methods.* Athena Scientific, Belmont, MA, 1997.

[11] 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.

[12] N. Biggs. Algebraic Graph Theory. 1993.

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

[14] 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.

[15] F. Chung. PageRank as a Discrete Green's Function. preprint.

[16] F. Chung. The Heat Kernel as the PageRank of a Graph. *Proceedings of the National Academy of Sciences*, 104(19735), no. 50, 2007.

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

[18] M. Dahleh, M. A. Dahleh, and G. Verghese. *Lectures on Dynamic Systems and Control.* Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology.

[19] P. Diaconis and D. Stroock. Geometric Bounds for Eigenvalues of Markov Chains. *Annals of Applied Probability*, 1(1):36–61, 1991.

[20] A. V. Fiacco and G. P. McCormick. *Nonlinear Programming: Sequential Unconstrained Minimization Techniques.* SIAM, 1990.

[21] R. Horn and C. R. Johnson. *Matrix analysis.* Cambridge University Press, New York, 1985.

[22] 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.

[23] A. Jadbabaie, N. Motee, and M. Barahona. On the Stability of the Kuramoto Model of Coupled Nonlinear Oscillators. *Proceedings of the American Control Conference*, 2004.

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

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

[26] C.T. Kelley. *Iterative Methods for Linear and Nonlinear Equations.* SIAM, Philadelphia, PA, 1995.

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

[28] 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.

[29] D. C. Lay. *Linear Algebra and its Applications.* Person Education, third edition, 2006.

[30] 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.

[31] 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.

[32] A. Nedic and A. Ozdaglar. Subgradient Methods in Network Resrouce Allocation: Rate Analysis. *Proc. of CISS*, 2008.

[33] A. Nedic and A. Ozdaglar. Approximate Primal Solutions and Rate Analysis for Dual Subgradient Methods. *SIAM Journal on Optimization*, 19(4):1757–1780, 2009.

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

[35] R. Olfati, S. Richard, and M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on Automatic Control*, 49(9):1520–1533, 2004.

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

[37] S. Eisenstat R. Dembo and T. Steihaug. Inexact Newton Methods. *SIAM Journal on Numerical Analysis*, 19, 1982.

[38] A. Sinclair. Improved bounds for mixing rates of markov chains and multicommodity flow. *Combinatorics, Probability and Computing*, 1:351–370, 1992.

[39] R. Srikant. *The Mathematics of Internet Congestion Control (Systems and Control: Foundations and Applications).* SpringerVerlag, 2004.

[40] J. 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.

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

[42] 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.

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