Accelerated Dual Descent for Network Flow Optimization

Michael Zargham[†], Alejandro Ribeiro[†], Asuman Ozdaglar[‡], Ali Jadbabaie[†]

***Abstract*—We present a fast distributed solution to the convex network flow optimization problem. Our approach uses a family of dual descent algorithms that approximate the Newton direction to achieve faster convergence rates than existing distributed methods. The approximate Newton directions are obtained through matrix splitting techniques and sparse Taylor approximations of the inverse Hessian. We couple this descent direction with a distributed line search algorithm which requires the same information as our descent direction to compute. We show that, similarly to conventional Newton methods, the proposed algorithm exhibits superlinear convergence within a neighborhood of the optimal value. Numerical experiments corroborate that convergence times are between one to two orders of magnitude faster than existing distributed optimization methods. A connection with recent developments that use consensus to compute approximate Newton directions is also presented.**

I. Introduction

This paper develops accelerated dual descent (ADD) algorithms for solving the minimum convex cost network flow problem using a limited number of local information exchanges while guaranteeing superlinear convergence to a neighborhood of the optimum. The convex min cost network flow problem and the study of its dual problem are key building blocks in the study of network optimization because they are close to combinatorial problems such as the shortest path problem, [1, Chapter 1], [2]. Solutions to min cost network flow problems have long been used in operations research and transportation networks [3], [4]. In particular, see the uncapacitated transshipment problem in [5]. Network flow problems and their duals are also relevant to computer vision [6] and the robust routing problem [7] where the objective is to choose a routing strategy with minimal variance when the edges are noisy communication channels. Furthermore, the network flow problem is a key subproblem in wireless routing and resource allocation [8]. Our formulation is also a stepping stone for the queue stabilization problem in a capacitated network with multiple commodity types and stochastic arrival rates, [9], [10]. In [11], we build on [12] to characterize the effects of incorporating capacity constraints in the minimum cost network flow problem. In [13], we construct an accelerated backpressure algorithm (ABP), which applies the ADD framework to solve queue stabilization problem. ABP follows the from of backpressure but rather than routing based on differentials in queue lengths, we route based on differentials in queue priorities which are dual variables computed according to ADD.

Minimum convex cost network flow problems can be solved in a distributed manner via dual subgradient descent, [1]. Nodes keep track of variables associated with their outgoing edges and undertake updates based on their local variables and variables available at adjacent nodes. Analysis of subgradient methods for distributed convex optimization can be found in [14] and [15] with the latter taking into account uncertainty int he network structure. However, practical applicability of the resulting algorithms is limited by exceedingly slow convergence rates, [16]. An alternative distributed algorithm based on the Gauss Seidel method is present in [17]. Like gradient descent, Gauss Seidel is a first order method.

The natural alternative to accelerate convergence is to use second order Newton methods [18, Algorithm 9.2], but they cannot be implemented in a distributed manner because matrix inversion is a global operation. Early works on Newton type methods for network optimization are found in [19], [20]. Both of these methods, however, are not fully distributed because they require some level of global coordination. Efforts to overcome this shortcoming include approximating the Hessian inverse with the inverse of its diagonals [21] and the use of consensus iterations to approximate the Newton step [22]. Improvements over subgradient methods can be achieved through Nesterov type accelerated methods summarized in [23]. These methods work well when the proximal operator has a simple closed form. To be distributed we require that the proximal operator

This research is supported by NSF CAREER CCF-0952867 and ONR BRC N00014-12-1-0997.

[†]Zargham, Ribeiro and Jadbabaie are with the Department of Electrical and Systems Engineering, University of Pennsylvania. Address: 200 South 33rd Street, Philadelphia, PA, 19104. Email: {zargham, aribeiro, jadbabai}@seas.upenn.edu.

[‡] Asuman Ozdaglar is with the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology. Address: 77 Massachusetts Avenue, Cambridge , MA 02139. Email: asuman@mit.edu.

be computable using local information, which limits the convergence rates that can be achieved. Krylov subspace methods, in particular conjugate gradient descent can achieve second order convergence when computed centrally, see chapter 6 of [24]. Unfortunately, conjugate gradient and other second order Krylov subspace methods rely heavily on inner products for an orthogonalization procedure which leads to improved convergence rates, see chapter 9 of [25]. Even in the best case, these inner products violate the communication limitations for this problem, for example [20].

The dual Hessian is a weighted Laplacian of the graph representing our communication network. Using this structure we can approximate the dual Hessian inverse using local information. Our particular insight is to consider a Taylor's expansion of the inverse Hessian [26, Section 5.8], which, being a polynomial with the Hessian matrix as variable, can be implemented through local information exchanges. More precisely, considering only the zeroth order term in the Taylor's expansion yields an approximation to the Hessian inverse based immediately available information. The first order approximation requires information available at neighboring nodes and in general, the Nth order approximation necessitates information from nodes located N hops away. The resultant family of ADD algorithms permits a tradeoff between the accuracy of the Hessian approximation and communication cost. We use ADD-N to represent the Nth member of the ADD family which uses information from terminals N hops away. To guarantee global convergence of the ADD-N algorithm we further introduce an approximate backtracking line search based on the work in [27].

In Section II we introduce the convex network flow optimization problem formally and state our assumptions. Basic results in Section III demonstrate the effect of our assumptions on the dual formulation. We also review dual gradient descent and dual Newtons method before proposing the ADD algorithm in Section IV. In Section IV-B we introduce our distributed backtracking line search algorithm. The main result of the paper is our proof that the ADD family follows three distinct convergence phases, found in Section V. The first two phases are akin to the linear and quadratic phases of Newtons method but in the terminal phase the Newton error begins to accumulate. In Section VI, we show that ADD can be implemented using a consensus scheme. We conduct numerical experiments in Section VII, demonstrating that ADD-N leads to a significant reduction in communication overhead as compared to gradient descent and the consensus based method. We evaluate the effect of using our distributed line search by comparing against solutions generated using a centralized line search. Finally, in Section VII-C we implement ADD-1 for the robust routing problem from [7] demonstrating practical convergence times where gradient descent is considered impractically slow.

II. Preliminaries

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 $\mathcal{E} = \{1, \ldots, E\}$. The network is deployed to support a single information flow specified by incoming rates $b_i > 0$ at source nodes and outgoing rates $b_i < 0$ at sink nodes. Rate requirements are collected in a vector b, which to ensure problem feasibility has to satisfy $\sum_{i=1}^{n} b_i = \mathbf{1}^T b = 0$. Our goal is to determine a flow vector $x = [x_e]_{e \in \mathcal{E}}$, with x_e denoting the amount of flow on edge $e = (i, j)$. Flow conservation is enforced as $Ax = b$, where A the $n \times E$ node-edge incidence matrix defined

$$[A]_{ie} = \begin{cases} 1 & \text{if edge } e \text{ leaves node } i, \\ -1 & \text{if edge } e \text{ enters node } i, \\ 0 & \text{otherwise.} \end{cases}$$

The Algebraic connectivity of $\mathcal{G}$ is the second smallest eigenvalue of the graph Laplacian AA'. We define the penalty as a convex scalar cost function $\phi_e(x_e)$ denoting the cost of x_e units of flow traversing edge e. The convex min-cost flow network optimization problem is then defined as

$$\text{minimize } f(x) = \sum_{e=1}^{E} \phi_e(x_e), \quad \text{subject to: } Ax = b. \tag{1}$$

Assumption 1. *The Network $\mathcal{G}$ has the following properties:*

(a) Connected with algebraic connectivity lower bounded by a constant ω

(b) Non-bipartite

In Assumption 1(a), we quantify the ability of the network to spread information via an algebraic connectivity bound. Assumption 1(b) that $\mathcal{G}$ is non-bipartite guarantees that the normalized Laplacian on the $\mathcal{G}$ has largest eigenvalue strictly upper bounded by 2. Due to our use of λ for dual variables, we use $\mu(X)$ to denote eigenvalues of a symmetric matrix $X \in \mathbb{S}^n$ and we order them $|\mu_1| \leq |\mu_2| \leq \cdots \leq |\mu_n|$.

Assumption 2. *The objective functions $\phi_e(\cdot)$ have the following properties for all e:*

(a) Twice continuously differentiable, strongly convex and satisfies $\gamma \leq \phi_e''(\cdot) \leq \Gamma$

(b) Lipschitz Hessian Inverse $|1/\phi_e''(y) - 1/\phi_e''(\bar{y})| \leq \xi|y - \bar{y}|$

Assumption 2 restricts the objectives the primal to those which will yield a dual problem meeting the standard criteria for application of Newton's method, [18][Chapter 9.5]. These assumptions are sufficient for convergence but are not necessary. Restricting to this case allows us to focus on the core mechanisms of a Newton type method. For analysis of relaxations on these conditions, the reader is directed to [28] and [29].

III. Network Optimization

In this work we focus our attention on solutions in the dual domain. Computing the Lagrange dual of a convex minimization with equality constraints yields maximization of a concave function. In the case of (1) the Lagrange dual is given by

$$\max_{\lambda} f(x(\lambda)) - \lambda'(Ax(\lambda) - b) \tag{2}$$

where the primal optimizers of the Lagrangian are defined

$$x(\lambda) = \arg\min_{x} f(x) - \lambda'(Ax - b). \tag{3}$$

Due to the separability of the objective $f(x) = \sum_e \phi_e(x_e)$ and Assumption 2 we can compute the flow on edge $e = (i,j)$ directly from the dual variables at node i and j according to

$$x_e(\lambda) = (\phi_e')^{-1}(\lambda^i - \lambda^j). \tag{4}$$

Superscript notation, λ^i is used for elements of the vector $\lambda \in \mathbb{R}^n$. The subscript is reserved for the time index k, introduced in the next subsection. For notational convenience we cast the dual as minimization

$$\min_{\lambda} \; q(\lambda) \;\; = \;\; \lambda'(Ax(\lambda) - b) - f(x(\lambda)) \tag{5}$$

by minimizing the negation of the objective in (2). From this point on we will consider solutions to the dual problem (5) and use (4) to compute the associated primal optimal variables. To further proceed we outline the consequences of Assumptions 1 and 2 with regards to the dual problem, (5).

Lemma 1. *The dual objective* $q(\lambda) = \lambda'(Ax(\lambda) - b) - f(x(\lambda))$ *has the following properties.*

(a) The dual Hessian is the weighted Laplacian

$$\nabla^2 q(\lambda) = A[\nabla^2 f(x(\lambda))]^{-1}A',$$

(b) is strongly convex on the subspace $\mathbf{1}^\perp$ *and satisfies*

$$\frac{1}{M}v'v \leq v'\nabla^2 q(\lambda)v \leq \frac{1}{m}v'v \qquad \forall v \in \mathbf{1}^\perp,$$

(c) and is a Lipschitz function of λ*, i.e.,*

$$||\nabla^2 q(\lambda) - \nabla^2 q(\bar{\lambda})|| \leq L||\lambda - \bar{\lambda}||.$$

Proof: See Appendix A for the proof of Lemma 1. ■

Lemma 1 recovers the key assumptions of Newton's Method defined in [18, Section 9.5], for the problem defined in (5). These results are direct consequences of Assumptions 1 and 2.

A. Gradient Descent

The benchmark distributed solution to (1) is the dual subgradient method. In our problem $q(\lambda)$ is differentiable so we have access to the gradient $g(\lambda) = \nabla q(\lambda)$. Consider an iteration index k, an arbitrary initial vector λ_0 and define iterates λ_k generated by the following recursion

$$\lambda_{k+1} = \lambda_k - \alpha_k g_k \qquad \text{for all } k \geq 0, \tag{6}$$

where $g_k = g(\lambda_k) = \nabla q(\lambda_k)$ denotes the gradient of the dual function $q(\lambda)$ at $\lambda = \lambda_k$. A first important observation here is that we can compute the gradient as $g_k = Ax(\lambda_k) - b$ with the vector $x(\lambda_k)$ having components $x_e(\lambda_k)$ as determined by (4) with $\lambda = \lambda_k$, [30, Section 6.4]. A second important observation is that because of the sparsity pattern of the node-edge incidence matrix A the ith element $[g_k]_i$ of the gradient g_k can be computed as

$$[g_k]_i = \sum_{e=(i,j)} x_e(\lambda_k) - \sum_{e=(j,i)} x_e(\lambda_k) - b_i \tag{7}$$

The algorithm in (6)-(7) lends itself to distributed implementation. Each node i maintains information about its dual iterates $[\lambda_k]_i$ and primal iterates $x_e(\lambda_k)$ of outgoing edges $e = (i,j)$. Gradient components $[g_k]_i$ are evaluated as per (7) using local primal iterates $x_e(\lambda_k)$ for $e = (i,j)$ and primal iterates of neighboring nodes $x_e(\lambda_k)$ for $e = (j,i)$. Dual variables are then updated as per (6). We proceed to update primal variables as per (4). This update necessitates local multipliers $[\lambda_k]_i$ and neighboring multipliers $[\lambda_k]_j$.

Distributed implementation is appealing because it avoids the cost and fragility of collecting all information at a centralized location. However, practical applicability of gradient descent algorithms is hindered by slow convergence rates; see e.g., [31], [32].

B. Newton's Method

The Newton method is a scaled version of gradient descent. In lieu of (6) iterates are given by

$$\lambda_{k+1} = \lambda_k + \alpha_k d_k \qquad \text{for all } k \geq 0, \tag{8}$$

where d_k is the Newton direction at iteration k and α_k is a properly selected step size. The Newton direction,

$$H_k d_k = -g_k, \tag{9}$$

where $H_k = H(\lambda_k) = \nabla^2 q(\lambda_k)$ is the Hessian of the dual function. From Lemma 1(a) we have

$$H_k = A[\nabla^2 f(x(\lambda_k))]^{-1} A'. \tag{10}$$

From the definition of $f(x)$ in (1) it follows that the primal Hessian $\nabla^2 f(x_k)$ is a diagonal matrix. From Assumption 1(a) we know $[\nabla^2 f(x_k)]^{-1}$ exists and can be computed locally because it is diagonal. From Lemma 1(a) we know that H_k is a weighted Laplacian of the connected graph $\mathcal{G}$ and thus has rank $n-1$ and zero eigenvalue associate with the eigenvector $\mathbf{1}$. Since $g_k \in \mathbf{1}^\perp$, the pseudo inverse can be used to exactly compute the Newton direction

$$d_k = -H_k^\dagger g_k. \tag{11}$$

However, the pseudoinverse $H_k^\dagger$ is a dense matrix and computing d_k requires global information. We are therefore interested in approximations of the Newton direction requiring local information only.

IV. Accelerated Dual Descent

To define an approximate Newton direction, i.e., one for which (9) is approximately true, we consider a finite number of terms of a suitable Taylor's expansion representation of the Newton direction. In order to proceed with this approach we first define a matrix splitting.

Definition 1. *Define the* ***Matrix Splitting*** $H_k = D_k - B_k$, *where diagonal matrix* D_k *is constructed*

$$[D_k]_{ii} = 2[H_k]_{ii} \qquad \forall i \in \mathcal{V} \tag{12}$$

and the matrix B_k *is*

$$[B_k]_{ii} = [H_k]_{ii}\, \forall i \qquad \textit{and} \qquad [B_k]_{ij} = -[H_k]_{ij}\, \forall i, j. \tag{13}$$

This splitting is motivated by the fact that the product $D_k^{-1/2} B_k D_k^{-1/2} = 1/2(I + P)$ closely similar to a lazy random walk on $\mathcal{G}$, [33, Chapter 1] because $2D_k^{-1/2} P D_k^{1/2}$ is a random walk matrix with no self-loops. Using the lazy random walk which consists of adding self loops and rescaling the weights, removes any periodicity that would arise on the random walk. An important consequence is that B_k is nonnegative and has eigenvalues in $[0, 1]$. These properties come from the fact that H_k is a weighted graph Laplacian as shown in Lemma 1(a). The Laplacian structure and connectedness of $\mathcal{G}$ also guarantee that D_k is positive definite because the diagonals of H_k must be positive. With our splitting, we can rewrite the Hessian as

$$H_k = D_k^{\frac{1}{2}} \left(I - D_k^{-\frac{1}{2}} B_k D_k^{-\frac{1}{2}} \right) D_k^{\frac{1}{2}}. \tag{14}$$

The Hessian pseudo-inverse is given by $H_k^\dagger = D_k^{-\frac{1}{2}} \left(I - D_k^{-\frac{1}{2}} B_k D_k^{-\frac{1}{2}} \right)^\dagger D_k^{-\frac{1}{2}}$. For the central term of this product we can use the Taylor's expansion identity $(I - X)^\dagger v = \left(\sum_{i=0}^\infty X^i\right) v$, which is valid for any vector v orthogonal to the unstable eigenvectors of X, [26, Chapter 5]. In our case the Laplacian structure of H and Assumption 1(b) guarantee that $-1 < \mu(D_k^{-\frac{1}{2}} B_k D_k^{-\frac{1}{2}}) \leq 1$, [33, Chapter 1] so we are restricted to the $n-1$ dimensional subspace orthogonal to $\mathbf{1}$. Fortunately g_k is orthogonal to $\mathbf{1}$ so we define the approximate Newton direction as a truncated Taylor expansion.

Definition 2. *We define the* ***Approximate Newton Direction***

$$d_k^{(N)} = -\sum_{r=0}^{N} D_k^{-\frac{1}{2}} \left(D_k^{-\frac{1}{2}} B_k D_k^{-\frac{1}{2}} \right)^r D_k^{-\frac{1}{2}} g_k \tag{15}$$

and the ***Approximate Hessian Inverse*** *on* $\mathbf{1}^\perp$

$$\bar{H}_k^{(N)} = \sum_{r=0}^{N} D_k^{-\frac{1}{2}} \left(D_k^{-\frac{1}{2}} B_k D_k^{-\frac{1}{2}} \right)^r D_k^{-\frac{1}{2}} \tag{16}$$

which naturally arises from the form of our approximate Newton direction.

The approximate Newton algorithm is obtained by replacing the Newton step d_k in (8) by its approximations $d_k^{(N)} = -\bar{H}_k^{(N)} g_k$. The resultant algorithm is characterized by the iteration

$$\lambda_{k+1} = \lambda_k - \alpha_k \bar{H}_k^{(N)} g_k. \tag{17}$$

The choice of N in (15) dictates how much information node i needs from the network to compute the ith element of the approximate Newton direction $d_k^{(N)}$ – recall that node i is associated with dual variable $[\lambda_k]_i$.

For the zeroth order approximation $d_k^{(0)}$ only the first term of the sum in (15) is considered and it therefore suffices to have access to the information in D_k to compute the approximate Newton step. Notice that the approximation in this case reduces to $d_k^{(0)} = D_k^{-1} g_k$ implying that we approximate H_k^{-1} by the inverse diagonals which coincides with the method in [21]. The first order approximation $d_k^{(1)}$ uses the first two terms of the sum in (15) yielding $d_k^{(1)} = \left(D_k^{-1} + D_k^{-1} B_k D_k^{-1}\right) g_k$. The key observation here is that the sparsity pattern of B_k; it is a weighted adjacency matrix. As a consequence, to compute the ith element of $d_k^{(1)}$ node i needs to collect information that is either locally available or available at nodes that share an edge with i. For the second order approximation $d_k^{(2)}$ we add the term $\left(D_k^{-1}B_k\right)^2 D_k^{-1}$ to the approximation $d_k^{(1)}$. The sparsity pattern of $\left(D_k^{-1}B_k\right)^2 D_k^{-1}$ is that of B_k^2, which has nonzero entries matching the 2-hop neighborhoods of each node. Therefore, to compute the ith element of $d_k^{(2)}$ node i requires access to information from neighboring nodes and from neighbors of these neighbors. In general, the Nth order approximation adds a term of the form $\left(D_k^{-1}B_k\right)^N D_k^{-1}$ to the $N-1$st order approximation. The sparsity pattern of this term is that of B_k^N, which coincides with the N-hop neighborhood, and computation of the local elements of the Newton step necessitates information from N hops away. We thus interpret (15) as a family of approximations indexed by N that yields Hessian approximations requiring information from N-hop neighbors in the network. This family of methods offers an explicit trade off between communication cost and precision of the Newton direction. We analyze convergence properties of these methods in the coming sections.

A. Basic properties

A basic guarantee for any iterative optimization algorithm is to show that it eventually approaches a neighborhood of the optimal solution. This is not immediate for ADD as defined by (17) because the errors in the $\bar{H}_k^{(N)}$ approximations to $H_k^{\dagger}$ may be significant. Notwithstanding, it is possible to prove that the $\bar{H}_k^{(N)}$ approximations are positive definite for all N and from there to conclude that the λ_k iterates in (17) eventually approach a neighborhood of the optimal λ^*. This claim is summarized in the following theorem, for proof see [30, Proposition A.24].

Theorem 1. *Let λ^* denote the optimal argument of the dual function $q(\lambda)$ of the optimization problem in (1) and consider the ADD-N algorithm characterized by iteration (17) with $\bar{H}_k^{(N)}$ as in (15). Assume $\alpha_k = \alpha$ for all k and that the network graph is not bipartite. Then, for all sufficiently small α,*

$$\lim_{k\to\infty} \lambda_k = \lambda^*. \tag{18}$$

By continuity of (4), convergence of the dual variable to an error neighborhood implies convergence of the primal variables to an error neighborhood. Theorem 1 is the weakest convergence proof we present but it is included because it serves as a benchmark for algorithm performance. Also, Theorem 1 uses a fixed step size which in many applications of interest is more practical than implementing a line search method.

Definition 3. *We define the* ***Newton Error*** *to be*

$$\epsilon_k = H_k d_k^{(N)} + g_k. \tag{19}$$

This is a measure of the deviation from true Newton direction because $H_k d_k = -g_k$ [cf. (9)] where d_k is the true Newton direction, which is equivalent to having $\epsilon_k = 0$ in (19). Therefore, the Newton step approximation error ϵ_k quantifies the deviation of the approximate Newton steps $d_k^{(N)}$ with respect to the actual Newton step d_k. An important property of ADD-N is that the deviation ϵ_k can be bounded for an arbitrary network and a given N. This fact is proven in the following lemma.

Lemma 2. *Define $\bar{\rho} \in (0,1)$ such that $\bar{\rho} > |\mu_{n-1}\left(B_k D_k^{-1}\right)|$ is a uniform upper bound on the second largest eigenvalue modulus of $B_k D_k^{-1}$. Then, the norms of the Newton approximation errors ϵ_k defined in (19) satisfy*

$$\|\epsilon_k\| \le \bar{\rho}^{N+1}\|g_k\|. \tag{20}$$

Proof: We begin eliminating the summation from our expression of the Newton error by observing that a telescopic property emerges.

$$\begin{aligned} H_k d_k^{(N)} + g_k &= H_k\left(-\sum_{i=0}^{N}\left(D_k^{-1}B_k\right)^i D_k^{-1}g_k\right) + g_k \\ &= \left(I - (D_k - B_k)\sum_{i=0}^{N}\left(D_k^{-1}B_k\right)^i D_k^{-1}\right) g_k \\ &= \left(I - \sum_{i=0}^{N}(B_kD_k^{-1})^i - (B_kD_k^{-1})^{i+1}\right) g_k \\ &= (B_kD_k^{-1})^{N+1} g_k \end{aligned}$$

We introduce the matrix $V \in \mathbb{R}^{n\times n-1}$, made up of $n-1$ orthonormal columns spanning $\mathbf{1}^{\perp}$. We observe that $VV' = I_n - \frac{\mathbf{11}'}{n}$, and since $g \in \mathbf{1}^{\perp}$ we have $g = VV'g$. Our descent occurs in $\mathbf{1}^{\perp}$ so we restrict our analysis to this subspace. Recall that $|\mu_{n-1}(X)|$ is the second largest eigenvalue modulus of the matrix X. We have $\|V'(B_kD_k^{-1})^{N+1}g_k\| = \|V'(B_kD_k^{-1})^{N+1}VV'g_k\| \le \|V'(B_kD_k^{-1})^{N+1}V\|\|V'g_k\| \le |\mu_{n-1}\left(B_kD_k^{-1}\right)|^{N+1}\|g_k\|$ from the triangle inequality. Lemma 1[b] tells us that $H_{ii} \le 1/m = n/\gamma$ by choosing v such that $v_i = 1$ and $v_j = 0$ for all $j \ne i$. Using the fact that $\sum_j B_{ij} = H_{ii}$ and the eigenvalue bound from [34] we have

$$|\mu_{n-1}\left(B_kD_k^{-1}\right)| \le 1 - \frac{\gamma}{n^2(1+\text{diam}(\mathcal{G}))} \tag{21}$$

where diam(G) is the diameter of the graph $\mathcal{G}$. and we Equation (21) shows by construction that there exists one such bound $\bar{\rho} \in (0,1)$, completing the proof. ■

Lemma 2 establishes the eigenvalue bound $\bar{\rho}$ as a key coefficient capturing the ability of information to spread through the network. Its appearance is natural because the accuracy of our local approximations to the Newton step depend on the network's ability to percolate information. Another key observation about Lemma 2 is that the Newton step approximation error is proportional to the norm of the dual gradient. Another way to interpret Lemma 2 is to observe that the relative Newton error $||\epsilon_k||/||g_k||$ is at worst a constant. Since the dual gradient norm $||g_k||$ tends to zero as we approach the dual optimum argument, the approximation error norm $||\epsilon_k||$ also approaches zero as iterations progress towards the optimum.

Analysis of the Newton's method takes advantage of the conditioning assumption in Assumption 1(a) by using the inverse eigenvalue bounds

$$m\, v'v \leq v'H(\lambda)^{\dagger}v \leq Mv'v \qquad \forall v \in \mathbf{1}^{\perp}. \tag{22}$$

Since we do not use the exact pseudo inverse we need to prove that these bounds hold for our approximate Hessian inverse.

Lemma 3. *The approximate inverse Hessian remains well conditioned in the subspace $\mathbf{1}^{\perp}$. I.e., for given ADD family index N and dual variable λ it holds that for all vectors $v \in \mathbf{1}^{\perp}$*

$$\frac{m}{2}\, v'v \leq v'\bar{H}^{(N)}(\lambda)v \leq Mv'v \tag{23}$$

for any λ where m and M are defined in Lemma 1(b).

Proof: Consider the definition of $\bar{H}_k^{(N)}$ in equation (16) along with the Taylor expansion of the pseudoinverse with v in the invertible subspace $\mathbf{1}^{\perp}$

$$v'H_k^{\dagger}v = v'\left(\bar{H}_k^{(N)} + \sum_{r=N+1}^{\infty} D_k^{-\frac{1}{2}}\left(D_k^{-\frac{1}{2}}B_kD_k^{-\frac{1}{2}}\right)^r D_k^{-\frac{1}{2}}\right)v. \tag{24}$$

Since each term $D_k^{-\frac{1}{2}}\left(D_k^{-\frac{1}{2}}B_kD_k^{-\frac{1}{2}}\right)^r D_k^{-\frac{1}{2}}$ is positive semidefinite due to our splitting choice in Definition 1 we have

$$v'H_k^{\dagger}v \geq v'\bar{H}_k^{(N)}v \tag{25}$$

and from (3) we achieve the desired upper bound. Again consider the definition of $\bar{H}_k^{(N)}$ in equation (16), this time removing the first term from the sum

$$v'\bar{H}_k^{(N)}v = v'D_k^{-1}v + v\left(\sum_{r=1}^{N} D_k^{-\frac{1}{2}}\left(D_k^{-\frac{1}{2}}B_kD_k^{-\frac{1}{2}}\right)^r D_k^{-\frac{1}{2}}\right)v. \tag{26}$$

Since each turn in the sum is positive semidefinite from our splitting choice in Definition 1 we have

$$v'\bar{H}_k^{(N)}v \geq v'D_k^{-1}v. \tag{27}$$

Recall that $[D_k]_{ii} = 2[H_k]_{ii}$. From Assumption 1(b) tells us $[H_k]_{ii} = \mathbb{I}(i)'H_k\mathbb{I}(i) \leq 1/m$ where $\mathbb{I}(i)$ is the indicator vector for node i. Inverting $[D_k]_{ii}^{-1} = 1/2[H_k]_{ii}^{-1} \geq m/2$ and subbing into (27), we recover the desired lower bound. ■

Lemma 3 guarantees uniform conditioning and strong convexity of our approximate Hessian inverse on the subspace in which we are descending. In fact, this property holds for vectors in the span of $\mathbf{1}$ as well but since we restrict our descent to the subspace $\mathbf{1}^{\perp}$, we only need it to hold within that subspace. This result guarantees that the conditioning of the approximate Hessian inverse is at most a factor of 2 worse than the conditioning of the Hessian itself.

B. Distributed backtracking line search

Algorithms ADD-N for different N differ in their information dependence. Our goal is to develop a family of distributed backtracking line searches parameterized by the same N and having the same information dependence. The idea is that the Nth member of the family of line searches is used in conjunction with the Nth member of the ADD family to determine the step and descent direction in (17). As with the ADD-N algorithm, implementing the distributed backtracking line search requires each node to get information from its N-hop neighbors.

Centralized backtracking line searches are typically intended as a method to find a stepsize α that satisfies Armijo's rule. This rule requires the stepsize α to satisfy the inequality

$$q(\lambda + \alpha d) \leq q(\lambda) + \sigma\alpha d'g, \tag{28}$$

for given descent direction d and search parameter $\sigma \in (0, 1/2)$. The backtracking line search algorithm is then defined as follows.

Algorithm 1. *Consider the objective function $q(\lambda)$ and given variable value λ_k and a descent direction d_k and dual gradient $g_k = \nabla q(\lambda_k)$. The backtracking line search algorithm is:*

```
Initialize $\alpha = 1$
while $q(\lambda_k + \alpha d_k) > q(\lambda_k) + \sigma\alpha d_k' g_k$
    $\alpha = \alpha\beta$
end
```

The scalars $\beta \in (0, 1)$ and $\sigma \in (0, 1/2)$ are given parameters.

This line search algorithm is commonly used with Newton's method because it guarantees a strict decrease in the objective and once in an error neighborhood it always selects $\alpha = 1$ allowing for quadratic convergence, [18, Section 9.5].

In order to create a distributed version of the backtracking line search we need a local version of the Armijo rule. We start by decomposing the dual objective $q(\lambda) = \sum_{i=1}^{n} q_i(\lambda)$ where the local objectives $q_i(\lambda)$ are defined as

$$q_i(\lambda) = \sum_{e=(j,i)} -\phi_e(x_e(\lambda)) + \lambda_i(a_i' x(\lambda) - b_i), \tag{29}$$

where the vector a_i' denotes the ith row of the incidence matrix A. Observe that according to (29) and the sparsity pattern of A the local objective $q_i(\lambda)$ depends only on the local dual variable λ_i and flow variables x_e for links adjacent to i.

Leveraging the definition in (29), we define an N-parameterized local Armijo rule.

Definition 4. *Define the* ***Local Armijo Rule*** *to be the condition that step size α_i must satisfy*

$$q_i(\lambda + \alpha_i d) \leq q_i(\lambda) + \sigma \alpha_i \sum_{j \in \mathcal{N}_i^{(N)}} d_j g_j, \tag{30}$$

for all i where $\mathcal{N}_j^{(N)}$ is the set of N-hop neighbors of node j, the scalar $\sigma \in (0, 1/2)$ is the same as in (28), $g = \nabla q(\lambda)$ and d is a descent direction.

Each node is able to compute its own step size α_i satisfying (30) using information from its N-hop neighborhood. we define a distributed backtracking line search according to the following algorithm.

Algorithm 2. *Given local objectives $q_i(\lambda_k)$ satisfying $\sum_i q_i(\lambda_k) = q(\lambda_k)$, descent direction d_k and dual gradient $g_k = \nabla q(\lambda_k)$.*

```
for $i = 1 : n$
    Initialize $\alpha_i = 1$
    while $q_i(\lambda + \alpha_i d) > q_i(\lambda) + \sigma \alpha_i \sum_{j \in \mathcal{N}_i^{(N)}} [d_k]_j [g_k]_j$
        $\alpha_i = \alpha_i \beta$
    end
end
```

The scalars $\beta \in (0, 1)$, $\sigma \in (0, 1/2)$ and $N \in \mathbb{Z}_+$ are given parameters.

The distributed backtracking line search described in Algorithm 2 works by allowing each node to execute its own modified version of Algorithm 1 using only information from N-hop neighbors. The α_i's generated by Algorithm 2 are not a traditional step size because in general it does not preserve the descent direction. To use Algorithm 2 with the ADD method we need to restate the ADD-N iteration as

$$\lambda_{k+1} = \lambda_k - \mathcal{A}_k \bar{H}_k^{(N)} g_k \tag{31}$$

where $\mathcal{A}_k$ is a diagonal matrix containing the steps α_i at time k. This would appear to undo the benefit of using an approximate Newton descent direction but in fact we are guaranteed to to make progress toward the optimal as long as we are outside a neighborhood of the optimal. Once inside that neighborhood $\mathcal{A}_k = I$ which recovers the update in (17) with a step size equal to $\alpha_k = 1$. These analytical results will be proven in the following section.

V. Convergence Rate

The basic guarantee in Theorem 1 is not stronger than convergence results for gradient descent. Our goal is to show that the approximate Newton method in (17) combined with the distributed line search in Algorithm 2 exhibits global convergence and local quadratic convergence once sufficiently close to the optimum. These properties are akin to corresponding properties of centralized (exact) Newton algorithms and are presented in the following theorem.

Theorem 2. *Consider an ADD-N algorithm with iterates λ_k as defined by (31) with approximate Newton step $H_k^{(N)} g_k$ as in (15) for given N. The step size $\mathcal{A}_k$ is selected according to the approximate backtracking line search defined in Algorithm 2 with parameters*

$$\sigma \in \left(0, \frac{1 + 2\bar{\rho}^{N+1}}{6}\right) \tag{32}$$

and $\beta \in (0, 1)$. Define the constant $\bar{\rho} > \rho\left(B_k D_k^{-1}\right) \in (0, 1)$ as a uniform upper bound on the largest eigenvalue modulus of the product of splitting matrices $D_k - B_k = H_k$. With the conditioning constants m and M and the Lipschitz constant L as defined in Lemma 1:

(i) Strict Decrease Phase. *For gradient at iteration k satisfying*

$$\|g_k\| > \eta := \frac{3(1 - 2\sigma)}{LM^2} \tag{33}$$

the dual objective is reduced by at least $\frac{1}{2}\beta\hat{\alpha} m N \eta^2$, i.e.,

$$q(\lambda_{k+1}) \leq q(\lambda_k) - \frac{1}{2}\beta\hat{\alpha} m N \eta^2. \tag{34}$$

(ii) Quadratic Phase. *For gradient at iteration k satisfying*

$$\frac{2\left(1 - \bar{\rho}^{N+1}\right)}{LM^2} < \|g_k\| \leq \frac{3(1 - 2\sigma)}{LM^2} \tag{35}$$

the gradient norm the ADD-N algorithm converges quadratically, i.e.,

$$\|g_{k+1}\| < \frac{LM^2}{2\left(1 - \bar{\rho}^{N+1}\right)} \|g_k\|^2. \tag{36}$$

(iii) Terminal Phase. *For gradient at iteration k satisfying*

$$\|g_k\| \leq \frac{2\left(1-\bar{\rho}^{N+1}\right)}{LM^2} \tag{37}$$

the ADD-N algorithm satisfies $\|g_{k+1}\| \leq \|g_k\|$ and thus remains in within the neighborhood. Further progress may be made but is not guaranteed.

According to Theorem 2, ADD-N algorithms exhibit three convergence phases with boundaries between them occurring over ranges of the norm of the dual gradient $\|g_k\|$. The first phase occurs when the gradient norm is greater than $3\left(1-2\sigma\right)/LM^2$, during which there is a strict decrease in the dual objective. This strict decrease guarantees that the second phase is reached. The second and third phases are characterized by a step size selection $\mathcal{A}_k = I$ [c.f. (31)] or equivalently $\alpha_k = 1$ [c.f. (17)]. Our restriction that the $\sigma \in (1, (1+2\bar{\rho}^{N+1})/6)$ is used to guarantee we do not skip the second phase and go straight to the third phase. During the second phase the residual, $||g_k||$ decreases quadratically implying that the accuracy of λ_k as an approximation of λ^* improves rapidly. The quadratic decrease in the residual during the second phase implies that the third and terminal phase is eventually reached. When the gradient norm reaches $2\left(1-\bar{\rho}^{N+1}\right)/LM^2$ convergence of ADD-N slows down to a linear rate or can in rare cases stop. The first two phases parallel corresponding phases for the canonical centralized Newton Method, [18, Chapter 9]. The third phase is unique to ADD-N. It corresponds to a situation in which the errors in the approximation $H_k^{(N)} g_k$ of the Newton step $H_k g_k$ become comparable to the value of the step itself. Notice that while the errors in the Newton step slow down the convergence rate, they generally do not halt the progress of ADD-N towards null residual.

The complete proof of Theorem 2 is presented in sections V-A and V-B. Section V-A starts with a generalization of the descent lemma [30, Section A.5] and analyzes the distributed backtracking line search of Section IV-B to prove the existence of the strict decrease phase of Part (i). In Section V-B we characterize the conditions for quadratic convergence, which we leverage to prove parts (ii) and (iii). These results do not require the distributed line search as they simply require a fixed stepize $\alpha = 1$ which in the case of the distributed line search is guaranteed.

Remark 1. Theorem 2 tells us that network structure affects the performance of ADD-N algorithms through the eigenvalue bound $\bar{\rho} > \rho\left(B_k D_k^{-1}\right)$ because the residual range during which ADD-N experiences quadratic convergence is lower bounded by $\|g_k\| < \bar{\rho}^{N+1}$. For small $\bar{\rho}$ this range is significant. As $\bar{\rho}$ approaches 1, however, this range shrinks thereby slowing the overall convergence rate. Thus, it is fair to say that ADD-N works well in networks with small $\bar{\rho}$ while networks with large $\bar{\rho}$ are difficult in that they require larger N to experience the same rate of convergence. From the bound in (21) we can infer topological conditions for this to happen. This bound approaches 1 if some of the following situations happen: (i) The number of nodes n is large. (ii) The graph diameter diam(G) is large. (iii) While not directly obvious from (21), its derivation from [34] indicates that $\bar{\rho}$ tends toward 1 when then network has large maximum degree, $\Delta(G)$.

Poor performance for large n and large diam(G) matches the intuition that convergence towards λ^* necessitates propagation of information through the network. Poor performance for large $\Delta(G)$ is counterintuitive but not incongruous with results on consensus on scale free networks, [35]. For a graph of fixed size n the network structure for which the bound in (21) is smallest is the one with the minimum product $\Delta(G)$diam(G) of maximum degree and diameter. Small world networks [36] have small degree and small diameter by design. It has already being observed that networks which have simultaneously low diameter and low maximum degree have desirable properties in engineered systems, such as fault tolerance and algebraic connectivity, [37].

A. Strict Decrease Phase – Proof of Theorem 2, part (i)

Traditional analysis of the centralized backtracking line search of Algorithm 1 leverages a lower bound on the stepsize α to prove strict decrease. We take a similar approach here and begin by finding a global lower bound on the stepsize $\hat{\alpha} \leq \alpha_i$ that holds for all nodes i. Before proceeding with the Lemma, we define some additional notation. The locally observable gradient vector at node i is given by

$$[\tilde{g}_k^{(i)}]_j = \begin{cases} [g_k]_j & \text{if } j \in \mathcal{N}_i^{(N)} \\ 0 & \text{else} \end{cases} \tag{38}$$

where $\mathcal{N}_i^{(N)}$ is the extended neighborhood of node i including all N hop neighbors. Further define the local update vector

$$\tilde{d}_k^{(i)} = -\bar{H}_k^{(N)} \tilde{g}_k^{(i)} \tag{39}$$

and finally define a reduced Hessian $\nabla^2 q_i(\lambda) = \tilde{H}^{(i)}(\lambda)$ by setting to zero the rows and columns corresponding to nodes outside of the neighborhood $\mathcal{N}_i^{(N)}$,

$$\left[\tilde{H}^{(i)}(\lambda)\right]_{sj} = \begin{cases} [H(\lambda)]_{sj} & \text{if } s,j \in \mathcal{N}_i^{(N)} \\ 0 & \text{else} \end{cases}. \tag{40}$$

Since the elements of H already satisfy $H_{ij} = 0$ for all $i, j \notin \mathcal{E}$ the resulting $\tilde{H}^{(i)}$ has the structure of a principal submatrix of H with the deleted rows left as zeros.

Lemma 4. *Consider the distributed line search in Algorithm 2 with parameter N, starting point $\lambda = \lambda_k$, and descent direction $d = d_k^{(N)} = -\bar{H}_k^{(N)} g_k$ computed by the ADD-N algorithm [cf.* (15)*]. The stepsize*

$$\hat{\alpha} = (1-\sigma)\frac{m^2}{M^2}$$

satisfies the local Armijo rule in (30)*, i.e.,*

$$q_i(\lambda_{k+1}) \leq q_i(\lambda_k) + \sigma\hat{\alpha} \sum_{j\in\mathcal{N}_i^{(N)}} [d_k]_j [g_k]_j$$

for all network nodes i and all k.

Proof: See Appendix B. ■

According to Lemma 4 we have

$$q_i(\lambda_{k+1}) - q_i(\lambda_k) \leq \hat{\alpha}\sigma \tilde{g}_k^{(i)\prime}\tilde{d}^{(i)}$$

because $\hat{\alpha}$ is a lower bound on α_i. Therefore, Algorithm 2 exits with $\alpha \in (\beta\hat{\alpha}, \hat{\alpha})$ and any $\alpha \leq \hat{\alpha}$ satisfies the exit condition in (30) therefore

$$q_i(\lambda_{k+1}) - q_i(\lambda_k) \leq \beta\hat{\alpha}\sigma \tilde{g}_k^{(i)\prime}\tilde{d}^{(i)}.$$

Applying Lemma 3 with the definition of $\tilde{d}^{(i)}$ we get

$$q_i(\lambda_{k+1}) - q_i(\lambda_k) \leq -\frac{\beta\hat{\alpha}\sigma m}{2}\|\tilde{g}_k^{(i)}\|^2. \tag{41}$$

Summing over all i and applying $\sum_{i=1}^n q_i(\lambda) = q(\lambda)$, we have

$$q(\lambda_{k+1}) - q(\lambda_k) \leq -\frac{\beta\hat{\alpha}\sigma m}{2}\sum_{i=1}^n \|\tilde{g}_k^{(i)}\|^2. \tag{42}$$

Using the definition of the 2-norm we can write $\sum_{i=1}^n \|\tilde{g}^{(i)}\|^2 = \sum_{i=1}\sum_{j\in\mathcal{N}_i^{(N)}} [g_k]_j^2$. Counting the appearance of each $[g_k]_j^2$ term in this sum we have that $\sum_{i=1}\sum_{j\in\mathcal{N}_i^{(N)}} [g_k]_j^2 = \sum_{i=1} |\mathcal{N}_i^{(N)}| [g_k]_i^2$. Since the network is connected it must be $|n_i^{(N)}| \geq N$, from which it follows $\sum_{i=1}\sum_{j\in n_i^{(N)}} [g_k]_j^2 \geq N\sum_{i=1}^n [g_k]_i^2$. Substituting this expression into (42) yields

$$q(\lambda_{k+1}) - q(\lambda_k) \leq -\frac{\beta\hat{\alpha}\sigma m N}{2}\sum_{i=1}^n [g_k]_i^2$$

Observe now that $\sum_{i=1}^n [g_k]_i^2 = \|g_k\|^2$ and substitute the lower bound $\eta < \|g_k\|$ to obtain the desired relation. This completes the proof of Theorem 2 part (i).

B. Quadratic and Terminal Convergence Phases – Proof of Theorem 2, parts (ii) and (iii)

The result in Theorem 2 part (i) guarantees global convergence into any error neighborhood $\|g_k\| \leq \eta$ around the optimal value because the dual objective is strictly decreasing by, at least, the noninfinitesimal quantity $\beta\hat{\alpha}\sigma m N\eta^2/2$ while we remain outside of this neighborhood. In particular, we are guaranteed to reach a point inside the neighborhood $\|g_k\| \leq \eta = 3(1-2\sigma)/(LM^2)$. Once this condition is met, we show that convergence towards the optimum is quadratic. The first step in proving that result is to show that step size $\mathcal{A}_k = I$ is always selected as claimed in the following lemma.

Lemma 5. *Consider the distributed line search in Algorithm 2 with parameter N, starting point $\lambda = \lambda_k$, and descent direction $d = d_k^{(N)} = -\bar{H}_k^{(N)} g_k$. If the search parameter σ is chosen such that*

$$\sigma \in \left(0, \frac{1}{2}\right)$$

and the norm of the dual gradient satisfies

$$\|g_k\| \leq \frac{3}{LM^2}(1-2\sigma),$$

then Algorithm 2 selects stepsize $\alpha_i = 1$ for all i.

Proof: See Appendix C. ■

To complete the proof we return to the ADD-N update in (17) and set $\mathcal{A}_k = 1$. With this restriction we derive a generalization of the descent lemma that allows us to capture the impact of the approximation of the Newton direction on the convergence rate.

Lemma 6. *Consider an ADD-N algorithm with iterates λ_k as defined by (17), approximate Newton step $H_k^{(N)} g_k$ as in (31), and stepsize $\mathcal{A}_k = I$. The norm of the dual gradient $\|g_k\|$ is reduced at each iteration according to the relation*

$$\|g_{k+1}\| \leq \frac{LM^2}{2}\|g_k\|^2 + \bar{\rho}^{N+1}\|g_k\| \tag{43}$$

where the Lipshitz constant, L and strict convexity coefficient, M are defined in Lemma 1 and $\bar{\rho}$ is a uniform bound eigenvalue bound defined in (21).

Proof: Consider the gradient norm $\|g_{k+1}\|$ at iteration $k+1$ and the definition of the Newton step approximation error $\epsilon_k = H_k d_k^{(N)} + g_k$ at iteration k as given in (19). We can then write

$$\|g_{k+1}\| = \|g_{k+1} + \epsilon_k - H_k d_k^{(N)} - g_k\|,$$

because the last three terms in the right hand side cancel each other out. Apply now the triangle inequality to write

$$||g_{k+1}|| \leq ||g_{k+1} - g_k - H_k d_k^{(N)}|| + ||\epsilon_k||.$$

The expression $||g_{k+1} - g_k - H_k d_k^{(N)}||$ can be rewritten as the integral $\| \int_0^1 \left(H(\lambda_k + t d_k^{(N)}) - H(\lambda_k) \right) d_k^{(N)} \|$ using the definitions of g_k and H_k as the gradient and Hessian of the dual object $q(\lambda_k)$ defined in (51). By Lipschitz continuity as required in Lemma 1, this expression is less than $L/2\|d_k^{(N)}\|^2$, yielding

$$||g_{k+1}|| \leq L/2\|d_k^{(N)}\|^2 + \|\epsilon_k\|.$$

To bound the first term in the expression above consider the definition of the approximate Newton step $d_k^{(N)} = -H_k^{(N)} g_k$ in (19) combined with (22) to get $\|d_k\| < M\|g_k\|$. We conclude that

$$||g_{k+1}|| \leq \frac{LM^2}{2}\|g_k\|^2 + \|\epsilon_k\|.$$

To complete the proof just observe that according to Lemma 2 the error norm $\|\epsilon_k\|$ is bounded above by $\bar{\rho}^{N+1}\|g_k\|$. ■

Lemma 6 shows that the reduction in the dual gradient at each iteration, (62) has a linear term and a quadratic term. The first (quadratic) term is the same term that appears in the analysis of Newton's method, [18, Chapter 9]. The second (linear) term corresponds to the error in the step approximation due to the truncation of the series in (15). Since $\bar{\rho} < 1$ the coefficient of the linear term can be made small with respect to that of the quadratic term for suitably selected N.

Parts (ii) and (iii) of Theorem 2 are simple characterizations of the values of the dual gradient $\|g_k\|$ for which the quadratic and linear terms dominate (62), respectively. Let us begin by characterizing the quadratic phase. Applying Lemma 6 we observe that when $2(1-\bar{\rho}^{N+1})/LM^2 < \|g_k\|$, we can rewrite the relation from Lemma 6 as

$$||g_{k+1}|| < \frac{LM^2}{2}\|g_k\|^2 + \frac{LM^2}{2}\frac{\bar{\rho}^{N+1}}{1-\bar{\rho}^{N+1}}\|g_k\|^2, \qquad (44)$$

Simplifying algebraically we have

$$||g_{k+1}|| < \frac{LM^2}{2(1-\bar{\rho}^{N+1})}\|g_k\|^2, \qquad (45)$$

which coincides with the relationship in (36) that corresponds to the claim in Part (ii) of Theorem 2. We claim quadratic convergence because substituting $2(1-\bar{\rho}^{N+1})/LM^2 < \|g_k\|$ into (45)we guarantee that that $||g_{k+1}|| < ||g_k||$. Finally in order guarantee the quadratic phase is realized we require that

$$\frac{2(1-\bar{\rho}^{N+1})}{LM^2} < \frac{3(1-2\sigma)}{LM^2}$$

which is achieved by limiting the line search parameter to $\sigma \in (0, 1/6 + \bar{\rho}^{N+1}/3)$.

For the terminal phase we consider $\|g_k\| \leq 2(1-\bar{\rho}^{N+1})/LM^2$ and subbing into (43) and simplifying we have

$$||g_{k+1}|| \leq \frac{LM^2}{2}\frac{2(1-\bar{\rho}^{N+1})}{LM^2}\|g_k\| + \bar{\rho}^{N+1}\|g_k\| = \|g_k\| \qquad (46)$$

which coincides with the relationship in Part (iii) of Theorem 2. We observe that the terminal phase does not guarantee further progress towards the optimal. It does guarantee that we stay within the neighborhood $\|g_k\| \leq 2(1-\bar{\rho}^{N+1})/LM^2$ but in practice we do not observe any cessation of progress in our numerical experiments; see Section VII.

VI. Consensus Implementation

We formulate ADD-N as a finite step analogue to the Consensus-Based Newton in [22] where we use the lazy random walk splitting in place of the splitting presented in that work. Defining a consensus scheme to solve the Newton equation, (9) we have the following update

$$d_k^{(r+1)} = D_k^{-1} B_k d_k^{(r)} - D_k^{-1} g_k \qquad (47)$$

where the splitting $H_k = D_k - B_k$ is the splitting from Definition 1. In this case we can limit communication costs by iteratively sharing information with 1-hop neighbors, rather than needing to send information to entire N-hop neighborhoods directly. Choosing the initial value to be $d_k^{(0)} = -D_k^{-1} g_k$ results in a sequence of approximations of the Newton direction as follows

$$\begin{aligned} d_k^{(0)} &= -D_k^{-1} g_k = -\bar{H}_k^{(0)} g_k \\ d_k^{(1)} &= D_k^{-1} B_k(-D_k^{-1} g_k) - D_k^{-1} g_k = -\bar{H}_k^{(1)} g_k \\ &\vdots \\ d_k^{(R)} &= -\sum_{r=0}^{R} D_k^{-\frac{1}{2}} (D_k^{-\frac{1}{2}} B_k D_k^{-\frac{1}{2}})^r D_k^{-\frac{1}{2}} g_k = -\bar{H}_k^{(R)} g_k \end{aligned}$$

We observe that after R consensus iterations our approximation $d_k^{(R)}$ is the same approximation arrived at by using ADD-N with $N = R$.

VII. Numerical results

We use numerical experiments to verify the practicality of the ADD-N algorithm. Our first key result is that the ADD-N algorithm requires significantly fewer iterations to converge than the gradient algorithm, [14] and can be executed with much smaller communication overhead than the consensus based Newton algorithm, [22]. The second is that the distributed line search method outlined in Algorithm 2 can be substituted for Algorithm 1 without loss of

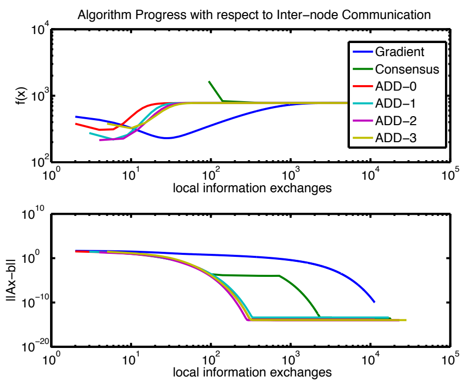


Fig. 1. Primal objective (top), $f(x_k)$ and primal feasibility (bottom), $\|Ax_k - b\|$ with respect to number of local information exchanges for a sample network optimization problem with 25 nodes and 75 edges. ADD converges an order of magnitude faster than consensus-based Newton and two orders of magnitude faster than gradient descent.

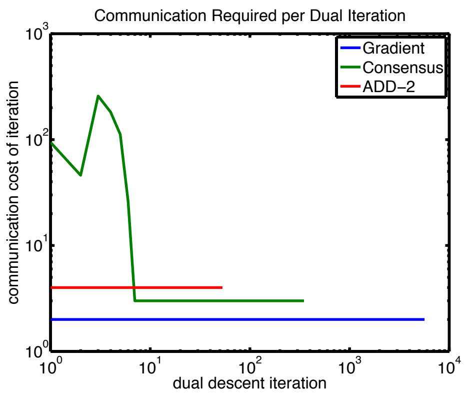


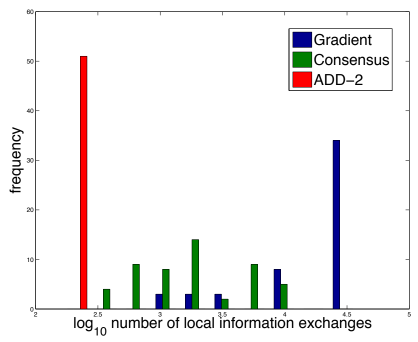
Fig. 2. The number of local information exchanges required per dual descent iteration is shown for sample network optimization problem with 25 nodes and 75 edges. ADD-N has a fixed local communication requirement per iteration equal to $N + 2$ which yields more consistent convergence rates with respect to communication requirements.

performance when using the ADD-N algorithm to select a descent direction. Finally, we consider the robust routing problem proposed in [7] and demonstrate that our framework not only solves this problem but that ADD-1 outperforms gradient descent by upwards of 2 orders of magnitude.

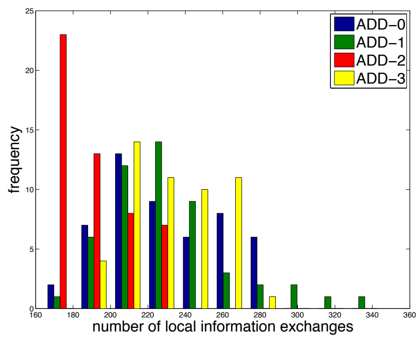
A. *Parameter Selection and Communication Overhead*

Figure 1 shows convergence metrics for a randomly generated network with 25 nodes and 75 edges. Edges in the network are selected uniformly at random. The flow vector b is chosen to place sources a full diam$(\mathcal{G})$ from the single sink. We use $\phi_e = \exp(-x_e)+\exp(x_e)$ as our objective function. We show results for ADD-0 through ADD-3, gradient descent, and consensus-based Newton. These algorithms are implemented with the same fixed stepsize $\alpha = 0.1$ in Figures 1-4. Different versions of ADD differ in the number of communication instances required per iteration. These numbers differ for consensus-based Newton and gradient descent as well. Fig. 1 demonstrates the algorithms' progress with respect to the number of times nodes exchange information with their 1-hop neighbors. We define one ”information exchange” as a transmit/receive event where each node i updates any subset of its local copies of variables belonging to its neighbors $j \in \mathcal{N}_i$. All versions of ADD are about an order of magnitude faster than consensus-based Newton and two orders orders of magnitude faster than gradient descent. As shown in Fig. 2, ADD-N has a fixed communication cost per iteration while consensus can require arbitrarily many communications. The number of communications per iteration is very large for the first few iterations of consensus-based Newton. It is clear that a major benefit of the ADD family is that unlike the consensus based Newton algorithm, precious communication resources are not wasted computing an extremely accurate Newton direction in early iterations when the return on this investment is minimal. Fig. 2 also shows that once the consensus Newton algorithm reaches its local phase it requires a very small number of dual iterations to remain below the error threshold. Intuitively, we expect the ADD-N algorithm to do as well as consensus Newton even with small N, once in the quadratic phase is reached. Another important conclusion of Fig. 1 is that even though increasing N in ADD decreases the number of iterations required, there is not a strict decrease in the number of communications. Indeed, as can be seen from Fig. 1, ADD-2 requires fewer communications than ADD-3. This fact demonstrates an inherent trade off between spending communication instances to refine the Newton step $d_k^{(N)}$ versus using them to take a step. We further examine this phenomenon in Fig. 3(b). These experiments are on random graphs with 25 nodes and 75 edges chosen uniformly at random. The flow vector b is selected by placing a source and a sink at diam$(\mathcal{G})$ away from each other. We consider an algorithm to have converged when its residual $\|g_k\| \leq 10^{-10}$.

Fig. 3(a) summarizes the the comparison between the ADD family and existing methods. Using ADD-N for any small N, is not only an order of magnitude faster than the next fastest approach, consensus-based Newton but it is significantly more consistent. This consistency is due to the fixed number of dual iterations needed to approximate



(a)



(b)

Fig. 3. **(a)** Histogram shows the number of local communications required to reach $\|g(\lambda_k)\| \leq 10^{-10}$ for gradient descent, consensus-based Newton and ADD-2 for 50 trials of the network optimization problem on random graphs with 25 nodes and 75 edges. ADD-2 converges faster and with much more consistency than gradient descent or consensus-based Newton. **(b)** Histogram of the number of local communications required to reach $\|g(\lambda_k)\| \leq 10^{-10}$ for ADD-N with respect to parameter N, for 50 trials of the network optimization problem on random graphs with 25 nodes and 75 edges. ADD-2 is shown to be the best on average by about 10 indicating that with respect to communication cost, larger N is not necessarily better.

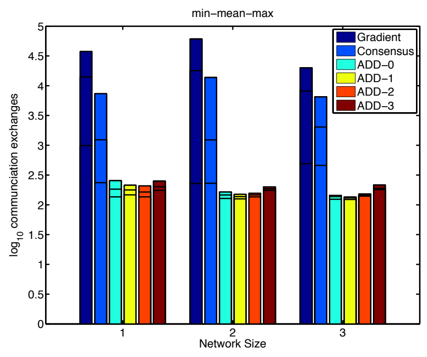


Fig. 4. Min, mean and max number of local communications required to reach $\|g(\lambda_k)\| \leq 10^{-10}$ for gradient descent, consensus-based Newton and ADD, computed for 35 trials each on random graphs with 25 nodes and 75 edges(1), 50 nodes and 350 edges(2), and 100 nodes and 1000 edges (3). The min and max are on the same order of magnitude for ADD, demonstrating small variance.

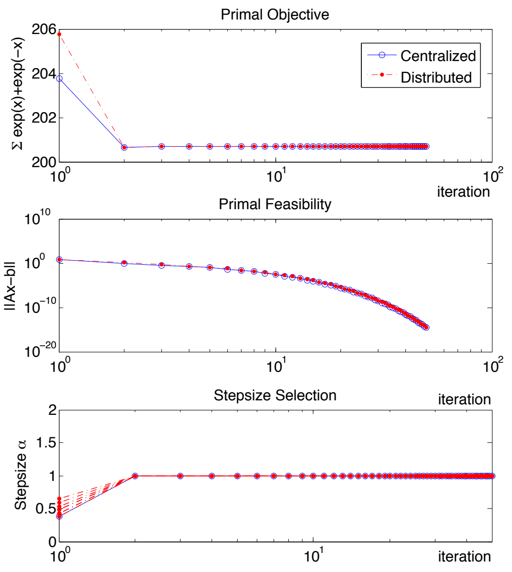


Fig. 5. The distributed line search results in solution trajectories nearly equivalent to those of the centralized line search. Top: the Primal Objective follows a similar trajectory in both cases. Middle: Primal Feasibility is achieved asymptotically. Bottom: unit stepsize is achieved for all nodes in the same number of steps it requires to achieve a global unit step size.

the Newton direction. It would be more correct to say the consensus-based Newton is inconsistent due to the wasted communication resources demonstrated by the dual iterations per primal iteration peak shown in Fig. 2. This inconsistency comes primarily from variations in the number of dual iterations required to approximate the Newton direction. The behavior of ADD is also explored for graphs of varying size and degree in Fig. 4. As the graph size increases the performance gap between ADD and competing methods increases. Consistency of ADD is also apparent since the maximum, minimum, and average information exchanges required to solve (1) for different network realizations are similar. This is not the case for consensus-based Newton or for gradient descent. Further note that ADD's communication cost increases only slightly with network size.

B. The distributed Line Search

We also use our numerical experiments to demonstrate that the distributed version of the backtracking line search is functionally equivalent to the centralized backtracking line search when the descent direction is chosen by the ADD method. Figure 5 shows an example of a network optimization problem with 25 nodes and 100

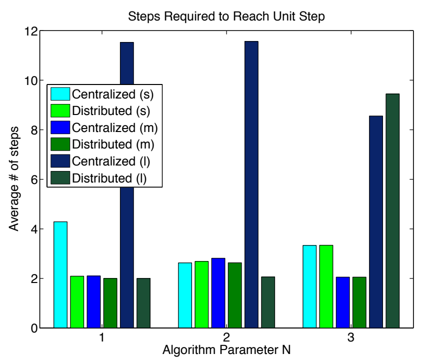


Fig. 6. The distributed line search reaches unit stepsize in 2 to 3 iterations. Fifty simulations were done for each algorithm with N=1, N=2 and N=3 and for Networks with 25 nodes and 100 edges (small), 50 nodes and 200 edges (medium) and 100 nodes and 400 edges (large).

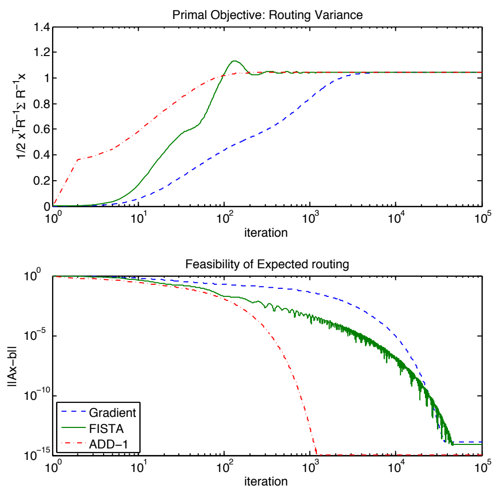


Fig. 7. The robust routing problem is solved efficiently by the ADD-1 algorithm, while the gradient descent method and FISTA method both require 10s of thousands of iterations. The top figure shows the total variance of the routing selected at iteration t. The bottom figure shows whether the current routing is feasible.

edges being solved using ADD-1 with the centralized and distributed backtracking line searches. The top plot shows that the trajectory of primal objective is not significantly affected by the choice line search. The middle plot shows that primal feasibility is approached asymptotically at the same rate for both algorithms. The bottom plot shows that a unit stepsize is achieved in the same number of steps despite the fact that in Algorithm 2 each node selects its own local line search parameter α_i. Thanks to Lemma 4 we are guaranteed strict local improvement eventually leading to local selection of step seize $\alpha_i = 1$ for all i which is equivalent to a global step size of $\alpha = 1$. Thus our implementation is truly distributed. In Figure 6 we look closer at the number of steps required to reach a unit stepsize. We compare the distributed backtracking line search to its centralized counterpart on networks with 25 nodes and 100 edges, 50 nodes and 200 edges and 100 nodes and 400 edges. For each network optimization problem generated we implemented distributed optimization using ADD-1, ADD-2, and ADD-3. Most trials required only 2 or 3 iterations to reach $\alpha = 1$ for both the centralized and distributed line searches. The variation came from the few trials which required significantly more iterations. As might be expected, increasing N causes the distributed and centralized algorithms to behave closer to each other. When we increase the size of the network most trials still only require 2 to 3 iterations to reach $\alpha = 1$ but for the cases which take more than 2 iterations we jump from around 10 iterations in the 25 nodes networks to around 40 iterations in 100 node networks.

C. The Robust Routing Problem

The robust routing problem is a network optimization problem focusing on selecting an optimal routing when links have uncertainties in their channel capacity, [7]. Each edge has a known variance σ_e and expected capacity r_e. The objective is to select a the flow variable $T_e \in [0, 1]$ that satisfies the conservation of constraint with the minimum total variance, when the expected flow arrivals are given by the vector b.

$$\min_{T_e \in [0,1]} \sum_e \sigma_e T_e^2 \qquad \text{s.t. } ART = b \tag{48}$$

We can recover the network flow problem by taking the local coordinate transform $x_e = R_e T_e$. Defining the diagonal matrices $\Sigma = \text{diag}(\sigma_e)$ and $R = \text{diag}(R_e)$ we can state the robust routing problem:

$$\min_{x_e \in [0,R_e]} x' R^{-1} \Sigma R^{-1} x \qquad \text{s.t. } Ax = b \tag{49}$$

We solve (49) by applying the ADD-1 algorithm and projecting the primal variables onto the interval $[0, R_e]$ during each primal update. While the effect of this projection is not considered analytically in this work, the effect of capacity constraints is discussed at length in [11]. Figure 7 shows a sample solution to (49) on a proximity network with 50 nodes, 224 edges. The matrix R is diagonal with values selected uniformly random on [0,1]. Sigma is diagonal with values are selected uniformly random on [0,10]. The vector b has a single sink with all other nodes being sources. This example emulates a wireless sensor network streaming data to a base station.

In our example, ADD-1 is compared to gradient descent and the fast iterative shrinking threshold algorithm (FISTA) presented in [38] and updated for distributed implementation in [39]. Distributed FISTA represents the state of the art for fast distributed gradient methods, when methods requiring network-wide message passing are excluded, see section I. In figure 7, FISTA is a significant improvement over gradient descent but it still requires an order of magnitude more iterations to reach feasibility $||Ax-b|| \leq 10^{-4}$ than ADD-1. Figure 8, the experiment is repeated 100 times and the progress of each algorithm is examined as a histogram after 100 and 1000 iterations. After 100 iterations ADD-1 had reached $||Ax-b|| \leq 10^{-4}$ for nearly 20% of the trials while neither gradient nor FISTA had reached that threshold for any trials. After 1000 iterations ADD-1 had reached feasibility with 60% at machine precision and over 95% smaller than 10^{-4}. Furthermore, relative error in the objective also had over 60% at machine precision and over 95% smaller than 10^{-4}. As with the ADD-1 method the error in the objective is on the same order as the feasibility, 10^{-5}. These experiments demonstrate that while FISTA significantly accelerates gradient descent, it is still fundamentally a first order method. The ADD-1 algorithm solves the robust routing problem an order of magnitude faster than FISTA.

VIII. Conclusion

A family of accelerated dual descent (ADD) algorithms to find optimal network flows in a distributed manner was introduced. Members of this family are characterized by a single parameter N determining the accuracy in the approximation of the dual Newton step. This same parameter controls the communication cost of individual algorithm iterations. We proved for any N there is a phase during which convergence toward the optimal is quadratic. ADD-1 and ADD-2 outperform gradient descent by two orders of magnitude and a related consensus-based Newton method by one order of magnitude.

This work has been extended into the stochastic case by [40] and to the capacity constrained case in [11]. We have also used the ADD as a foundation for Accelerated backpressure [13] which stabilizes queues in capacitated multi commodity communication networks with stochastic packet arrival rates. Finally, since the ADD algorithm approximates Newton's method without computing the inverse it could be investigated as a computationally efficient alternative.

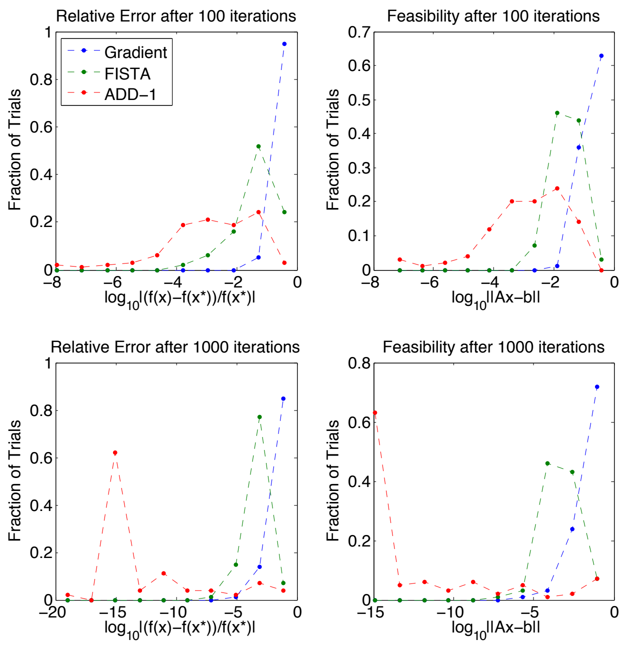


Fig. 8. Given 100 hundred trials of the robust routing problem, we find that after 100 iterations ADD-1 makes significantly more progress towards the optimal point than FISTA and gradient descent which have high values of infeasibility. After 1000 iterations ADD-1 was reached the optimal to machine precision in more than 60% of the trials while gradient descent and FISTA have at best a feasibility threshold of 10^{-5} and significant errors remaining in the objective value.

Appendix A – Proof of Lemma 1

(a) Consider given dual $\bar{\lambda}$ and primal $\bar{x} = x(\bar{\lambda})$ variables, and consider the second order approximation of the primal objective centered at the current primal iterates $\bar{x}$,

$$\hat{f}(y) = f(\bar{x}) + \nabla f(\bar{x})'(y-\bar{x}) + \frac{1}{2}(y-\bar{x})'\nabla^2 f(\bar{x})(y-\bar{x}). \quad (50)$$

We consider the optimization problem $\max_y -\hat{f}(y)$ subject to $Ay = b$, which is a quadratic maximization approximating (1). The dual of the approximate problem is also quadratic,

$$\min_{\lambda \in \mathbb{R}^n} \hat{q}(\lambda) = \min_{\lambda \in \mathbb{R}^n} \frac{1}{2}\lambda' A \nabla^2 f(\bar{x})^{-1} A' \lambda + p'\lambda + r. \quad (51)$$

The dual Hessian $\nabla^2 \hat{q}(\lambda) = A[\nabla^2 f(\bar{x})]^{-1}A'$ is computed by differentiating (51) twice with respect to λ. Finally, we observe that our approximation is exact the primal dual point $(\bar{x}, \bar{\lambda})$. That is $\hat{f}(\bar{x}) = f(\bar{x})$ and $\hat{q}(\bar{\lambda}) = q(\bar{\lambda})$. Since our approximation is constructed for arbitrary $\bar{\lambda}$, $\nabla^2 q(\bar{\lambda}) = A[\nabla^2 f(x(\bar{\lambda}))]^{-1}A'$recovers the desired relation. ■

(b) From part (a) we have $\nabla^2 q(\lambda) = A[\nabla^2 f(x(\lambda))]^{-1}A'$. We can get the lower bound by choosing $y = A'v$ to correspond with the eigenvector of $\nabla^2 f(x(\lambda))$ with the largest eigenvalue Γ defined in Assumption 2(a). Then $v'H(\lambda)v = y'[\nabla^2 f(x(\lambda))]^{-1}y \geq y'y\frac{1}{\Gamma}$. Since $v \in \mathbf{1}^{\perp}$, $y'y \geq \omega v'v$ thus $M = \Gamma/\omega$ where is ω is the algebraic connectivity defined in Assumption 1(a). Likewise we construct the lower bound by selecting $y = A'v$ to correspond with the eigenvector γ of $\nabla^2 f(x(\lambda))$ and get that $m = \gamma/n$ because $\mu_n(AA') \leq n$ holds from the fact that AA' is the unweighted graph Laplacian of $\mathcal{G}$. ∎

(c) From part (a) we have $\nabla^2 q(\lambda) = A[\nabla^2 f(x(\lambda))]^{-1}A'$. Considering a change of coordinates $y = A'v$ and the definition of the matrix norm as $||X|| = \max_{y\neq 0}||Xy||/||y||$ we have $||\nabla^2 q(\lambda) - \nabla^2 q(\bar\lambda)||$ upper bounded by $n||[\nabla^2 f(x(\lambda))]^{-1} - [\nabla^2 f(x(\bar\lambda))]^{-1}||$ because $\mu_n(AA') \leq n$ holds from the fact that AA' is the unweighted graph Laplacian of $\mathcal{G}$. Since $[\nabla^2 f(x(\lambda))]^{-1}$ is diagonal, the matrix norm $||[\nabla^2 f(x(\lambda))]^{-1} - \nabla^2 f(x(\bar\lambda))]^{-1}||$ reduces to $\max_e |1/\phi_e''(x(\lambda)) - 1/\phi_e''(x(\bar\lambda))|$. which is finite and positive from Assumption 2(a). Applying Assumption 2(b), we have

$$||\nabla^2 q(\lambda) - \nabla^2 q(\bar\lambda)|| \leq n\xi \max_e ||x_e(\lambda) - x_e(\bar\lambda)||. \tag{52}$$

We can differentiate equation (4) because $\gamma \leq \phi''(\cdot) \leq \Gamma$ which also guarantees us positivity and the upper bound

$$\left|\frac{\partial}{\partial\lambda_i}[\phi_e']^{-1}(\lambda)\right| = \frac{1}{\phi_e''([\phi_e']^{-1}(\lambda))} \leq \frac{1}{\gamma} \tag{53}$$

and in turn (53) guarantees that $x_e(\lambda)$ is Lipschitz continuous with constant $1/\gamma$ for each e. Applying this to (52) we have

$$||\nabla^2 q(\lambda) - \nabla^2 q(\bar\lambda)|| \leq \frac{n\xi}{\gamma}||\lambda - \bar\lambda|| \tag{54}$$

and we conclude that $\nabla^2 q(\lambda)$ is Lipschitz with constant $L = n\xi/\gamma$. Thus completing the proof. ∎

Appendix B – Proof of Lemma 4

From the mean value theorem centered at λ_k we can write the dual function's value as

$$q_i(\lambda_k + \alpha_i\tilde d_k^{(i)}) = q_i(\lambda_k) + \alpha_i\tilde g_k^{(i)\prime}\tilde d_k^{(i)} + \frac{\alpha_i^2}{2}\tilde d_k^{(i)\prime}\tilde H^{(i)}(z)\tilde d_k^{(i)}$$

where the vector $z = \lambda_k + t\alpha_i\tilde d_k^{(i)}$ for some $t \in (0,1)$; see e.g., [18, Section 9.1]. We use the relation $0 \preceq \tilde H^{(i)} \preceq H$ which follows from taking principle sub matrices of a positive semi-definite matrix and the bound $\|H^{\dagger}\| > m$ which follows from (22), to transform this equality into the bound

$$q_i(\lambda_k + \alpha_i\tilde d_k^{(i)}) \leq q_i(\lambda_k) + \alpha_i\tilde g_k^{(i)\prime}\tilde d_k^{(i)} + \frac{\alpha_i^2}{2m}\|\tilde d_k^{(i)}\|^2.$$

Introduce now a splitting of the term $\alpha_i\tilde g_k^{(i)}\tilde d_k^{(i)}$ to generate convenient structure for our upper bound on $q_i(\lambda_k + \alpha_i\tilde d_k^{(i)})$,

$$q_i(\lambda_k) + \alpha_i\sigma\tilde g_k^{(i)\prime}\tilde d_k^{(i)} + \alpha_i(1-\sigma)\tilde g_k^{(i)\prime}\tilde d_k^{(i)} + \frac{\alpha_i^2}{2m}\|\tilde d_k^{(i)}\|^2.$$

Further apply the definition of the local update vector $\tilde d_k^{(i)} = \bar H_k^{(N)}\tilde g_k^{(i)}$ and use the well-conditioning of the approximate inverse Hessian $\bar H_k^{(N)}$ as per Lemma 3 to claim that $m/2 \leq \|\bar H_k^{(N)}\| \leq M$ and define the upper bound on $q_i(\lambda_k + \alpha_i\tilde d_k^{(i)})$ as

$$q_i(\lambda_k) + \alpha_i\sigma\tilde g_k^{(i)\prime}\tilde d_k^{(i)} - \alpha_i(1-\sigma)\frac{m}{2}\|\tilde g_k^{(i)}\|^2 + \frac{\alpha_i^2M^2}{2m}\|\tilde g_k^{(i)}\|^2.$$

Factoring common terms in this upper bound yields

$$q_i(\lambda_k) + \alpha_i\sigma\tilde g_k^{(i)\prime}\tilde d_k^{(i)} + \frac{\alpha_i m}{2}\|\tilde g_k^{(i)}\|^2\left[-(1-\sigma) + \frac{\alpha_i M^2}{(m^2)}\right].$$

Substituting $\hat\alpha$ for α_i where $[-(1-\sigma)+\hat\alpha M^2/(m^2)] = 0$, the second term vanishes from this expression yielding the inequality

$$q_i(\lambda_k + \hat\alpha\tilde d_k^{(i)}) \leq q_i(\lambda_k) + \hat\alpha\sigma\tilde g_k^{(i)\prime}\tilde d_k^{(i)}.$$

Observing that $g_k^{(i)\prime}\tilde d_k^{(i)} = \sum_{j\in\mathcal{N}_i^{(N)}}[d_k]_j[g_k]_j$, the proof is completed. ∎

Appendix C – Proof of Lemma 5

Recall from (38), the local gradient $\tilde g_k^{(i)}$ is a sparse vector with nonzero elements $[\tilde g_k^{(i)}]_j = g_k^j$ for $j \in \mathcal{N}_i^{(N)}$. Due the fact that the local objective $q_i(\lambda)$ in (29) depends only on values in $\mathcal{N}_i^{(N)}$, we have $q_i(\lambda_k + \alpha d_k) = q_i(\lambda_k + \alpha\tilde d_k^{(i)})$ where α is a dummy variable for a_i. Applying the Lipschitz dual Hessian assumption to the local update vector $\tilde d_k^{(i)}$ we get

$$\|H(\lambda_k + \alpha\tilde d_k^{(i)}) - H(\lambda_k)\| \leq \alpha L\|\tilde d_k^{(i)}\|. \tag{55}$$

Also, recall the i-local version of the Hessian $\tilde H^{(i)}(\lambda)$ defined in (40). Since the elements of H already satisfy $H_{ij} = 0$ for all $i,j \notin \mathcal{E}$ the resulting $\tilde H^{(i)}$ has the structure of a principal submatrix of H with the deleted rows left as zeros. Since the norm $\|H(\lambda_k + \alpha\tilde d_k^{(i)}) - H(\lambda_k)\|$ in (55) is the maximum eigenvalue modulus of the matrix $H(\lambda_k + \alpha\tilde d_k^{(i)}) - H(\lambda_k)$, it is larger than the norm $\|\tilde H^{(i)}(\lambda_k + \alpha\tilde d_k^{(i)}) - \tilde H^{(i)}(\lambda_k)\|$ because the latter is the maximum over a subset of the eigenvalues of the former. Combining this observation with (55) yields

$$\|\tilde H^{(i)}(\lambda_k + \alpha\tilde d_k^{(i)}) - \tilde H^{(i)}(\lambda_k)\| \leq \alpha L\|\tilde d_k^{(i)}\|. \tag{56}$$

Interpret the dual update as a function of $\tilde q_i(\alpha)$ defined as

$$\tilde q_i(\alpha) := q_i(\lambda_k + \alpha\tilde d_k^{(i)}). \tag{57}$$

Differentiating with respect to α and using the definition of the local gradient $\tilde{g}_k^{(i)}$ we get the derivative of $\tilde{q}_i(\alpha)$ as

$$\tilde{q}_i'(\alpha) = \nabla q_i(\lambda_k + \alpha \tilde{d}_k^{(i)})\tilde{d}_k^{(i)} = \tilde{g}^{(i)}(\lambda_k + \alpha \tilde{d}_k^{(i)})\tilde{d}_k^{(i)}. \tag{58}$$

Differentiating with respect to α a second time and using the definition of $\tilde{H}^{(i)}$ in (40) yields

$$\tilde{q}_i''(\alpha) = \tilde{d}_k^{(i)\prime}\tilde{H}^{(i)}(\lambda_k + \alpha \tilde{d}_k^{(i)})\tilde{d}_k^{(i)}. \tag{59}$$

Return now to (56) and replace the matrix norm on the right hand side with left and right multiplication by the unit vector $\tilde{d}_k^{(i)}/\|\tilde{d}_k^{(i)}\|$. This yields $\tilde{d}_k^{(i)\prime}\left[\tilde{H}^{(i)}(\lambda_k + \alpha \tilde{d}_k^{(i)}) - \tilde{H}^{(i)}(\lambda_k)\right]\tilde{d}_k^{(i)} \leq \alpha L\|\tilde{d}_k^{(i)}\|^3$. Applying the derivatives $\tilde{q}_i''(\alpha)$ in (59), we can simplify to $\tilde{q}_i''(\alpha) - \tilde{q}_i''(0) \leq \alpha L\|\tilde{d}_k^{(i)}\|^3$. Integrating the above expression with respect to α results in

$$\tilde{q}_i'(\alpha) - \tilde{q}_i'(0) \leq \frac{\alpha^2}{2}L\|\tilde{d}_k^{(i)}\|^3 + \alpha \tilde{q}_i''(0),$$

which upon a second integration with respect to α yields

$$\tilde{q}_i(\alpha) - \tilde{q}_i(0) \leq \frac{\alpha^3}{6}L\|\tilde{d}_k^{(i)}\|^3 + \frac{\alpha^2}{2}\tilde{q}_i''(0) + \alpha\tilde{q}_i'(0).$$

Substitute $\alpha = 1$ and the definitions of the derivatives $\tilde{q}_i'(0)$ and $\tilde{q}_i''(0)$ given in (58) and (59) to get

$$\tilde{q}_i(1) - \tilde{q}_i(0) \leq \frac{L}{6}\|\tilde{d}_k^{(i)}\|^3 + \frac{1}{2}\tilde{d}_k^{(i)\prime}\tilde{H}^{(i)}(\lambda_k)\tilde{d}_k^{(i)} + \tilde{g}_k^{(i)\prime}\tilde{d}_k^{(i)}.$$

According to (40) the reduced Hessian $\tilde{H}^{(i)}$ has the structure of a principal submatrix of the Hessian H and $H \succeq 0$ it follows that $0 \preceq \tilde{H}^{(i)} \preceq H$ and that as a consequence

$$\tilde{d}_k^{(i)\prime}\tilde{H}^{(i)}(\lambda_k)\tilde{d}_k^{(i)} \leq \tilde{d}_k^{(i)\prime}H_k\tilde{d}_k^{(i)}.$$

Incorporating the local update $\tilde{d}_k^{(i)} = \bar{H}_k^{(N)}\tilde{g}_k^{(i)}$ we obtain the upper bound for local objective improvement $\tilde{q}_i(1) - \tilde{q}_i(0) \leq$

$$\frac{L}{6}\|\bar{H}_k^{(N)}\tilde{g}_k^{(i)}\|^3 + \frac{1}{2}\left(\bar{H}_k^{(N)}\tilde{g}_k^{(i)}\right)' H_k\bar{H}_k^{(N)}\tilde{g}_k^{(i)} - \tilde{g}_k^{(i)\prime}\bar{H}_k^{(N)}\tilde{g}_k^{(i)}.$$

Recall the sparsity pattern of the local gradient $\tilde{g}_k^{(i)}$ to write

$$-\tilde{g}_k^{(i)}\bar{H}_k^{(N)}\tilde{g}_k^{(i)} = \sum_{j\in\mathcal{N}_i^{(N)}}[g_k]_j[d_k]_j, \tag{60}$$

and split the right hand side of (60) to generate

$$\sum_{j\in\mathcal{N}_i^{(N)}}[g_k]_j[d_k]_j = \sum_{j\in\mathcal{N}_i^{(N)}}\sigma[g_k]_j[d_k]_j + (1-\sigma)[g_k]_j[d_k]_j. \tag{61}$$

Substitute now (61) into (60) and the result updates our upper bound $\tilde{q}_i(1) - \tilde{q}_i(0) \leq \frac{L}{6}\|\bar{H}_k^{(N)}\tilde{g}_k^{(i)}\|^3 + \frac{1}{2}\tilde{d}_k^{(i)\prime}H\tilde{d}_k^{(i)} + \sigma\sum_{j\in\mathcal{N}_i^{(N)}}[g_k]_j[d_k]_j + (1-\sigma)\sum_{j\in\mathcal{N}_i^{(N)}}[g_k]_j[d_k]_j$. Using the expression for the quadratic form in (60) to substitute the last term in the previous equation yields

$$\begin{aligned}\tilde{q}_i(1) - \tilde{q}_i(0) \leq\ & \frac{L}{6}\|\bar{H}_k^{(N)}\tilde{g}_k^{(i)}\|^3 + \frac{1}{2}\tilde{d}_k^{(i)\prime}H\tilde{d}_k^{(i)} \\ &+ \sigma\sum_{j\in\mathcal{N}_i^{(N)}}[g_k]_j[d_k]_j - (1-\sigma)\tilde{g}_k^{(i)\prime}\bar{H}_k^{(N)}\tilde{g}_k^{(i)}\end{aligned} \tag{62}$$

We consider the first term in (62) rewriting $||\bar{H}_k^{(N)}\tilde{g}_k^{(i)}||^2 = \tilde{g}_k^{(i)\prime}\left(\bar{H}_k^{(N)}\right)^2\tilde{g}_k^{(i)}$ since $\bar{H}_k^{(N)}$ is symmetric. A change of coordinates $y = \left(\bar{H}_k^{(N)}\right)^{1/2}\tilde{g}_k^{(i)}$ combined with the upper bound from Lemma 3 reveals that $||\bar{H}_k^{(N)}\tilde{g}_k^{(i)}||^3 \leq M^2\tilde{g}_k^{(i)\prime}\bar{H}_k^{(N)}\tilde{g}_k^{(i)}$ which subbed into (62) gives us

$$\begin{aligned}\tilde{q}_i(1) - \tilde{q}_i(0) \leq\ & \frac{LM^2}{6}\tilde{g}_k^{(i)\prime}\bar{H}_k^{(N)}\tilde{g}_k^{(i)} + \frac{1}{2}\tilde{d}_k^{(i)\prime}H\tilde{d}_k^{(i)} \\ &+ \sigma\sum_{j\in\mathcal{N}_i^{(N)}}[g_k]_j[d_k]_j - (1-\sigma)\tilde{g}_k^{(i)\prime}\bar{H}_k^{(N)}\tilde{g}_k^{(i)}.\end{aligned} \tag{63}$$

We consider the second term in (63) and note that from the definition of $\tilde{d}^{(i)}$ it follows that $\tilde{d}_k^{(i)\prime}H_k\tilde{d}_k^{(i)} = \tilde{g}_k^{(i)\prime}\bar{H}_k^{(N)}H_k\bar{H}_k^{(N)}\tilde{g}_k^{(i)}$. Applying the telescoping sum trick that we applied in the proof of Lemma 2 we have $\bar{H}_k^{(N)}H_k\bar{H}_k^{(N)} = \bar{H}_k^{(N)}\left(I - (B_kD^{-1})^{N+1}\right)$. By positive semi definiteness of $B_kD_k^{-1}$ we can conclude that $\tilde{d}_k^{(i)\prime}H_k\tilde{d}_k^{(i)} \leq \tilde{g}_k^{(i)\prime}\bar{H}_k^{(N)}\tilde{g}_k^{(i)}$ and subbing into (63) we have

$$\begin{aligned}\tilde{q}_i(1) - \tilde{q}_i(0) \leq\ & \frac{LM^2}{6}\tilde{g}_k^{(i)\prime}\bar{H}_k^{(N)}\tilde{g}_k^{(i)} + \frac{1}{2}\tilde{g}_k^{(i)\prime}H\tilde{g}_k^{(i)} \\ &+ \sigma\sum_{j\in\mathcal{N}_i^{(N)}}[g_k]_j[d_k]_j - (1-\sigma)\tilde{g}_k^{(i)\prime}\bar{H}_k^{(N)}\tilde{g}_k^{(i)}.\end{aligned} \tag{64}$$

Reorganizing terms we have $\tilde{q}_i(1) - \tilde{q}_i(0) \leq$

$$\sigma\sum_{j\in\mathcal{N}_i^{(N)}}[g_k]_j[d_k]_j + \tilde{g}_k^{(i)\prime}\bar{H}_k^{(N)}\tilde{g}_k^{(i)}\left[-(1-\sigma) + \frac{LM^2}{6}\|\tilde{g}_k^{(i)}\| + \frac{1}{2}\right].$$

Use $\|\tilde{g}_k^{(i)}\| \leq \|g_k\| \leq 3(1-2\sigma)/(LM^2)$ to write $\tilde{q}_i(1) - \tilde{q}_i(0) \leq$

$$\sigma\sum_{j\in\mathcal{N}_i^{(N)}}[g_k]_j[d_k]_j + M\|\tilde{g}_k^{(i)}\|^2\left[-(1-\sigma) + \frac{1}{2} - \sigma + \frac{1}{2}\right].$$

The bracketed portion simply sums to zero. Thus we have $\tilde{q}_i(1) - \tilde{q}_i(0) \leq \sigma\sum_{j\in\mathcal{N}_i^{(N)}}[g_k]_j[d_k]_j$. Substituting the definition of $\tilde{q}_i(\lambda)$ in (57) into this equation we arrive at $q_i\left(\lambda_k + d_k^{(i)}\right) \leq q_i(\lambda_k) + \sigma\sum_{j\in\mathcal{N}_i^{(N)}}[d_k]_j[g_k]_j$ which means that the exit condition (30) in Algorithm 2 is met with $\alpha_i = 1$ for all nodes i. Therefore all nodes exit Algorithm 2 with $\alpha_i = 1$ from which it follows that the selected stepsize is $\mathcal{A}_k = I$. ∎

REFERENCES

[1] D. P. Bertsekas. *Network Optimization: Continuous and Discrete Models*. Athena Scientific, Belmont, MA, 1998.

[2] R. T. Rockafellar. *Network Flows and Monotropic Programming*. Wiley, New York, NY, 1984.

[3] J. B. Orlin. Minimum convex cost dynamic network flows. *MATHEMATICS OF OPERATIONS RESEARCH*, May 1984.

[4] E. Miandoabchi R. Farahani. *Graph Theory for Operations Research and Management*. IGI Global, 2012.

[5] A. V. Goldberg, E. Tardos, and R. E. Tarjan. *Network Flow Algorithms*. Springer-Verlag, Berlin, 1990.

[6] V. Kolmogorov and A. Shioura. New algorithms for the dual of the convex cost network flow problem with application to computer vision. *Mathematical Programming*, 2007.

[7] Yuchen Wu, Alejandro Ribeiro, and Georgios B. Giannakis. Robust routing in wireless multi-hop networks. *Proc. Conf. on Info. Sciences and Systems*, 2007.

[8] Lin Xiao, M. Johansson, and S.P. Boyd. Simultaneous routing and resource allocation via dual decomposition. *IEEE Transactions on Communications*, 52(7):1136–1144, 2004.

[9] L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Transactions on Automatic Control*, 37:1936–1948, 1992.

[10] M. J. Neely. *Stochastic Network Optimization with Application to Communication and Queueing Systems*. Morgan and Claypool, 2010.

[11] M. Zargham, A. Ribeiro, and A. Jadbabaie. Accelerated dual descent for constrained convex network flow optimization. *Proceedings of IEEE CDC*, 2013.

[12] J. Sun and H. Kuo. Applying a newton method to strictly convex separable network quadratic programs. *SIAM Journal of Optimization*, 8, 1998.

[13] M. Zargham, A. Ribeiro, and A. Jadbabaie. Accelerated backpressure algorithm. *Proceedings of the IEEE Global Communications Conference*, 2013.

[14] A. Nedic and A. Ozdaglar. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54, 2009.

[15] I. Lobel and A. Ozdaglar. Distributed subgradient methods for convex optimization over random networks,. *IEEE Transactions on Automatic Control*, 56, 2011.

[16] N. Z. Shor. *Minimization Methods for Non-differentiable Functions*. Springer, 1985.

[17] Dimitri P. Bertsekas and Didier El Baz. Distributed asynchronous relaxation methods for convex network flow problems. *SIAM Journal of Optimization and Control*, 1987.

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

[19] Bertsekas and Gafni. Projected newton methods and optimization of multi-commodity flow. *IEEE Transactions on Automatic Control*, 28:1090–1096, 1983.

[20] J. G. Klincewicz. A newton method for convex separable network flow problems. *Bell Laboratories*, 1983.

[21] S. Authuraliya and S. Low. Optimization flow control with newton-like algorithm. *Telecommunications Systems*, 15:345–358, 2000.

[22] A. Jadbabaie, A. Ozdaglar, and M. Zargham. A distributed newton method for network optimization. In *Proceedings of IEEE CDC*, 2009.

[23] Paul Tseng. On accelerated proximal gradient methods for convex-concave optimization. *SIAM Journal on Optimization*.

[24] Yousef Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, 2003.

[25] David Watkins. *The Matrix Eigenvalue Problem*. SIAM, 2007.

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

[27] M. Zargham, A. Ribeiro, and A. Jadbabaie. A distributed line search for network optimization. In *Proceedings of IEEE ACC*, 2012.

[28] C. T. Kelley. Solving nonlinear equations with newton's method. *SIAM Fundamentals of Algorithms*, 1, 2003.

[29] F. J. Bonnans, C.J. Gilbert, C. Lemarchal, and C. A. Sagastizbal. *Numerical optimization: Theoretical and practical aspects*. Springer-Verlag, Berlin, Germany, 2006.

[30] D.P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Cambridge, Massachusetts, 1999.

[31] A. Nedić and A. Ozdaglar. Approximate primal solutions and rate analysis for dual subgradient methods. *SIAM Journal on Optimization, forthcoming*, 2008.

[32] A. Nedić and A. Ozdaglar. Subgradient methods in network resource allocation: Rate analysis. In *Proc. of CISS*, 2008.

[33] Fan Chung. *Spectral Graph Theory*. The American Mathematical Society, 1997.

[34] H. Landau and A. Odlyzko. Bounds for eigenvalues of certain stochastic matrices. *Linear Algebra and its Applications*, 38:5–15, 1981.

[35] M. E. Newman. The structure and function of complex networks. *SIAM Review*, 45:167–256, 2003.

[36] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, 1998.

[37] R. Olfati-Saber. Ultrafast consensus in small-world networks. In *Proceedings of IEEE ACC*, 2005.

[38] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[39] A.I. Chen and A. Ozdaglar. A fast distributed proximal-gradient method. *Proc. of Allerton Conference on Communication, Control, and Computing*, 2012.

[40] M. Zargham, A. Ribeiro, and A. Jadbabaie. Network optimization under uncertainty. *Proceedings of IEEE CDC*, 2012.