2011 American Control Conference
on O'Farrell Street, San Francisco, CA, USA
June 29 - July 01, 2011

Accelerated Dual Descent for Network Optimization

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

***Abstract*—Dual descent methods are commonly used to solve network optimization problems because their implementation can be distributed through the network. However, their convergence rates are typically very slow. This paper introduces a family of dual descent algorithms that use approximate Newton directions to accelerate the convergence rate of conventional dual descent. These approximate directions can be computed using local information exchanges thereby retaining the benefits of distributed implementations. The approximate Newton directions are obtained through matrix splitting techniques and sparse Taylor approximations of the inverse Hessian. We show that, similarly to conventional Newton methods, the proposed algorithm exhibits superlinear convergence within a neighborhood of the optimal value. Numerical analysis corroborates that convergence times are between one to two orders of magnitude faster than existing distributed optimization methods.**

I. Introduction

Conventional approaches to network optimization are based on subgradient descent in either the primal or dual domain; see, e.g., [6], [9], [12], [18]. For many classes of problems, subgradient descent algorithms yield iterations that can be implemented through distributed updates based on local information exchanges. However, practical applicability of the resulting algorithms is limited by slow convergence rates. To overcome this limitation Newton methods can be used, but this would require computation which cannot be accomplished through local information exchanges. This issue is solved in this paper through the introduction of a family of approximations to the Newton step.

The particular problem we consider is the network flow problem. Network connectivity is modeled as a directed graph and the goal of the network is to support a single information flow specified by incoming rates at an arbitrary number of sources and outgoing rates at an arbitrary number of sinks. Each edge of the network is associated with a concave function that determines the cost of traversing that edge as a function of flow units transmitted across the link. Our objective is to find the optimal flows over all links. Optimal flows can be found by solving a concave optimization problem with linear equality constraints (Section II). In particular, the use of subgradient descent in the dual domain allows the development of a distributed iterative algorithm. In this distributed implementation nodes keep track of variables associated with their outgoing edges and undertake updates based on their local variables and variables available at adjacent nodes (Section II-A). Distributed implementation is appealing because it avoids the cost and fragility of collecting all information at a centralized location. However, due to low convergence rates of subgradient descent algorithms, the number of iterations necessary to find optimal flows is typically very large [13], [15]. The natural alternative is the use of second order Newton's methods, but they cannot be implemented in a distributed manner (Section II-B).

Indeed, implementation of Newton's method necessitates computation of the inverse of the dual Hessian and a distributed implementation would require each node to have access to a corresponding row. It is not difficult to see that the dual Hessian is in fact a weighted version of the network's Laplacian and that as a consequence its rows could be locally computed through information exchanges with neighboring nodes. Its inversion, however, requires global information. Our insight is to consider a Taylor's expansion of the inverse Hessian, 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 on local information only – which, incidentally, coincides with the method of Hessian diagonal inverses proposed in [1]. The first order approximation necessitates information available at neighboring nodes and in general, the Nth order approximation necessitates information from nodes located N hops away (Section III). The resultant family of algorithms, denoted ADD-N permits a tradeoff between accurate Hessian approximation and communication cost. Despite the fact that the proposed distributed algorithms rely on approximate Newton directions, we show that they exhibit local quadratic convergence as their centralized counterparts (Section IV). An approximate backtracking line search is added to the basic algorithm to ensure global convergence (Section IV-B).

Newton-type methods for distributed network optimization have been recently proposed in [1], [8], [19]. While specifics differ, these papers rely on consensus iterations to compute approximate Newton directions. Quite surprisingly, it is possible to show that the methods in [1], [8], [19] and an ADD-N algorithm proposed here are equivalent under some conditions. Numerical experiments study the communication cost of ADD relative to [1], [8], [19] and to conventional subgradient descent. ADD reduces this cost by one order of magnitude with respect to [1], [8], [19] and by two orders of magnitude with respect to subgradient descent (Section VI). Our work is related to [2], [10] which apply approximate conjugate gradient methods to achieve superlinear convergence rates. [2] is a centralized algorithm and [10] uses a Taylor

This research is supported by Army Research Lab MAST Collaborative Technology Alliance, AFOSR complex networks program, ARO P-57920-NS, NSF CAREER CCF-0952867, and NSF CCF-1017454, ONR MURI N000140810747 and NSF-ECS-0347285.

[†]Michael Zargham, Alejandro Ribeiro and Ali Jadbabaie are with the Department of Electrical and Systems Engineering, University of Pennsylvania.

[‡] Asuman Ozdaglar is with the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology.

expansion motivated algorithm but lacks formal proofs.

II. Network Optimization

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 ith component of vector x is denoted as x^i. The notation $x \geq 0$ means that all components $x^i \geq 0$. 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 $\sum_{i=1}^n b^i = 1$. 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 implies that it must be $Ax = b$, with A the $n \times E$ node-edge incidence matrix defined as

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

The element in the ith row and jth column of a matrix A is written as $[A]_{ij}$. The transpose of A is denoted as A'. We define the reward as the negative of scalar cost function $\phi_e(x^e)$ denoting the cost of x^e units of flow traversing edge e. We assume that the cost functions ϕ_e are strictly convex and twice continuously differentiable. The max reward network optimization problem is then defined as

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

Our goal is to investigate Newton-type iterative distributed methods for solving the optimization problem in (1). Before doing that, let us discuss the workhorse distributed solution based on dual subgradient descent (Section II-A) and the conventional centralized Newton's method (Section II-B).

A. Dual Subgradient Method

Dual subgradient descent solves (1) by descending in the dual domain. Start then by defining the Lagrangian function of problem (1) as $\mathcal{L}(x, \lambda) = -\sum_{e=1}^{E} \phi_e(x^e) + \lambda'(Ax - b)$ and the dual function $q(\lambda)$ as

$$\begin{aligned} q(\lambda) &= \sup_{x\in\mathbb{R}^E} \mathcal{L}(x, \lambda) = \sup_{x\in\mathbb{R}^E} \left(-\sum_{e=1}^{E} \phi_e(x^e) + \lambda' Ax \right) - \lambda' b \\ &= \sum_{e=1}^{E} \sup_{x^e\in\mathbb{R}} \Big(-\phi_e(x^e) + (\lambda' A)^e x^e \Big) - \lambda' b, \end{aligned} \tag{2}$$

where in the last equality we wrote $\lambda' Ax = \sum_{e=1}^{E} (\lambda' A)^e x^e$ and exchanged the order of the sum and supremum operators.

It can be seen from (2) that the evaluation of the dual function $q(\lambda)$ decomposes into E one-dimensional optimization problems that appear in the sum. We assume that each of these problems has an optimal solution, which is unique because of the strict convexity of the functions ϕ_e. Denote this unique solution as $x^e(\lambda)$ and use the first order optimality conditions for these problems in order to write

$$x^e(\lambda) = (\phi_e')^{-1}(\lambda^i - \lambda^j), \tag{3}$$

where $i \in \mathcal{N}$ and $j \in \mathcal{N}$ respectively denote the source and destination nodes of edge $e = (i, j)$. As per (3) the evaluation of $x^e(\lambda)$ for each node e is based on local information about the edge cost function ϕ^e and the dual variables of the incident nodes i and j.

The dual problem of (1) is defined as $\min_{\lambda\in\mathbb{R}^n} q(\lambda)$. The dual function is convex because all dual functions of minimization problems are, and differentiable because the ϕ_e functions are strictly convex. Being convex and differentiable, the dual problem can be solved using gradient descent. Consider an iteration index k, an arbitrary initial vector λ_0 and define iterates λ_k generated by the following:

$$\lambda_{k+1} = \lambda_k - \alpha_k g_k \qquad \text{for all } k \geq 0, \tag{4}$$

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 (3) with $\lambda = \lambda_k$, [3, Section 6.4]. Differentiability of $g(\lambda)$ follows from strict convexity of (1). 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

$$g_k^i = \sum_{e=(i,j)} x^e(\lambda_k) - \sum_{e=(j,i)} x^e(\lambda_k) - b_i \tag{5}$$

The algorithm in (4) lends itself to distributed implementation. Each node i maintains information about its dual iterates λ_k^i and primal iterates $x^e(\lambda_k)$ of outgoing edges $e = (i, j)$. Gradient components g_k^i are evaluated as per (5) 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 (4). Having updated the dual iterates, we proceed to update primal variables as per (3). This update necessitates local multipliers λ_k^i and neighboring multipliers λ_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., [13], [15]. This motivates consideration of Newton's method which we describe next.

B. Newton's Method for Dual Descent

Newton's Method is a descent algorithm along a scaled version of the gradient. In lieu of (4) iterates are given by

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

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

$$H_k d_k = -g_k, \tag{7}$$

where $H_k = H(\lambda_k) = \nabla^2 q(\lambda_k)$ is the Hessian of the dual function at the current iterate and $g_k = g_k(\lambda_k)$ is, we recall, the corresponding gradient.

To obtain an expression for the dual Hessian, consider given dual λ_k and primal $x_k = x(\lambda_k)$ variables, and consider the second order approximation of the primal objective centered at the current primal iterates x_k,

$$\hat{f}(y) = f(x_k) + \nabla f(x_k)'(y - x_k) + \frac{1}{2}(y - x_k)'\nabla^2 f(x_k)(y - x_k) \tag{8}$$

The primal optimization problem in (1) is now replaced by the maximization of the approximating function $-\hat{f}(y)$ in (8) subject to the constraint $Ay = b$. This approximated problem is a quadratic program whose dual is the (also quadratic) program with properly defined vector p and scalar r,

$$\min_{\lambda \in \mathbb{R}^n} g(\lambda_k) = \min_{\lambda \in \mathbb{R}^n} \frac{1}{2}\lambda_k' A \nabla^2 f(x_k)^{-1} A' \lambda_k + p'\lambda_k + r. \tag{9}$$

The vector p and the constant r can be expressed in closed form as functions of $\nabla f(x_k)$ and $\nabla^2 f(x_k)$, but they are irrelevant for the discussion here. The important consequence of (9) is that the dual Hessian is given by

$$H_k = Q = A\left(-\nabla^2 f(x_k)^{-1}\right) A'. \tag{10}$$

From the definition of $f(x)$ in (1) it follows that the primal Hessian -$\nabla^2 f(x_k)$ is a diagonal matrix, which is negative definite by strict convexity of $f(x)$. Therefore, its inverse exists and can be computed locally. Further observe that the dual Hessian, being the product of the incidence matrix times a positive definite diagonal matrix times the incidence matrix transpose, is a weighted version of the network graph's Laplacian. As a particular consequence it follows that $\mathbf{1}$ is an eigenvector of H_k associated with eigenvalue 0 and that H_k is invertible on the the subspace $\mathbf{1}^\perp$. Since the gradient g_k lies in $\mathbf{1}^\perp$ we can find the Newton step as $d_k = -H_k^\dagger g_k$. However, computation of the pseudoinverse $H_k^\dagger$ requires global information. We are therefore interested in approximations of the Newton direction requiring local information only.

III. Approximate Newton's Method

To define an approximate Newton direction, i.e., one for which (7) is approximately true, we will consider a finite number of terms of a suitable Taylor's expansion representation of the Newton direction. At iteration k, split the Hessian into diagonal elements D_k and off diagonal elements B_k and write $H_k = D_k - B_k$. Further rewrite the Hessian as $H_k = D_k^{-\frac{1}{2}}\left(I - D_k^{-\frac{1}{2}} B_k D_k^{-\frac{1}{2}}\right) D_k^{-\frac{1}{2}}$, which implies that the Hessian pseudo-inverse is given by $H_k^{-\dagger} = D_k^{-\frac{1}{2}}\left(I - D_k^{-\frac{1}{2}} B_k D_k^{-\frac{1}{2}}\right)^{-\dagger} D_k^{-\frac{1}{2}}$. Notice now that 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 eigenvectors of X associated with eigenvalue 1. Since g_k is orthogonal to $\mathbf{1}$, it follows that

$$d_k = -H_k^\dagger g_k = -\sum_{i=0}^{\infty} D_k^{-\frac{1}{2}}\left(D_k^{-\frac{1}{2}} B_k D_k^{-\frac{1}{2}}\right)^i D_k^{-\frac{1}{2}} g_k.$$

Since the Newton direction is now represented as an infinite sum, we can define a family of approximations characterized by truncations of this sum,

$$d_k^{(N)} = -\sum_{i=0}^{N} D_k^{-\frac{1}{2}}\left(D_k^{-\frac{1}{2}} B_k D_k^{-\frac{1}{2}}\right)^i D_k^{-\frac{1}{2}} g_k := -\bar{H}_k^{(N)} g_k, \tag{11}$$

where we have defined the approximate Hessian pseudo inverse $\bar{H}_k^{(N)} := \sum_{i=0}^{N} D_k^{-\frac{1}{2}}\left(D_k^{-\frac{1}{2}} B_k D_k^{-\frac{1}{2}}\right)^i D_k^{-\frac{1}{2}}$. The approximate Newton algorithm is obtained by replacing the Newton step d_k in (6) by its approximations $d_k^{(N)} = -\bar{H}_k^{(N)} g_k$. The resultant algorithm is characterized by the iteration

$$\lambda_{k+1} = \lambda_k - \alpha_k \bar{H}_k^{(N)} g_k. \tag{12}$$

While not obvious, the choice of N in (11) dictates how much information node i needs from the network in order to compute the ith element of the approximate Newton direction $d_k^{(N)}$ – recall that node i is associated with dual variable λ_k^i.

For the zeroth order approximation $d_k^{(0)}$ only the first term of the sum in (11) 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 [1].

The first order approximation $d_k^{(1)}$ uses the first two terms of the sum in (11) 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, and as a consequence the sparsity pattern of $D_k^{-1} B_k D_k^{-1}$, is that of the graph Laplacian, which means that $[D_k^{-1} B_k D_k^{-1}]_{ij} \neq 0$ if and only if i and j correspond to an edge in the graph, i.e, $(i, j) \in E$. 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 is easy to realize 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 (11) 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 a trade off between communication cost and precision of the Newton direction. We analyze convergence properties of these methods in the coming sections.

A. Convergence

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 (12) 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 (12) eventually approach a neighborhood of the optimal λ^*. This claim is stated in the next proposition[1].

Proposition 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 (12) with $\bar{H}_k^{(N)}$ as in (11). Assume $\alpha_k = \alpha$ for all k and that the network graph is not bipartite. Then, for all sufficiently small α,*

$$\lim_{k\to\infty} \lambda_k = \lambda^* \tag{13}$$

By continuity of (3), convergence of the dual variable to an error neighborhood implies convergence of the primal variables to an error neighborhood. Requiring the graph to *not* be bipartite is a technical condition to avoid instabilities created by Laplacian eigenvalues at -1. The restriction is not significant in practice.

IV. Convergence Rate

The basic guarantee in Proposition 1 is not stronger than convergence results for regular gradient descent. Our goal is to show that the approximate Newton method in (12) exhibits quadratic convergence in a sense similar to centralized (exact) Newton algorithms. Specifically, we will show that selecting N large enough, it is possible to find a neighborhood of λ^* such that if the iteration is started within that neighborhood iterates converge quadratically.

Before introducing this result let us define the Newton approximation error ϵ_k as

$$\epsilon_k = H_k d_k^{(N)} + g_k. \tag{14}$$

We further introduce the following standard assumptions to bound the rate of change in the Hessian of the dual function.

Assumption 1. *The Hessian $H(\lambda)$ of the dual function $q(\lambda)$ satisfies the following conditions*

(Lipschitz dual Hessian) There exists some constant $L > 0$ such that $\|H(\lambda) - H(\bar{\lambda})\| \leq L\|\lambda - \bar{\lambda}\|\ \forall \lambda, \bar{\lambda} \in \mathbb{R}^n$.

(Strictly convex dual function) There exists some constant $M > 0$ such that $\|H(\lambda)^{-1}\| \leq M \qquad \forall \lambda \in \mathbb{R}^n$.

As is usual in second order optimization methods we use the gradient norm $\|g_k\| = \|g(\lambda_k)\|$ to measure the progress of the algorithm. The aforementioned quadratic convergence result establishes that for any graph we can always select N large enough so that if an iterate λ_k is sufficiently close to the optimal λ^*, the gradient norm $\|g_{k+m}\|$ of subsequent iterates λ_{k+m} decays like 2^{2^m}. This is formally stated in the following.proposition

[1]Proofs are omitted for brevity and can be found in [20].

Proposition 2. *Consider ADD-N algorithms characterized by the iteration (12) with $\bar{H}_k^{(N)}$ as defined in (11). Let Assumption 1 hold and further assume that the step size is $\alpha_{k+m} = 1$ for all $k \geq m$. Let ϵ be a uniform bound in the norm of the Newton approximation error ϵ_k in (14) so that $\|\epsilon_k\| \leq \epsilon$ for all k. Define the constant*

$$B = \epsilon + M^2 L \epsilon^2. \tag{15}$$

Further assume that at time k it holds $\|g_k\| \leq 1/(2M^2L)$ and that N is chosen large enough to ensure that for some $\delta \in (0, 1/2)$, $B + M^2LB^2 \leq \delta/(4M^2L)$. Then, for all $m \geq 1$,

$$\|g_{k+m}\| \leq \frac{1}{2^{2^m} M^2 L} + B + \frac{\delta}{M^2 L} \frac{(2^{2^m-1} - 1)}{2^{2^m}}. \tag{16}$$

In particular, as $m \to \infty$ it holds

$$\limsup_{m\to\infty} \|g_{k+m}\| \leq B + \frac{\delta}{2M^2L}. \tag{17}$$

Proposition 2 has the same structure of local convergence results for Newton's method [5, Section 9.5]. In particular, quadratic convergence follows from the term $1/\left(2^{2^m}\right)$ in (16). The remaining terms in (16) are small constants that account for the error in the approximation of the Newton step.

Notice that Proposition 2 assumes that at some point in the algorithm's progression, $\|g_k\| \leq 1/(2M^2L)$. Quadratic convergence is only guaranteed for subsequent iterates λ_{k+m}. This is not a drawback of ADD, but a characteristic of all second order descent algorithms. To ensure that some iterate λ_k does come close to λ^* so that $\|g_k\| \leq 1/(2M^2L)$ we use a distributed adaptation of backtracking line search (Section IV-A).

The proof of Proposition 2 relies on two lemmas which we present for reference. The first result concerns the bound ϵ which was required to hold uniformly for all iteration indexes k. While it is clear that increasing N reduces $\|\epsilon_k\|$, it is not immediate that a uniform bound should exist. The fact that a uniform bound does exists is claimed in the following lemma.

Lemma 1. *Given an arbitrary $\epsilon > 0$, there exists an N such that the Newton approximation errors ϵ_k as defined in (14) have uniformly bounded norms $\|\epsilon_k\| \leq \epsilon$ for all iteration indexes k.*

The proof is omitted for brevity but the key step uses [11] to uniformly bound the spectral gap away from one as follows

$$\rho\left(B_k D_k^{-1}\right) \leq 1 - \frac{1}{n\Delta(G)(\text{diam}(G)+1)b_{\max}} \tag{18}$$

where $\Delta(G)$ is the maximum degree of any node in G, $\text{diam}(G)$ is the diameter of G and $b_{\max}$ is an upper bound on dual off diagonal elements of the dual hessian: $[H_k]_{ij} \leq b_{\max}\ \forall i \neq j$.

Another preliminary result necessary for the proof of Proposition 2 is an iterative relationship between the gradient norm $\|g_{k+1}\|$ at iteration $k+1$ and the norm $\|g_k\|$ at iteration k. This relationship follows from a multi-dimensional extension of the descent lemma (see [4]) as we explain next.

Lemma 2. *Let Assumption 1 hold. Let $\{\lambda_k\}$ be a sequence generated by the method (6). For any stepsize rule α_k, we have $\|g_{k+1}\| \leq (1-\alpha_k)\|g_k\| + M^2 L \alpha_k^2 \|g_k\|^2 + \alpha_k \|\epsilon_k\| + M^2 L \alpha_k^2 \|\epsilon_k\|^2$.*

The proof of Proposition 2 follows from recursive application of the result in Lemma 2.

A. Distributed backtracking line search

Proposition 2 establishes *local* quadratic convergence for properly selected members of the ADD family. To guarantee *global* convergence we modify ADD to use time varying step sizes α_k selected through distributed backtracking line search [5, Algorithm 9.2]. Line search implementation requires computation of the gradient norm $\|g_k\| = \sum_{i=1}^n g_k^{i\,2}$. This can be easily achieved using distributed consensus algorithms, e.g., [7]. However, since these consensus algorithms are iterative in nature, an approximate norm η_k is computed in lieu of $\|g_k\|$. We assume that approximate gradient norms are computed with an error not exceeding a given constant $\gamma/2 \geq 0$,

$$\Big|\eta_k - \|g_k\|\Big| \leq \gamma/2, \tag{19}$$

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

$$n_{k+1} \leq (1 - \sigma\beta^m)\eta_k + B + \gamma. \tag{20}$$

The expression in (20) coincides with the regular (centralized) backtracking line search except for the use of the approximate norm η_k instead of the actual norm $\|g_k\|$ and the (small) additive constants B and γ respectively defined in (15) and (19).

While we introduce line search to ensure *global* convergence we start by noting that stepsizes selected according to the rule in (20) do not affect *local* convergence. As we state next, this is because if $\|g_k\| \leq 1/(2M^2L)$ as required in Proposition 2 the rule in (20) selects stepsizes $\alpha_k = 1$.

Proposition 3. *If at iteration k of ADD-N the gradient norm satisfies $\|g_k\| \leq 1/(2M^2L)$, the inexact backtracking stepsize rule in (19) selects $\alpha_k = 1$.*

As per Proposition 3, if ADD-N with backtracking line search is initialized at a point at which $\|g_0\| \leq 1/(2M^2L)$, stepsizes $\alpha_k = 1$ are used. Therefore, Proposition 2 holds and convergence to the optimum λ^* is quadratic, which in practice implies convergence in a few steps. Otherwise, selecting step sizes α_k satisfying (20) ensures a strict decrease in the norm of the residual function as claimed by the proposition below.

Proposition 4. *Consider ADD-N algorithms characterized by the iteration (12) with $\bar{H}_k^{(N)}$ as defined in (11). Let Assumption 1 hold and stepsizes α_k being selected according to the inexact backtracking rule in (20). Further assume that $\|g_k\| > 1/2M^2L$ and that N is chosen large enough to ensure that the constants B and γ in (15) and (19) satisfy*

$$B + 2\gamma \leq \frac{\beta}{16M^2L}, \tag{21}$$

where β is the backtracking rule constant and M and L are defined in Assumption 1. Then, the gradient norm at iteration $k+1$ decreases by at least $\beta/(16M^2L)$,

$$\|g_{k+1}\| \leq \|g_k\| - \frac{\beta}{16M^2L}. \tag{22}$$

Proposition 4 shows that if ADD-N is initialized at a point with gradient norm $\|g_0\| > 1/2M^2L$ we obtain a decrease in the norm of the gradient of at least $\beta/16M^2L$. This holds true for all iterations as long as $\|g_k\| > 1/2M^2L$. This establishes that we need at most $16\|g_0\|M^2L/\beta$ iterations until we obtain $\|g_k\| \leq 1/2M^2L$. At this point the quadratic convergence result in Proposition 2 comes in effect and ADD-N converges in a few extra steps.

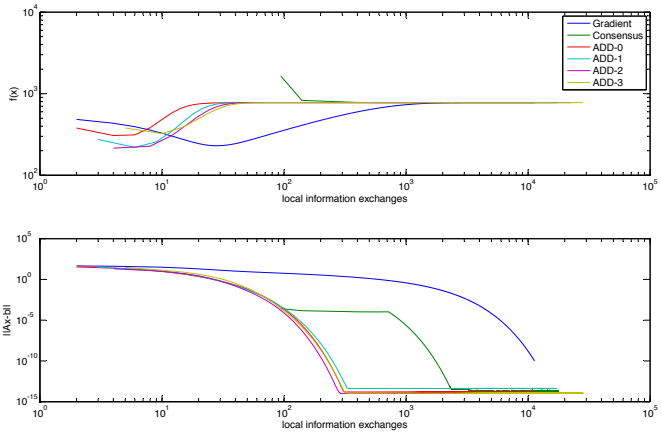

Fig. 1. Primal objective (top), $f(x_k)$ and primal feasibility (bottom), $\|Ax_k - b\|$ with respect to number of local information exchanges for a sample network optimization problem with 25 nodes and 75 edges. ADD converges an order of magnitude faster than consensus-based Newton and two orders of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

V. Numerical results

Numerical experiments are undertaken to study ADD's performance with respect to the choice of the number of approximating terms N. These experiments show that $N = 1$ or $N = 2$ work best in practice. ADD is also compared to dual gradient descent [14] and the consensus-based Newton method in [8].

Figure 1 shows convergence metrics for a randomly generated network. Edges in the network are selected uniformly at random. The flow vector b is chosen to place sources and sinks a full diam($\mathcal{G}$) away from each other. All figures show results for ADD-0 through ADD-3, gradient descent, and consensus-based Newton. In Fig. 1, objective value $f[x(\lambda_k)]$ and constraint violation $\|Ax(\lambda_k) - b\|$ are shown as functions of the number of local node to node communications which have occurred. Observe that in terms of this metric all versions of ADD are about an order of magnitude faster than consensus-based Newton and two orders of magnitude faster than gradient descent.

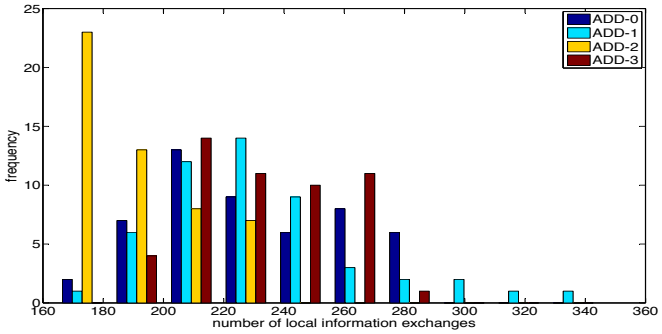

Fig. 2. Histogram of the number of local communications required to reach $\|g(\lambda_k)\| \leq 10^{-10}$ for ADD-N with respect to parameter N, for 50 trials of the network optimization problem on random graphs with 25 nodes and 75 edges. ADD-2 is shown to be the best on average by about 10% indicating that with respect to communication cost, larger N is not necessarily better.

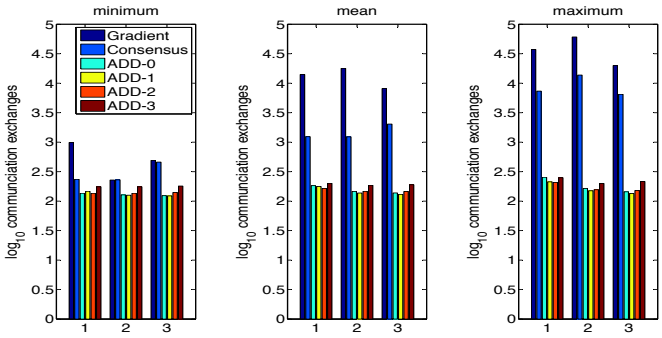

Fig. 3. Min (left), mean (center) and max (right) number of local communications required to reach $\|g(\lambda_k)\| \leq 10^{-10}$ for gradient descent, consensus-based Newton and ADD, computed for 35 trials each on random graphs with 25 nodes and 75 edges(1), 50 nodes and 350 edges(2), and 100 nodes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

An important conclusion drawn from our simulations 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. The parameter N 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 examine this phenomenon in Fig. 2. These experiments are on random graphs with 25 nodes and 75 edges chosen uniformly at random. The flow vector b places a source $\text{diam}(\mathcal{G})$ away from a sink. We consider an algorithm to have converged when its residual $\|g_k\| \leq 10^{-10}$.

The behavior of ADD is also explored for graphs of varying size and degree in Fig. 3. 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 neither consensus-based Newton nor gradient descent. Further note that ADD's communication cost increases only slightly with network size.

VI. 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. It is always possible to find members of this family for which convergence to optimal operating points is quadratic.

Simulations demonstrated that $N = 1$ and $N = 2$, respectively denoted as ADD-1 and ADD-2 perform best in practice. ADD-1 corresponds to Newton step approximations using information from neighboring nodes only, while ADD-2 requires information from nodes two hops away. 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.

Possible extensions include applications to network utility maximization [19], general wireless communication problems [17], and stochastic settings [16].

References

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

[2] Bertsekas and Gafni, *Projected newton methods and optimization of multi-commodity flow*, 1983, pp. 1090–1096.

[3] D.P. Bertsekas, *Nonlinear programming*, Athena Scientific, Cambridge, Massachusetts, 1999.

[4] D.P. Bertsekas, A. Nedić, and A.E. Ozdaglar, *Convex analysis and optimization*, Athena Scientific, Cambridge, Massachusetts, 2003.

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

[6] 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** (2007), no. 1, 255–312.

[7] A. Jadbabaie, J. Lin, and S. Morse, *Coordination of groups of mobile autonomous agents using nearest neighbor rules*, IEEE Transactions on Automatic Control **48** (2003), no. 6, 988–1001.

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

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

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

[11] H. J. Landau and A. M. Odlyzko, *Bounds for eigenvalues of certain stochastic matrices*, Linear Algebra and its Applications **38** (1981), 5–15.

[12] S. Low and D.E. Lapsley, *Optimization flow control, I: Basic algorithm and convergence*, IEEE/ACM Transactions on Networking **7** (1999), no. 6, 861–874.

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

[14] ———, *Distributed subradient methods for multi-agent optimization*, IEEE Transactions on Automatic Control, forthcoming (2008).

[15] ———, *Subgradient methods in network resource allocation: Rate analysis*, Proc. of CISS, 2008.

[16] A. Ribeiro, *Ergodic stochastic optimization algorithms for wireless communication and networking*, IEEE Transactions on Signal Processing (2009).

[17] A. Ribeiro and G. B. Giannakis, *Separation theorems of wireless networking*, IEEE Transactions on Information Theory (2007).

[18] R. Srikant, *Mathematics of Internet congestion control*, Birkhauser, 2004.

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

[20] M. Zargham, A. Ribeiro, A. Ozdaglar, and A. Jadbabaie, *Accelerated dual descent for network optimization*, arXiv.org (2011).