52nd IEEE Conference on Decision and Control
December 10-13, 2013. Florence, Italy

Accelerated Dual Descent for Constrained Convex Network Flow Optimization

Michael Zargham[†], Alejandro Ribeiro[†], Ali Jadbabaie[†]

***Abstract*—We present a fast distributed solution to the capacity constrained convex network flow optimization problem. Our solution is based on a distributed approximation of Newtons method called Accelerated Dual Descent (ADD). Our algorithm uses a parameterized approximate inverse Hessian, which is computed using matrix splitting techniques and a Neumann series truncated after N terms. The algorithm is called ADD-N because each update requires information from N-hop neighbors in the network. The parameter N characterizes an explicit trade off between information dependence and convergence rate. Numerical experiments show that even for N=1 and N=2, ADD-N converges orders of magnitude faster than subgradient descent.**

I. Introduction

The goal of this paper is to develop a fast distributed solution to the capacity constrained convex network flow optimization problem. Solutions to minimum cost network flow problems have long been used in operations research and transportation networks [1], [2]. The seminal results of Rockafellar in [3] tie the network flow problem to min cut and other combinatorial problems such as shortest path, [4][Chapter 1]. Network flow problems also arise in routing problems for communication networks, [5] where there is a need for distributed solutions. For example, solving a network flow problem is key subproblem in the wireless routing and resource allocation problem in [6].

Dual subgradient descent is a distributed algorithm used to solve the convex network flow optimization. Analysis of subgradient methods for convex optimization problems can be found in [7] and [8] with the latter taking into account uncertainty in the network structure. However, practical applicability of subgradient type algorithms is limited by exceedingly slow convergence rates, [9] and [10]. An alternative distributed algorithm based on the Gauss Seidel method is present in [11]. Like gradient descent, Gauss Seidel is a first order method. Faster methods such as dual Newton's method, require global information and the existence of a dual Hessian. The Accelerated Dual Descent (ADD) method is a distributed algorithm requiring information from an N-hop neighborhood in order to update the dual variables, [12], [13].

The ADD method is based on Newtons Method and therefore requires the existence of the dual Hessian. We show that capacity constraints result in a nonsmooth dual function. Therefore, there are points where dual Hessian does not exist and Newton type methods such as ADD cannot be directly applied. In this work we prove the existence of a generalized dual Hessian. We compute the generalized dual Hessian via local information and use it to implement the ADD-N algorithm. Using ADD-N with the generalized Hessian is proven to result in a descent direction guaranteeing convergence of the algorithm.

We begin the paper in Section II with the formal definition of the network flow optimization problem considered here. 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. Using the capacity constraints to restrict to positive valued flows recovers the standard implementation of the network flow optimization problem in [4], [14]. Each edge of the network is associated with a convex function that determines the cost of transmitting a unit of flow across it. The objective is to find the flows that minimize the sum of link costs by solving a convex optimization problem with linear constraints.

In Section II-A, we show that the dual problem is nonsmooth and we prove the existence of a generalized dual Hessian consistent with [15]. In Section III, we construct the ADD-N update using a Neumann expansion for the Hessian inverse, [16][Section 5.8] truncated after N terms. Since our inverse Hessian approximation is an N degree matrix polynomial our descent direction can be computed using N-hop neighbor information. In Section III-A, we demonstrate how the ADD-N direction can be implemented using N one hop exchanges per dual iteration. In Section IV, the ADD algorithm is shown to exhibit faster convergence rate relative to subgradient descent in numerical experiments.

II. Constrained 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 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

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 Ribero and Ali Jadbabaie are with the Department of Electrical and Systems Engineering, University of Pennsylvania.

matrix defined as

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

We define the reward as the negative of a sum of convex cost functions $\phi_e(x^e)$ denoting the cost of x^e units of flow traversing edge e. We assume that the cost functions ϕ_e are strictly convex, surjective and twice continuously differentiable on the interval $c_l^e \le x^e \le c_u^e$ where $c_l^e \le 0$ and $c_u^e \ge 0$ are the capacity constraints on edge e. The direction of flow can be forced to match the direction of the edge by choosing $c_l^e = 0$. The network flow optimization problem is then defined as

$$\begin{array}{ll} \text{min} & f(x) = \sum_{e=1}^{E} \phi_e(x^e) \\ \text{subject to:} & Ax = b \\ & c_l \le x \le c_u. \end{array} \tag{1}$$

Keeping the capacity constraints and dualizing the equality constraint, we have the partial dual

$$\begin{aligned} q(\lambda) &= \min_x \sum_{e=1}^{E} \phi_e(x^e) + \lambda'(Ax - b) \\ &\quad \text{subject to: } c_l \le x \le c_u. \end{aligned} \tag{2}$$

We separate this equation into an edge dimensional sum of optimization problems

$$\begin{aligned} q(\lambda) &= -\lambda' b + \textstyle\sum_{e=1}^{E} \min_x \phi_e(x^e) + x^e(\Delta\lambda_e) \\ &\qquad\qquad \text{subject to: } c_l^e \le x^e \le c_u^e \end{aligned} \tag{3}$$

where $\Delta\lambda_e = \lambda_j - \lambda_i$ where $e = (i, j)$. Each one of these optimization problems is a constrained convex optimization local to one edge allowing us to write the primal optimizers explicitly

$$x^e(\Delta\lambda_e) = \left[\frac{d}{dx^e}\phi^e\right]^{-1}(\Delta\lambda_e)\Bigg|_{c_l^e}^{c_u^e}. \tag{4}$$

We are therefore interested in solving

$$\lambda^* = \arg\max_{\lambda} q(\lambda) \tag{5}$$

and computing the primal variables $x^* = x(\lambda^*)$ which are guaranteed to be optimal by strong duality via Slater's condition, [10]. Considering (2), we can compute the gradient $g(\lambda) = \nabla q(\lambda)$ in terms of the primal optimizers,

$$g(\lambda) = Ax(\lambda) - b \tag{6}$$

where $x(\lambda)$ denotes the vector of $x^e(\Delta\lambda_e)$. The projection in (4) causes (6) to become a non-smooth function. This work focuses on addressing this non-smoothness.

A. Non-smooth Newton Method

Consider an iteration index k, an arbitrary initial vector λ_0 and define iterates λ_k generated by the following recursion

$$\lambda_{k+1} = \lambda_k + \alpha_k d_k \qquad \text{for all } k \ge 0, \tag{7}$$

where d_k is an ascent direction satisfying $g_k' d_k > 0$, $g_k = g(\lambda_k)$ and α_k is a given stepsize sequence. The Newton method is obtained by making d_k in (7) the Newton step, $g_k = -H_k d_k$. The Newton step is explicitly given by $d_k = -H_k^{\dagger} g_k$ for any k where the dual Hessian H_k exists[1].

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,

$$\begin{aligned} \hat{f}(y) &= f(x_k) + \nabla f(x_k)'(y - x_k) \\ &\quad + \frac{1}{2}(y - x_k)'\nabla^2 f(x_k)(y - x_k). \end{aligned} \tag{8}$$

The primal optimization problem in (1) is now replaced by the minimization of the approximating function $\hat{f}(y)$ in (8) subject to the constraints $Ay = b$ and $c_l \le y \le c_u$. This approximated problem is a quadratic program whose dual is a piece-wise quadratic function

$$\begin{aligned} \hat{q}(\lambda_k) &= -\frac{1}{2}\lambda' A \left[\nabla^2 f(x_k)^{-1} A'\lambda_k\right]_{c_l}^{c_u} + p'\lambda_k + r \\ &- \frac{1}{2}\left[\lambda_k' A \nabla^2 f(x_k)^{-1}\right]_{c_l}^{c_u} \nabla^2 f(x_k) \left[\nabla^2 f(x_k)^{-1} A'\lambda_k\right]_{c_l}^{c_u} \end{aligned} \tag{9}$$

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. When $\hat{f}(x_k)$ is centered at $x_k = x(\lambda_k)$, we have $\hat{f}(x(\lambda_k)) = f(x(\lambda_k))$ and it follows that $\hat{q}(\lambda_k) = q(\lambda_k)$ for any λ_k where the implicit primal optimizers $x_k = x(\lambda_k)$ are used. The important consequence of (9) is that we can compute the dual Hessian. When the $c_l \le f(x_k)^{-1}A'\lambda_k \le c_u$, the projection is inactive and we recover the Hessian

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

discussed in [13]. We define the set of unsaturated edges

$$\mathcal{U}_k = \{e \in E | c_l^e < x^e(\lambda_k) < c_u^e\}. \tag{11}$$

The dual Hessian $\nabla^2 q(\lambda_k)$ is dependent on which edges e are unsaturated, i.e. $e \in \mathcal{U}_k$. When edge $e = (i, j)$ saturates, we have that the ith element of the gradient $g_i(\lambda_k)$ is unaffected by changes in λ_k^j so $\partial g(\lambda_k)/\partial\lambda_k^j = 0$. Thus the Hessian elements $[H_k]_{i,j} = [H_k]_{j,i}$ are zero. We define the off diagonal elements of the generalized Hessian $H_k = \nabla^2 q(\lambda_k)$, according to

$$\frac{\partial^2 q(\lambda_k)}{\partial\lambda_k^j \partial\lambda_k^i} = \begin{cases} -[A\nabla^2 f(x_k)^{-1} A']_{ij} & \text{if } (i,j) \in \mathcal{U}_k \\ 0 & \text{if } (i,j) \notin \mathcal{U}_k \end{cases} \tag{12}$$

[1] $H_k^{\dagger}$ denotes the Moore-Penrose pseudoinverse of the dual function's Hessian $H_k = H(\lambda_k)$ and $g_k \in \mathbf{1}^{\perp}$ for all k.

and the diagonal elements are the sums of the off diagonals

$$[H_k]_{ii} = -\sum_{j:(i,j)\in\mathcal{U}_k} [A\nabla^2 f(x_k)^{-1}A']_{ij}. \qquad (13)$$

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 positive definite by strict convexity of $f(x)$. Therefore, its inverse exists and can be computed locally. Further observe that $L_k = A\nabla^2 f(x_k)^{-1}A'$ is a weighted version of the network graph's Laplacian. One consequence of its Laplacian form is that $H_k = -L_k$ is negative semi-definite. Another is that $\mathbf{1}$ is an eigenvector of H_k associated with eigenvalue 0 and that H_k is invertible on the subspace $\mathbf{1}^\perp$.

B. Existence of the Generalized Hessian

We still need to understand how the projection in (4) and resulting non-smoothness in (6) impacts the existence of this dual Hessian.

Theorem 1. *Properties of the Dual Gradient:*

1) *The function $g : \mathbb{R}^n \to \mathbb{R}^n$ is Lipschitz Continuous.*

$$||g(\lambda) - g(\bar{\lambda})|| \le L||\lambda - \bar{\lambda}|| \qquad \forall \lambda, \bar{\lambda} \in \mathbb{R}^n \qquad (14)$$

2) *The function $g : \mathbb{R}^n \to \mathbb{R}^n$ is Semi-smooth:*

$$\lim_{V\in\partial g(\lambda+th), t\to 0} \{Vh\} \text{ Exists} \forall h \in \mathbb{R}^n \qquad (15)$$

where $\partial g(\lambda)$ is the generalized Hessian defined

$$\partial g(\lambda) = co\left\{ \lim_{\bar{\lambda}\to\lambda, \bar{\lambda}\in D_g} \nabla g(\bar{\lambda}) \right\} \qquad (16)$$

where $co\{\cdot\}$ denotes convex hull and D_g is the set of points on which $g(\lambda)$ is differentiable.

Proof: From our assumptions on $f(x)$ in Section II we know that $\phi_e(x^e)$ is strictly convex, surjective and twice continuously differentiable, therefore the function $\left[\frac{d}{dx^e}\phi^e\right]^{-1}(\cdot)$ is continuous. We denote the vector of these functions $\nabla f^{-1}(\cdot)$.The primal optimal variables are computed according to (4), the saturation operator $|_{c_l^e}^{c_u^e}$ preserves continuity but the function is no longer differentiable. Applying (6), we observe from

$$g(\lambda - \bar{\lambda}) = A\big(x(\lambda) - x(\bar{\lambda})\big) \qquad (17)$$

that continuity of $x(\lambda)$ is sufficient for continuity of $g(\lambda)$. In order to show $g(\lambda)$ is Lipschitz we substitute the definition of $x(\lambda)$ from (4).

$$\big\|g(\lambda) - g(\bar{\lambda})\big\| = \big\|A(\nabla f^{-1}(\Delta\lambda)|_{c_l}^{c_u} - \nabla f^{-1}(\Delta\bar{\lambda})|_{c_l}^{c_u})\big\| \qquad (18)$$

Since $f(x)$ is twice continuously differentiable and surjective, it follows that ∇f^{-1} is continuously differentiable. Let $y(x) = \nabla f^{-1}(x)$ which means $\nabla f(y(x)) = x$ by the definition of the function inverse. We take the gradient of this expression with respect to x yielding $\nabla^2 f(y(x))\nabla y(x) = \mathbf{1}$ by the chain rule. Since $f(x)$ is strictly convex we have an expression for the gradient of ∇f^{-1}

$$\nabla y(x) = \nabla[\nabla f^{-1}](x) = [\nabla f(y(x))]^{-1}\mathbf{1}. \qquad (19)$$

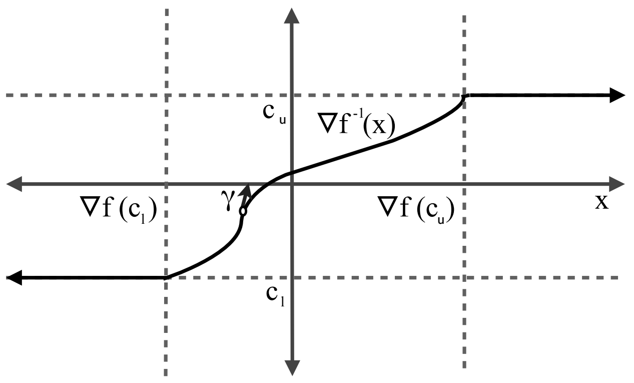

Fig. 1. The constant γ is the steepest gradient point of the function $\nabla f^{-1}(x)$ on its active domain, thus making it a property of the primal objective $f(x)$ and the capacity constraints. This image simplifies the problem to 1 dimension to make it visualizable.

Now we will find the steepest possible slope of ∇f^{-1}

$$\begin{aligned} \gamma \;&=\; \max\ ||\nabla y(x)|| \\ &\quad \text{s.t. } \nabla f(c_l) \le x \le \nabla f(c_u). \end{aligned} \qquad (20)$$

We need not consider any x outside of the set $\mathcal{J} = \{\nabla f(c_l) \le x \le \nabla f(c_u)\}$ because using that fact that ∇f^{-1} is monotonic increasing we know that any such x yields $\nabla f^{-1}(x) \le c_l$ or $\nabla f^{-1}(x) \ge c_u$. We can guarantee the existence of γ as defined in (20) by restricting the domain of ∇y to the set $\mathcal{J}$ because any continuous function from a compact space into a metric space is bounded. Using (20) we can conclude that

$$||(\nabla f^{-1}(\Delta\lambda)|_{c_l}^{c_u} - \nabla f^{-1}(\Delta\bar{\lambda})|_{c_l}^{c_u})|| \le \gamma||\Delta\lambda - \Delta\bar{\lambda}|| \quad (21)$$

Recalling that $\Delta\lambda = A\lambda$ for any λ and subsituting (21) back into (18) we have

$$\big\|g(\lambda) - g(\bar{\lambda})\big\| \le \gamma||A||^2||\lambda - \bar{\lambda}|| \qquad (22)$$

which completes the proof of part 1) with $L = \gamma||A||^2$.

To continue with part 2), we consider the points for which $g(\lambda)$ is not differentiable

$$\mathcal{S} = \left\{ \lambda : \begin{array}{ll} \left[\frac{d}{dx^e}\phi^e\right]^{-1}(\Delta\lambda_e) = c_l^e & \text{for any } e \text{ or} \\ \left[\frac{d}{dx^e}\phi^e\right]^{-1}(\Delta\lambda_e) = c_u^e, & \text{for any } e \end{array} \right\}. \quad (23)$$

For any $\lambda \notin \mathcal{S}$, we have $\partial g(\lambda) = \{H(\lambda)\}$ as defined in (12) and (13) because $g(\lambda)$ is differentiable at those points. The limit as we approach a point in $\mathcal{S}$ depends which edges are saturating and on whether the direction of the limit is from inside or outside of the saturation region. From within the saturation region, $\bar{\lambda} \in \{\lambda \in \mathbb{R}^n | (i,j) \in \mathcal{U}_k\}$

$$\left[\lim_{\lambda\to\bar{\lambda}, \lambda\in\mathbb{R}^n-\mathcal{S}} \nabla g(\bar{\lambda})\right]_{ij} = 0 \qquad (24)$$

but when outside the saturation region $\bar{\lambda} \in \{\lambda \in \mathbb{R}^n | (i,j) \in \mathcal{U}_k\}$

$$\left[\lim_{\lambda\to\bar{\lambda}, \lambda\in\mathbb{R}^n-\mathcal{S}} \nabla g(\bar{\lambda})\right]_{ij} = [H_k]_{ij} \qquad (25)$$

with H_k as defined in (10). The set of generalized Hessians $\partial g(\lambda)$ defined in (16) allows

$$[\nabla g(\lambda)]_{i,j} \in [[H_k]_{ij}, 0] \tag{26}$$

for any edge $e = (i, j)$ for which $x^e(\lambda) = c_l^e$ or $x^e(\lambda) = c_u^e$. By our construction of the set $\partial g(\lambda)$ in (24)-(26), for any matrix $V \in \partial g(\lambda)$ including $\lambda \in \mathcal{S}$, the expression Vh exists for all $h \in \mathbb{R}^n$. We conclude that (15) holds, completing the proof. ■

Theorem 1 characterizes the dual gradient $g(\lambda)$, providing a set of generalized Hessian matrices at the points where $g(\lambda)$ is not differentiable. In our proof, equation, (26) it is shown that our choice of $H(\lambda)$ defined in (12) and (13) is an extreme point in the set of generalized Hessians $\partial g(\lambda)$. While we have shown that the entire set of generalized Hessians exists our algorithm uses this extreme point for its direct relationship to the Hessian in the unconstrained (smooth) version of network flow optimization problem analyzed in [13]. Theorem 1 also specifies that the dual gradient $g(\lambda)$ is Lipshitz continuous which is a building block for our convergence analysis.

Theorem 2. *Suppose $g(\lambda)$ is semi-smooth at λ^* and all $V \in \partial g(\lambda)$ are non-singular and $g(\lambda)$ is Lipshitz continous, then the iteration*

$$\lambda_{k+1} = \lambda_k - V_k^{-1} g(\lambda_k) \tag{27}$$

is well defined and globally convergent to the unique solution λ^* of

$$g(\lambda) = 0 \tag{28}$$

at a q-superlinear rate.

Theorem 2 is *Newton's Method for piecewise quadratic functions* taken from [15][Theorem 1.1]. It states that we can solve a peicewise quadratic optimization problem such as the one we have in (9) using Newton's method as long as the generalized Hessian exists and is full rank. In our case, it converges as long as the inverse of the primal Hessian $\nabla^2 f(x_k)^{-1}$ is full rank and the reduced graph $\hat{\mathcal{G}}_k = \{\mathcal{V}, \mathcal{U}_k\}$ remains connected. Unfortunately, we cannot guarantee $\hat{\mathcal{G}}_k$ will remain connected. However, the approximate Newton method, called ADD developed in [13] when applied with an enhance splitting technique, allows us to sidestep this concern. Furthermore, unlike the true Newton method, can be computed using only local information.

III. Non-smooth Accelerated Dual Descent

The ADD-N algorithm uses matrix splitting to generate an approximation of the Newton direction requiring information from no more than N hops away. We consider a finite number of terms of a suitable Taylor's expansion representation of the Newton direction. At iteration k, split the Hessian $H_k = B_k - D_k$, with diagonal elements D_k and off diagonal elements B_k where both D_k and B_k are nonnegative elementwise. To handle potential singularities we apply an enhanced splitting $H_k = \bar{B}_k - \bar{D}_k$ where

$$\bar{D}_k = 2D_k + I \qquad \text{and} \qquad \bar{B}_k = D_k + I + B_k \tag{29}$$

observing that $\bar{D}_k$ is a still a diagonal matrix and $\bar{B}_k$ retains the structure of a weighted adjacency. Using our spitting, we can define the approximate Hessian inverse as defined in [13],

$$\begin{aligned} \bar{H}_k^{(N)} &= -\sum_{r=0}^{N} \bar{D}_k^{-\frac{1}{2}} \left(\bar{D}_k^{-\frac{1}{2}} \bar{B}_k \bar{D}_k^{-\frac{1}{2}}\right)^r \bar{D}_k^{-\frac{1}{2}} \\ &= -\sum_{r=0}^{N} \left(\bar{D}_k^{-1} \bar{B}_k\right)^r \bar{D}_k^{-1} \end{aligned} \tag{30}$$

which exploits the Neumann series for the inverse of a matrix of the form $(I - X)$. Given the approximate Hessian inverse $\bar{H}_k^{(N)}$ our approximate Newton direction is given

$$d_k^{(N)} = \bar{H}_k^{(N)} g_k \tag{31}$$

where the accelerated dual descent iteration is

$$\lambda_{k+1} = \lambda_k + \alpha_k d_k^{(N)} \tag{32}$$

and α_k is the step size at iteration k.

The Nth order approximation $\bar{H}_k^{(N)}$ adds a term of the form $\left(\bar{D}_k^{-1} \bar{B}_k\right)^N \bar{D}_k^{-1}$ to the $N - 1$st order approximation. The sparsity pattern of this term is that of $\bar{B}_k^N$, which coincides with the N-hop neighborhood because it is the N^{th} power of a weighted adjacency matrix. Thus, computation of the local elements of the Newton step necessitates information from N hops away; see [13] for further details. We thus interpret (31) as a family of approximations indexed by N that yields Hessian approximations requiring information from N-hop neighbors in the network.

A. Algorithm Implementation

The direct implementation of the dual update described in (32) can be computationally cumbersome. In practice we implement ADD-N as an N steps of the consensus iteration

$$d_k^{(r+1)} = \bar{D}_k^{-1} \bar{B} d_k^{(r)} + \bar{D}_k^{-1} g_k \tag{33}$$

where the iteration is started with $d_k^{(0)} = \bar{D}_k^{-1} g_k$ which follows clearly from (30). Further explanation of the equivalence of ADD-N and N step consensus methods can be found in [13]. Our practical implementation of ADD-N is given in Algorithm 1. We assume that node i keeps track of the i^{th} element of node dimensional variables such as λ_k and the nonzero elements of the i^{th} row of matrices such as H_k and the splitting matrices. We also assume that the edge variables such as x^e are observable by both nodes i and j linked by edge $e = (i, j)$. With this framework, steps 3 and 10 are the only ones which require communication with neighbors– one hop neighbors only. Therefore, each full iteration $\lambda_{k+1} \leftarrow \lambda_k$ costs $N + 1$ local information exchanges.

B. Convergence

Theorem 3. *Algorithm 1 with fixed step size $\alpha_k = \alpha > 0$ small enough converges globally to the optimal dual variable λ^* for the dual problem in* (9) *and $x^* = x(\lambda^*)$ is the optimal solution to the original constrained network flow optimization problem,* (1).

Algorithm 1: Accelerated Dual Descent.

```
1  Initialize dual: $\lambda_0$
2  for $k = 0, 1, 2, \ldots$ do
3      Compute differentials: $\Delta\lambda_k = A\lambda_k$
4      Update Primals: $x_k = \nabla f^{-1}(\Delta\lambda_k)\big|_{c_l}^{c_u}$
5      Compute Constraint Violation: $g_k = Ax_k - b$
6      Compute the Generalized Hessian:
           $[H_k]_{ij} = \begin{cases} 1/\phi''_e(x_k^e) & \text{if } e = (i,j) \in \mathcal{U}_k \\ 0 & \text{else} \end{cases}$
           $[H_k]_{ii} = -\sum_j [H_k]_{ij}$
7      Compute Splitting: $\bar{D}_k = I - 2\ \mathrm{diag}(H_k)$ and
       $\bar{B}_k = \bar{D}_k + H_k$
8      Initialize direction: $d_k^{(0)} = \bar{D}_k^{-1} g_k$
9      for $r = 0, 1, \ldots, N-1$ do
10         Update direction: $d_k^{(r+1)} = \bar{D}_k^{-1}\bar{B}d_k^{(r)} + \bar{D}_k^{-1}g_k$
11     end
12     Update dual: $\lambda_{k+1} = \lambda_k + \alpha_k d_k^{(N)}$
13 end
```

Proof: As per the Descent Lemma [17][A.24], it is sufficient to show that the approximate Hessian inverse $\bar{H}_k^{(N)}$ is negative definite for all k to guarantee that the ADD-N iteration defined in (31) and (32) converges to the optimal dual variable λ^* for some fixed step size α. We consider the individual terms of $\bar{H}_k^{(N)} = \sum_{r=0}^{N} T_r$ as defined in (30). The matrix D_k is diagonal positive semi-definite, therefore $\bar{D}_k = 2D_k + I$ is diagonal and positive definite. Using the fact that $L_k = -H_k$ is a Laplacian from (12) and (13) and the basic splitting $L_k = D_k - B_k$, the matrix $\bar{B}_k = D_k + I + B_k$ is strictly diagonally dominant. Furthermore, by construction $\bar{B}_k$ is a nonnegative symmetric matrix. A symmetric, nonnegative, strictly diagonally dominant matrix is positive definite [16][Section 6.1]. We can conclude that each term

$$T_r = \bar{D}_k^{-1/2}(\bar{D}_k^{-1/2}\bar{B}_k\bar{D}_k^{-1/2})^r\bar{D}_k^{-1/2} \tag{34}$$

is positive definite because its symmetric and a product of positive definite matrices. Since each term T_r is positive definite the finite sum $\sum_{r=0}^{N} T_r$ is positive definite, so from (30) we get that $\bar{H}_k^{(N)}$ is negative definite and we conclude that the dual iteration converges to λ^*.

The primal problem (1) satisfies Slater's condition for strong duality [10][Section 5.2.3], therefore there is a zero duality gap, indicating that the primal optimal x^* is the argument of the minimization in (2) when $\lambda = \lambda^*$. Thus $x^* = x(\lambda^*)$ computed according to (4) is the optimal primal variable, completing the proof. ■

Theorem 3 guarantees that the ADD-N algorithm converges to the optimal primal and dual variables given an appropriately chosen fixed step size. A relatively loose bound on the allowable step size can be determined by considering the Descent Lemma [17][A.24]. In our case such a bound will be inversely proportional to the Lipschitz constant L defined in Theorem 1. In practice, the appropriate step size scale can be determined by tuning. We also make no theoretical claim about the convergence rate, which appears in practice to be consistent with the results in [13].

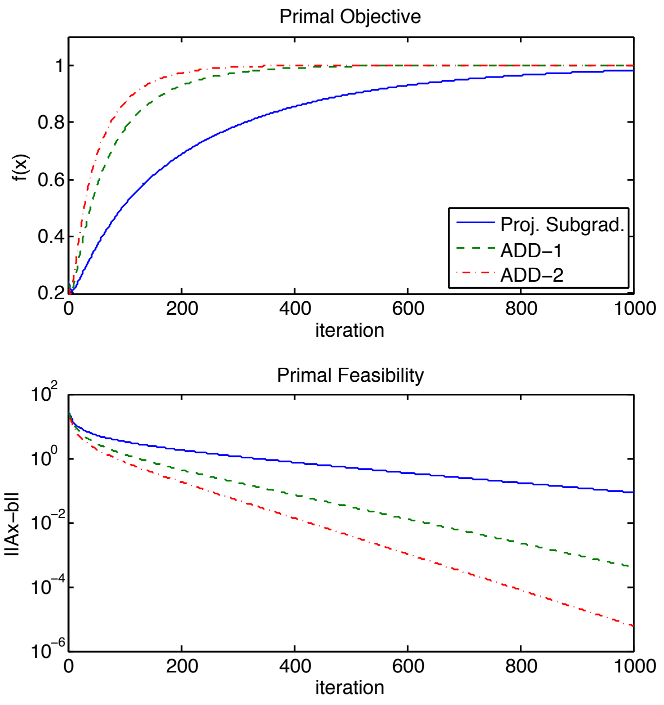

Fig. 2. ADD-1 and ADD-2 out perform projected subgradient by an order of magnitude on a sparse network with 20 nodes and 35 edges with capacity constraints

IV. Numerical Experiments

Numerical experiments were run on a variety of problems ranging in scale from 10 to 100 nodes and 9 to 500 edges. Two types of networks were considered, uniformly randomly generated networks and worst case scenario line graphs where one node is a sink and all other nodes are sources. The strictly convex objective function

$$\phi_e(x^e) = \exp(x^e) + \exp(-x^e)$$

can be thought of a prohibitive cost of heavily loading a single link which encourages the network to make use of all of its resources. Across the board we observe results consistent with the unconstrained problem in [13]. The ADD-N algorithm converges one to two orders of magnitude faster than the subgradient descent type algorithms for problems large and small, sparse and dense. Figure 2 is a typical example of the convergence behavior on a sparse network. Sparse problems take considerably long to solve due to slower mixing rates for the information spreading matrix $\bar{D}^{-1}\bar{B}$. Mixing rates of markov chains are discussed extensively in [18]. In Figure 3 there is a significant improvement in convergence rate, the

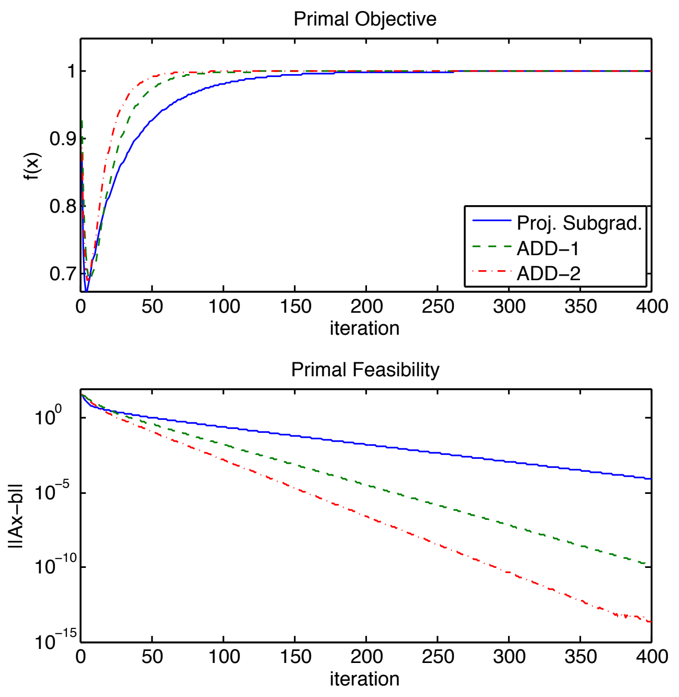

Fig. 3. ADD-1 and ADD-2 out perform projected subgradient by an order of magnitude on a denser network with 20 nodes and 100 edges with capacity constraints

time required to reach a feasibility of 10^{-4} goes from 800 iterations in the first example to only 100.

V. Conclusion

The Accelerated Dual Descent (ADD) algorithm has been generalized to the capacity constrained network flow optimization problem, through proof and computation of a generalized dual Hessian. This improvement expands the applicability of the ADD method. In particular, combining the results for the capacitated network flow problem with the stochastic generalization in [19] opens the door to networking applications via the backpressure formulation, [20], [21]. We have used the ADD as a foundation for Accelerated backpressure [22] which stabilizes queues in capacitated multi commodity communication networks with stochastic packet arrival rates.

In this work, 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. While we observe a per iteration improvement when increasing N, we do so at the cost of additional communication. Since further increases in N yield diminishing returns with respect to convergence rate, we recommend the use of ADD-1 and ADD-2 which outperform gradient descent by upwards of two orders of magnitude.

References

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

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

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

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

[5] Michal Piro and Deepankar Medhi. *Routing, Flow, and Capacity Design in Communication and Computer Networks*. Elsevier, 2004.

[6] 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.

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

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

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

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

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

[12] M. Zargham, A. Ribeiro, A. Ozdaglar, and A. Jadbabaie. Accelerated dual descent for network optimization. In *Proceedings of IEEE ACC*, 2011.

[13] M. Zargham, A. Ribeiro, A. Ozdaglar, and A. Jadbabaie. Accelerated dual descent for network optimization. *IEEE Transactions on Automatic Control*, (submitted).

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

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

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

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

[18] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Gossip algorithms: Design, analysis, and applications. In *Proceedings of IEEE INFOCOM*, 2005.

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

[20] 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.

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

[22] M. Zargham, A. Ribeiro, and A. Jadbabaie. Accelerated backpressure algorithm. *ArXiv: http://arxiv.org/abs/1302.1475*, 2013.