2012 American Control Conference
Fairmont Queen Elizabeth, Montréal, Canada
June 27-June 29, 2012

A Distributed Line Search for Network Optimization

Michael Zargham†, Alejandro Ribeiro†, Ali Jadbabaie†

***Abstract*—Dual descent methods are used to solve network optimization problems because descent directions can be computed in a distributed manner using information available either locally or at neighboring nodes. However, choosing a stepsize in the descent direction remains a challenge because its computation requires global information. This work presents an algorithm based on a local version of the Armijo rule that allows for the computation of a stepsize using only local and neighborhood information. We show that when our distributed line search algorithm is applied with a descent direction computed according to the Accelerated Dual Descent method [18], key properties of standard backtracking line search using the Armijo rule are recovered. We use simulations to demonstrate that our algorithm is a practical substitute for its centralized counterpart.**

I. Introduction

Conventional approaches to distributed network optimization are based on iterative descent in either the primal or dual domain. The reason for this is that for many types of network optimization problems there exist descent directions that can be computed in a distributed fashion. Subgradient descent algorithms, for example, implement iterations through distributed updates based on local information exchanges with neighboring nodes; see e.g., [7], [10], [12], [17]. However, practical applicability of the resulting algorithms is limited by exceedingly slow convergence rates typical of gradient descent algorithms. Furthermore, since traditional line search methods require global information, fixed stepsizes are used, exacerbating the already slow convergence rate, [14], [15].

Faster distributed descent algorithms have been recently developed by constructing approximations to the Newton direction using iterative local information exchanges, [1], [9], [18]. These results build on earlier work in [2] and [11] which present Newton-type algorithms for network flow problems that, different from the more recent versions in [9] and [18], require access to all network variables. To achieve global convergence and recover quadratic rates of centralized Newton's algorithm [9] and [18] use distributed backtracking line searches that use average consensus to verify global exit conditions. Since each backtracking line search step requires running a consensus iteration with consequently asymptotic convergence [6], [5], the exit conditions of the backtracking line search can only be achieved up to some error. Besides introducing inaccuracies, computing stepsizes with a consensus iteration is not a suitable solution because the consensus iteration itself is slow. Thus, the quadratic convergence rate of the algorithms in [9] and [18] is to some extent hampered by the linear convergence rate of the line search. This paper presents a distributed line search algorithm based on local information so that each node in the network can solve its own backtracking line search using only locally available information.

Work on line search methods for descent algorithms can be found in [16], [20], [8]. The focus in [16] and [20] is on nonmonotone line searches which improve convergent rates for Newton and Newton-like descent algorithms. The objective in [8] is to avoid local optimal solutions in nonconvex problems. While these works provide insights for developing line searches they do not tackle the problem of dependence on information that is distributed through nodes of a graph.

To simplify discussion we restrict attention to 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). Evaluating a line search algorithm requires us to choose a descent direction. We choose to work with the family of Accelerated Dual Descent (ADD) methods introduced in [18]. Algorithms in this family are parameterized by the information dependence between nodes. The Nth member of the family, shorthanded as ADD-N, relies on information exchanges with nodes not more than N hops away. Similarly, we propose a group of line searches that can be implemented through information exchanges with nodes in this N hop neighborhood.

Our work is based on the Armijo rule which is the workhorse condition used in backtracking line searches, [13, Section 7.5]. We construct a local version of the Armijo rule at each node by taking only the terms computable at that node, using information from no more than N hops away(Section III). Thus the line search always has the same information requirements as the descent direction computed via the ADD-N algorithm. Our analytical results (Section IV) leverage the information dependence properties of the algorithm to show that key properties of the backtracking line search are preserved: (i) We guarantee the selection of unit stepsize within a neighborhood of the optimal value (Section IV-A). (ii) Away from this neighborhood, we guarantee a strict decrease in the optimization objective (Section IV-B). These properties make our algorithm a practical distributed

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.

alternative to standard backtracking line search techniques. See [19] for the proofs. Simulations further demonstrate that our line search is functionally equivalent to its centralized counterpart (Section V).

II. Network Optimization

Consider a network represented by a directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ with node set $\mathcal{N} = \{1, \dots, n\}$, and edge set $\mathcal{E} = \{1, \dots, E\}$. 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

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

where $[A]_{ij}$ denotes the element in the ith row and jth column of the matrix 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 maximum reward network optimization problem is then defined as

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

Our goal is to investigate a distributed line search technique for use with Accelerated Dual Descent (ADD) methods for solving the optimization problem in (1). We begin by discussing the Lagrange dual problem of the formulation in (1) in Section II-A) and reviewing the ADD method in Section II-B.

A. Dual Formulation

Dual descent algorithms solve (1) by descending on the Lagrange dual function $q(\lambda)$. To construct the dual function consider the Lagrangian $\mathcal{L}(x, \lambda) = -\sum_{e=1}^E \phi_e(x^e) + \lambda'(Ax - b)$ and define

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

where in the last equality we wrote $\lambda' A x = \sum_{e=1}^E (\lambda' A)^e x^e$ 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 the 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. Therefore, the dual problem can be solved using any descent algorithm of the form

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

where the descent direction d_k satisfies $g_k' d_k < 0$ for all times k with $g_k = g(\lambda_k) = \nabla q(\lambda_k)$ denoting the gradient of the dual function $q(\lambda)$ at $\lambda = \lambda_k$. An important observation here is that we can compute the elements of g_k as

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

with the vector $x(\lambda_k)$ having components $x^e(\lambda_k)$ as determined by (3) with $\lambda = \lambda_k$, [3, Section 6.4]. An important fact that follows from (5) is that the ith element g_k^i of the gradient g_k can be computed using information that is either locally available $x^{(i,j)}$ or available at neighbors $x^{(j,i)}$. Thus, the simplest distributed dual descent algorithm, known as subgradient descent takes $d_k = -g_k$. Subgradient descent suffers from slow convergence so we work with an approximate Newton direction.

B. Accelerated Dual Descent

The Accelerated Dual Descent (ADD) method is a parameterized family of dual descent algorithms developed in [18]. An algorithm in the ADD family is called ADD-N and each node uses information from N-hop neighbors to compute its portion of an approximate Newton direction. Two nodes are N-hop neighbors if the shortest undirected path between those nodes is less than or equal to N.

The exact Newton direction d_k is defined as the solution of the linear equation $H_k d_k = -g_k$ where $H_k = H(\lambda_k) = \nabla^2 q(\lambda_k)$ denotes the Hessian of the dual function. We approximate d_k using the ADD-N direction defined as

$$d_k^{(N)} = -\bar{H}_k^{(N)} g_k \tag{6}$$

where the approximate Hessian inverse, $\bar{H}_k^{(N)}$ is defined

$$\bar{H}_k^{(N)} = \sum_{r=0}^N D_k^{-\frac{1}{2}} \left(D_k^{-\frac{1}{2}} B_k D_k^{-\frac{1}{2}} \right)^r D_k^{-\frac{1}{2}} \tag{7}$$

using a Hessian splitting: $H_k = D_k - B_k$ where D_k is the diagonal matrix $[D_k]_{ii} = [H_k]_{ii}$. The resulting accelerated dual descent algorithm

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

can be computed using information from N-hop neighbors because the dependence structure of g_k shown in equation (5) causes the Hessian to have a local structure as well: $[H_k]_{ij} \neq 0$ if and only if $(i,j) \in \mathcal{E}$. since H_k has the sparsity pattern of the network, B_k and thus $D_k^{-\frac{1}{2}} B_k D_k^{-\frac{1}{2}}$ must also have the sparsity pattern of the graph. Each term $D_k^{-\frac{1}{2}} \left(D_k^{-\frac{1}{2}} B_k D_k^{-\frac{1}{2}} \right)^r D_k^{-\frac{1}{2}}$ is a matrix which is non-zero only for r-hop neighbors so the sum is non-zero only for N-hop neighbors.

Analysis of the ADD-N algorithm fundamentally depends on a network connectivity coefficient $\bar{\rho}$, which is defined in [18] as the bound

$$\rho\left(B_k D_k^{-1}\right) \leq \bar{\rho} \in (0,1) \tag{9}$$

where $\rho(\cdot)$ denotes the second largest eigenvalue modulus. When $\bar{\rho}$ is small, information in the network spreads efficiently and $d_k^{(N)}$ is a more exact approximation of d_k. See [18] for details.

III. 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 N^{th} member of the family of line searches is used in conjunction with the N^{th} member of the ADD family to determine the step and descent direction in (8). 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 method to find a stepsize α that satisfies Armijo's rule. This rule requires the stepsize α to satisfy the inequality

$$q(\lambda + \alpha d) \leq q(\lambda) + \sigma \alpha d' g, \tag{10}$$

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 $q(\cdot)$ and given variable value λ and corresponding descent direction d and dual gradient g. The backtracking line search algorithm is:*

Initialize $\alpha = 1$
while $q(\lambda + \alpha d) > q(\lambda) + \sigma \alpha d' g$
  $\alpha = \alpha \beta$
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, [4, 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 takes the form

$$q_i(\lambda) = \sum_{e=(j,i)} \phi_e(x^e) - \lambda_i (a_i' x - b_i). \tag{11}$$

The vector a_i' is the i^{th} row of the incidence matrix A. Thus the local objective $q_i(\lambda)$ depends only on the flows adjacent to node i and λ^i.

An N-parameterized local Armijo rule is therefore given by

$$q_i(\lambda + \alpha_i d) \leq q_i(\lambda) + \sigma \alpha_i \sum_{j \in \mathcal{N}_i^{(N)}} d^j g^j, \tag{12}$$

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 (10), $g = \nabla q(\lambda)$ and d is a descent direction. Each node is able to compute a stepsize α_i satisfying (12) using N-hop information. The stepsize used for the dual descent update (4) is

$$\alpha = \min_{i \in \mathcal{N}} \alpha_i. \tag{13}$$

Therefore, we define the distributed backtracking line search according to the following algorithm.

Algorithm 2. *Given local objectives $q_i(\cdot)$, descent direction d and dual gradient g.*

for $i = 1 : n$
  Initialize $\alpha_i = 1$
  while $q_i(\lambda + \alpha_i d) > q_i(\lambda) + \sigma \alpha_i \sum_{j \in \mathcal{N}_i^{(N)}} d^j g^j$
    $\alpha_i = \alpha_i \beta$
  end
end
$\alpha = \min_i \alpha_i$

The scalars $\beta \in (0,1)$, $\sigma \in (0, 1/2 - \bar{\rho}^{N+1}/2)$ and $N \in \mathbb{Z}^+$ are 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. Minimum consensus of α_i requires at most diameter of $\mathcal{G}$ iterations. If each node shares its current α_i along with g_k^i with its N-hop neighbors the maximum number of iterations drops to $\lceil \text{diam}(\mathcal{G})/N \rceil$.

The parameter σ is restricted by the network connectivity coefficient $\bar{\rho}$ and the choice of N because these are scalars which encode information availability. Smaller $\bar{\rho}^{N+1}$ indicates more accessible information and thus allows for greater σ and thus a more aggressive search. As $\bar{\rho}^{N+1}$ approaches zero, we recover the condition $\sigma \in (0,1)$ from Algorithm 1.

IV. Analysis

In this section we show that when implemented with the Accelerated Dual Descent update in (8) the distributed backtracking line search defined in Algorithm 2 recovers

the key properties of Algorithm 1: strict decrease of the dual objective and selection of $\alpha = 1$ within an error neighborhood.

We proceed by outlining our assumptions. The standard Lipshitz and strict convexity assumptions regarding the dual Hessian are defined here.

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 $\|H(\lambda)^{-1}\| \leq M \qquad \forall \lambda \in \mathbb{R}^n$.

In addition to assumptions about the dual Hessian we assume that key properties of the inverse Hessian carry forward to our approximation.

Assumption 2. *The approximate inverse Hessian remains well conditioned,*

$$m \leq \|\bar{H}^{(N)}\| \leq M.$$

within the subspace $\mathbf{1}^{\perp}$.

These assumptions make sense because $\bar{H}^{(N)}$ is a truncated sum whose limit as N approaches infinity is H^{-1}, a matrix we already assume to be well conditioned on $\mathbf{1}^{\perp}$ even when solving this problem in the centralized case. Furthermore the first term in the sum is D^{-1} which is well conditioned by construction.

We begin our analysis by characterizing the stepsize α chosen by Algorithm 2 when the descent direction d is chosen according the the ADD-N method.

Lemma 1. *For any α_i satisfying the distributed Armijo rule in equation (12) with descent direction $d = -\bar{H}^{(N)}g$ we have*

$$q_i(\lambda + \alpha_i d) - q_i(\lambda) \leq 0.$$

Lemma 1 tells us that when using the distributed backtracking line search with the ADD-N algorithm, we achieve improvement in each element of the decomposed objective $q_i(\lambda)$. From the quadratic form in equation (**??**) it also follows that if equation (12) is satisfied by a stepsize α_i, then it is also satisfied by any $\alpha \leq \alpha_i$ and in particular $\alpha = \min_i \alpha_i$ satisfies equation (12) for all i.

A. Unit Stepsize Phase

A fundamental property of the backtracking line search using Armijo's rule summarized in Algorithm 1is that it always selects $\alpha = 1$ when iterates λ are within a neighborhood of the optimal argument. This property is necessary to ensure quadratic convergence of Newton's method and is therefore a desirable property for the distributed line search summarized in Algorithm 2. We prove here that this is true as stated in the following theorem.

Theorem 1. *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 algortihm [cf.* (6) *and* (7)*. If the search parameter σ is chosen such that*

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

and the norm of the dual gradient satisfies

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

then Algorithm 2 selects stepsize $\alpha = 1$.

Theorem 1 guarantees that for an appropriately chosen line search parameter σ the local backtracking line search will always choose a step size of $\alpha = 1$ once the norm of the dual gradient becomes small. Furthermore, the condition on the line search parameter tells us that $\bar{\rho}$ and our choice of N fully capture the impact of distributing the line search. The distributed Armijo rule requires $\left(1 - \bar{\rho}^{N+1} - 2\sigma\right) > 0$ while the standard Armijo rule requires $(1 - 2\sigma) > 0$. It is clear that in the limit $N \to \infty$ these conditions become the same with a rate controlled by $\bar{\rho}$.

B. Strict Decrease Phase

A second fundamental property of the backtracking line search with the Armijo rule is that there is a strict decrease in the objective when iterates are outside of an arbitrary noninfinitesimal neighborhood of the optimal solution. This property is necessary to ensure global convergence of Newton's algorithm as it ensures the quadratic convergence phase is eventually reached. Our goal here is to prove that this strict decrease can be also achieved using the distributed backtracking line search specified by Algorithm 2.

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 the same approach here and begin by finding a global lower bound on the stepsize $\hat{\alpha} \leq \alpha_i$ that holds for all nodes i. We do this in the following lemma.

Lemma 2. *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 algortihm [cf.* (6) *and* (7)*. The stepsize*

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

satisfies the local Armijo rule in (12)*, i.e.,*

$$q_i(\lambda_{k+1}) \leq q_i(\lambda_k) + \sigma\hat{\alpha} \sum_{j \in n_i^{(N)}} d_k^j g_k^j$$

for all network nodes i and all k.

We proceed with the second main result using Lemma 2, in the same manner that strict decrease is proven for the Newton method with the standard backtracking line search in [4][Section 9.5].

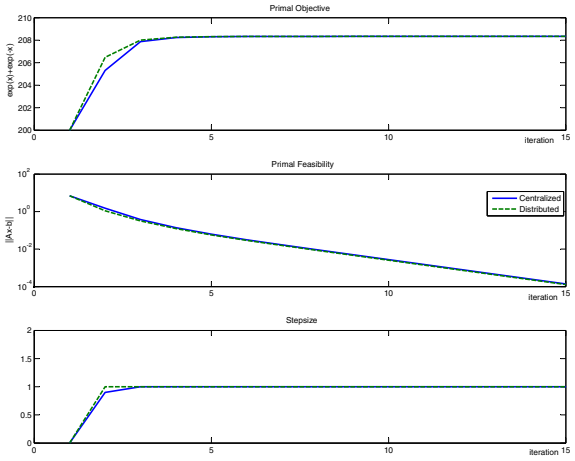

Fig. 1. The distributed line search results in solution trajectories nearly equivalent to those of the centralized line search. Top: the Primal Objective follows a similar trajectory in both cases. Middle: Primal Feasibility is achieved asymptotically. Bottom: unit stepsize is achieved in roughly the same number of steps.

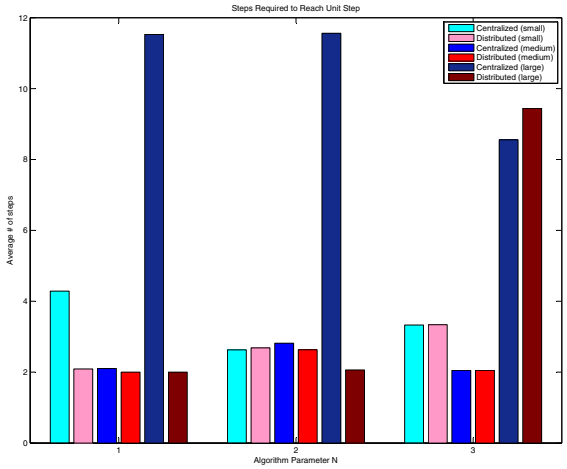

Fig. 2. The distributed line search reaches unit stepsize in 2 to 3 iterations. Fifty simulations were done for each algorithm with N=1, N=2 and N=3 and for Networks with 25 nodes and 100 edges (small), 50 nodes and 200 edges (medium) and 100 nodes and 400 edges (large).

Theorem 2. *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.* (6) *and* (7)*]. If the norm of the dual gradient is bounded away from zero as $\|g_k\| \geq \eta$, the function value at $\lambda_{k+1} = \lambda_k + \alpha_k d_k^{(N)}$ satisfies*

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

I.e., the dual function decreases by at least $\alpha\sigma m N \eta^2$

Theorem 2 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 $\beta\hat{\alpha}\sigma m N \eta^2$ while we remain outside of this neighborhood. In particular, we are guaranteed to reach a point inside the neighborhood $\|g_k\| \leq \eta = 3m/(LM^3)\left(1 - \bar{\rho}^{N+1} - 2\sigma\right)$ at which point Theorem 1 will be true and the ADD-N algorithm with the local line search becomes simply

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

This iteration is shown to have quadratic convergence properties in [18].

V. Numerical Results

Numerical experiments 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. The simulations use networks generated by selecting edges are added uniformly at random but are restricted to connected networks. The primal objective function is given by $\phi^e(x) = e^{cx^e} + e^{-cx^e}$ where c captures the notion of edge capacity. For simplicity we let $c = 1$ for all edges.

Figure 1 shows an example of a network optimization problem with 25 nodes and 100 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 roughly the same number of steps.

In Figure 2 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.

VI. Conclusion

We presented an alternative version of the backtracking line search using a local version of the Armijo rule which allows the stepsize for the dual update in the single commodity network flow problem to be computed using only local information. When this distributed backtracking line search technique is paired with the ADD method for selecting the dual descent direction we recover the key properties of

the standard centralized backtracking line search: a strict decrease in the dual objective and unit stepsize in a region around the optimal. We use simulations to demonstrate that the distributed backtracking line search is functionally equivalent to its centralized counterpart.

This work focuses on line searches when the ADD-N method is used to select the descent direction, however the proof method relies primarily on the sparsity structure of the inverse hessian approximation. This implies that our line search method could be applied with other descent directions provided they have are themselves depend only on local information.

REFERENCES

[1] S. Authuraliya and S. 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*, IEEE Transactions on Automatic Control **28** (1983), 1090–1096.

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

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

[5] M. Cao, D.A. Spielman, and A.S. Morse, *A lower bound on convergence of a distributed network consensus algorithm*, Proceedings of IEEE CDC, 2005.

[6] R. Carli, F. Fagnani, A. Speranzon, and S. Zampieri, *Communication constraints in coordinated consensus problems*, Proceedings of IEEE ACC, 2006.

[7] 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.

[8] W. Hager and H. Zhang, *A new conjugate gradient method with guaranteed descent and an efficient line search*, SIAM journal of Optimization **16** (2005), 170–192.

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

[10] 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.

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

[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] D. G. Luenberger, *Linear and nonlinear programming*, Klewer Academic Publishers, Boston, 2003.

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

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

[16] G. Di Pillo, *On nonmonotone line search*, Journal of Optimization Theory and its Applications **112**, 315–330.

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

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

[19] ———, *Accelerated dual descent for network optimization*, IEEE Transactions on Automatic Control ((submitted)).

[20] H. Zhang and W. Hager, *a nonmonotone line search technique and its application to unconstrained optimization*, SIAM journal of Optimization **14** (2004), 1043–1056.