SIAM J. OPTIM.
Vol. 20, No. 6, pp. 3260–3279

FULLY DISTRIBUTED ALGORITHMS FOR CONVEX OPTIMIZATION PROBLEMS*

DAMON MOSK-AOYAMA†, TIM ROUGHGARDEN†, AND DEVAVRAT SHAH‡

Abstract. We design and analyze a fully distributed algorithm for convex constrained optimization in networks without any consistent naming infrastructure. The algorithm produces an approximately feasible and near-optimal solution in time polynomial in the network size, the inverse of the permitted error, and a measure of curvature variation in the dual optimization problem. It blends, in a novel way, gossip-based information spreading, iterative gradient ascent, and the barrier method from the design of interior-point algorithms.

Key words. convex optimization, distributed algorithms, gradient ascent

AMS subject classifications. 90C25, 68W15

DOI. 10.1137/080743706

1. Introduction.

The development of modern networks, such as sensor and peer-to-peer networks, has stimulated interest in decentralized approaches to computational problems. These networks often have unreliable nodes with limited power, computation, and communication constraints. Frequent changes in the network topology make it hard to establish infrastructure for coordinated centralized computation. However, efficient use of network resources requires solving global optimization problems. This motivates the study of fully distributed algorithms for global optimization problems that do not rely on any form of network infrastructure.

Informally, we call an algorithm *fully distributed* with respect to a network connectivity graph G if each node of G operates without using any information beyond that in its local neighborhood in G. More concretely, we assume that each node in the network knows only its neighbors in the network, and that nodes do not have unique identifiers that can be attached to the messages that they send. This constraint is natural in networks that lack infrastructure (such as IP addresses or static GPS locations), including ad-hoc and mobile networks. It also severely limits how a node can aggregate information from beyond its local neighborhood, thereby providing a clean way to differentiate between distributed algorithms that are "truly local" and those which gather large amounts of global information at all of the nodes and subsequently perform centralized computations.

Previous work [8] observed that when every network node possesses a positive real number, the minimum of these can be efficiently computed by a fully distributed algorithm, and leveraged this fact to design fully distributed algorithms for evaluating various separable functions, including the summation function. This paper studies the significantly more difficult task of *constrained optimization* for a class of problems that capture many key operational network problems such as routing and congestion

[*] Received by the editors December 15, 2008; accepted for publication (in revised form) July 5, 2010; published electronically October 28, 2010. An announcement of the results in this paper appeared in the 21st International Symposium on Distributed Computing, Lemesos, Cyprus, September 2007.

http://www.siam.org/journals/siopt/20-6/74370.html

[†] Department of Computer Science, Stanford University, Stanford, CA 94305-5008 (damonma@cs.stanford.edu, tim@cs.stanford.edu).

[‡] Laboratory for Information and Decision Systems, Department of Electrical Engineering and Computer Science, MIT, 32-D670, 77 Massachusetts Avenue, Cambridge, MA 02139-4301 (devavrat@mit.edu).

control. Specifically, we consider a connected network of n nodes described by a *network graph* $G_N = (V, E_N)$ with $V = \{1, \ldots, n\}$. Each node is assigned a nonnegative variable x_i. The goal is to choose values for the x_i's to optimize a global network objective function under network resource constraints. We assume that the global objective function f is separable in the sense that $f(x) = \sum_{i=1}^n f_i(x_i)$. The feasible region is described by a set of nonnegative linear constraints.

Example 1 (network resource allocation). Given a capacitated network $G_N = (V, E_N)$, users wish to transfer data to specific destinations. Each user is associated with a particular path in the network, and has a utility function that depends on the rate x_i that the user is allocated. The goal is to maximize the global network utility, which is the sum of the utilities of individual users. The rate allocation $x = (x_i)$ must satisfy capacity constraints, which are linear [6].

1.1. Related work. While many previous works have designed distributed algorithms for network optimization problems, these primarily concern a different notion of locality among decision variables. In the *constraint graph* G_C of a mathematical program, the vertices correspond to variables and edges correspond to pairs of variables that participate in a common constraint. As detailed below, several previous network optimization algorithms are fully distributed with respect to this constraint graph. In contrast, the primary goal of this paper is to design fully distributed algorithms with respect to the network graph G_N. For example, in the network resource allocation problem above, the constraint graph G_C contains an edge between two users if and only if their paths intersect. Operationally, we want an algorithm for rate allocation that is distributed with respect to the true underlying network G_N. Note that typically $G_C \not\subseteq G_N$ (i.e., $E_C \not\subseteq E_N$), and hence a fully distributed algorithm with respect to G_C is *not* fully distributed with respect to G_N. An example of this distinction is provided in Figure 1.

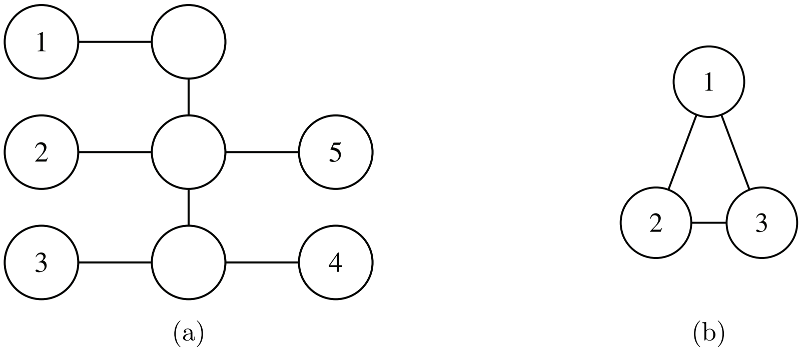

FIG. 1. (a) *A network graph for a resource allocation problem with three users. The path of user* 1 *is from node* 1 *to node* 4*, the path of user* 2 *is from node* 2 *to node* 4*, and the path of user* 3 *is from node* 3 *to node* 5*.* (b) *The constraint graph for this problem instance, which contains edges not present in the network graph.*

The design of distributed algorithms for convex minimization with linear constraints has been of interest since the early 1960's. The essence of the work before the mid-1980's is well documented in the book by Rockafellar [10]. Rockafellar [10] describes distributed algorithms for *monotropic programs*, which are separable convex minimization problems with linear constraints. These algorithms leverage the decomposable structure of the Lagrange dual problem arising from the separable primal objective. This structure has also been used to design parallel and asynchronous algorithms for monotropic programs; see the book by Bertsekas and Tsitsiklis [2] for further details. All of these algorithms are by design distributed with respect to an appropriate constraint graph G_C, as opposed to an underlying network G_N. For the special case of a network routing problem, the distributed algorithm of Gallager [4] is intuitively "closer" to being distributed with respect to G_N; however, it still requires direct access to route information and hence is only fully distributed with respect to the constraint graph G_C.

The network resource allocation problems described above are special cases of monotropic programs. Kelly, Maulloo, and Tan [6] used these known distributed algorithmic solutions to explain the congestion control protocols for the resource allocation problem. Moreover, they show that in an idealized model with perfect feedback (in the form of packet drops) by network queues, these algorithms can also be interpreted as distributed over G_N. See also Garg and Young [5] for similar results that emphasize the rate of convergence to an optimal solution. See the book by Srikant [11] for further work on congestion control.

Flow control also serves as the motivation for the work of Bartal, Byers, and Raz [1] on distributed algorithms for positive linear programming (building on earlier work by Papadimitriou and Yannakakis [9] and Luby and Nisan [7]). In this model, there is a primal agent for each primal variable and a dual agent for each dual variable (or primal constraint). In [1], direct communication is permitted between a dual agent and all of the primal agents appearing in the corresponding constraint; in this model, Bartal, Byers, and Raz [1] give a decentralized algorithm that achieves a $(1 + \epsilon)$-approximation in a polylogarithmic number of rounds.

1.2. Our contribution. The main contribution of this paper is the design and analysis of a fully distributed algorithm for a class of convex minimization problems with linear constraints. Our algorithm is distributed with respect to G_N, irrespective of the structure of G_C. The only informational assumption required is that each node (variable) knows which constraints it is involved in. As an example, our algorithm provides a fully distributed solution to the network resource allocation problem without relying on any assumptions about the network queuing dynamics.

This algorithm is simple in that it could be implemented in a network in which the communication infrastructure is limited. It effectively reduces the convex optimization problem to the problem of computing sums via the technique of dual gradient ascent. The algorithm in [8] computes sums approximately using a simple gossip-based communication protocol.

In more detail, we consider the problem of minimizing a convex separable function over linear inequalities. Given an error parameter ϵ, our algorithm produces an ϵ-approximately feasible solution with objective function value close to that of an optimal feasible solution. The running time of our algorithm is polynomial in $1/\epsilon$, the number of constraints, the inverse of the conductance of the underlying network graph, and a measure of curvature variation in the dual objective function.

We now briefly highlight our main techniques. Our algorithm is based on the Lagrange dual problem. As noted earlier, due to the separable primal objective function, the dual problem can be decomposed so that an individual node can recover the value of its variable in a primal solution from a dual feasible solution. We solve the dual problem via a dual ascent algorithm. The standard approach for designing such an algorithm only leads to a distributed algorithm with respect to the constraint graph of the problem.

Specifically, consider the setup of Figure 1 and the rate control problem considered in [6]. Here, three flows starting from nodes 1, 2, and 3 wish to send data at rates x_1, x_2, and x_3 so as to maximize $\log x_1 + \log x_2 + \log x_3$ subject to the network link capacity constraints. Let us assume that the only constraining link in Figure 1 is the link that is shared by the three flows. Following the dual-decomposition-based approach, let λ be the dual variable corresponding to the constraint imposed by this shared link capacity and involving x_1, x_2, and x_3. The values of the variables x_1, x_2, and x_3 are naturally determined by nodes 1, 2, and 3, respectively. Suppose the value of the dual variable λ is maintained by the end node of the common link shared by the three flows. To perform steps of the primal-dual algorithm, nodes 1, 2, and 3 require exact information about λ, and the common link requires a summation of the exact values of x_1, x_2, and x_3 to update λ.

In a general network setup, to update each flow variable, one requires a weighted summation of the dual variables across the links used by that particular flow. To update each dual variable, one requires a summation of the values of the flow variables that are using the corresponding link. If these summations are performed using direct communication between the nodes that maintain the values of the different variables, the algorithm will be distributed with respect to the constraint graph G_C, but not necessarily with respect to the network graph G_N. In our dual ascent algorithm, we need to overcome the following technical challenges: (a) making the algorithm distributed with respect to G_N, and (b) respecting the nonnegativity constraints on the primal variables.

To overcome the first challenge, we employ the fully distributed randomized summation algorithm from [8] as a subroutine. This leads to a fully distributed overall algorithm design. However, the randomization and approximate nature of this algorithm makes the analysis technically challenging. We overcome the second challenge by introducing a barrier function that is inspired by (centralized) interior-point mathematical programming algorithms. We believe that this barrier technique may have further applications in the design of distributed optimization algorithms.

1.3. Organization.

The remainder of this paper is organized as follows. Section 2 defines the class of problems we study, presents some of their properties, and describes the main result of this paper. Section 3 presents our distributed algorithm for solving a convex optimization problem in the class, under the assumption that certain parameters of the problem instance are known to the algorithm. An analysis of the convergence rate of the algorithm appears in section 4. Section 5 describes how to set and efficiently search for the necessary parameter values. In section 6, we discuss modifications to our algorithm, which is presented in the case of linear equality constraints, for handling linear inequality constraints.

2. Problem statement and main result.

We consider an undirected graph $G_N = (V, E_N)$ with $V = \{1, \ldots, n\}$, where each node i has a nonnegative decision variable $x_i \geq 0$. We write $\mathbf{R}$, $\mathbf{R}_+$, and $\mathbf{R}_{++}$ to denote the set of real numbers, the set of nonnegative real numbers, and the set of positive real numbers, respectively. The vector $x \in \mathbf{R}^n$ contains the variables in the optimization problem.

Throughout this paper, $\|v\|$ denotes the ℓ_2-norm of a vector $v \in \mathbf{R}^d$. The ball of radius r about the point v is denoted by $B(v, r)$ and is defined as $B(v, r) = \{w \mid \|w - v\| \leq r\}$.

We consider optimization problems of the following general form. The objective function is $f(x) = \sum_{i=1}^n f_i(x_i)$, and we assume that each $f_i : \mathbf{R}_+ \to \mathbf{R}$ has a continuous second derivative and is convex, with $\lim_{x_i \downarrow 0} f_i'(x_i) < \infty$ and $\lim_{x_i \uparrow \infty} f_i'(x_i) = \infty$.

The constraints are linear equality constraints of the form $Ax = b$, specified by a matrix $A \in \mathbf{R}_+^{m \times n}$ and a vector $b \in \mathbf{R}_{++}^m$, and nonnegativity constraints $x_i \geq 0$ on the variables. Section 6 describes modifications to our approach for handling linear inequality constraints. We assume that $m \leq n$, and the matrix A has linearly independent rows. For $i = 1, \dots, n$, let $a_i = [A_{1i}, \dots, A_{mi}]^T$ denote the ith column of the matrix A. In this distributed setting, we assume that node i is given the vectors b and a_i, but not the other columns of the matrix A.

For a real matrix M, we write $\sigma_{\min}(M)$ and $\sigma_{\max}(M)$ to denote the smallest and largest singular values, respectively, of M, so that $\sigma_{\min}(M)^2$ and $\sigma_{\max}(M)^2$ are the smallest and largest eigenvalues of $M^T M$. Note that $\sigma_{\min}(M) = \min\{\|Mv\| \mid \|v\| = 1\}$ and $\sigma_{\max}(M) = \max\{\|Mv\| \mid \|v\| = 1\}$. If M is symmetric, then the singular values and the eigenvalues of M coincide, so $\sigma_{\min}(M)$ and $\sigma_{\max}(M)$ are the smallest and largest eigenvalues of M.

We refer to the following convex optimization problem as the primal problem:

$$\text{(P)} \qquad \begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & Ax = b, \\ & x_i \geq 0, \quad i = 1, \dots, n. \end{array}$$

Let OPT denote the optimal value of (P). Associated with the primal problem (P) is the Lagrangian function $L(x, \lambda, \nu) = f(x) + \lambda^T(Ax - b) - \nu^T x$, which is defined for $\lambda \in \mathbf{R}^m$ and $\nu \in \mathbf{R}^n$, and the Lagrange dual function

$$\begin{aligned} g(\lambda, \nu) &= \inf_{x \in \mathbf{R}_+^n} L(x, \lambda, \nu) \\ &= -b^T\lambda + \sum_{i=1}^n \inf_{x_i \in \mathbf{R}_+} \left(f_i(x_i) + \left(a_i^T \lambda - \nu_i \right) x_i \right). \end{aligned}$$

The Lagrange dual problem to (P) is

$$\text{(D)} \qquad \begin{array}{ll} \text{maximize} & g(\lambda, \nu) \\ \text{subject to} & \nu_i \geq 0, \quad i = 1, \dots, n. \end{array}$$

Although we seek a solution to the primal problem (P), to avoid directly enforcing the nonnegativity constraints, we introduce a logarithmic barrier. For a parameter $\theta > 0$, we consider the primal barrier problem

$$(\text{P}_\theta) \qquad \begin{array}{ll} \text{minimize} & f(x) - \theta \displaystyle\sum_{i=1}^n \ln x_i \\ \text{subject to} & Ax = b. \end{array}$$

The Lagrange dual function corresponding to (P_θ) is

$$g_\theta(\lambda) = -b^T\lambda + \sum_{i=1}^n \inf_{x_i \in \mathbf{R}_{++}} \left(f_i(x_i) - \theta \ln x_i + a_i^T \lambda x_i \right),$$

and the associated Lagrange dual problem is the unconstrained optimization problem

$$(\text{D}_\theta) \qquad \text{maximize} \quad g_\theta(\lambda).$$

We assume that the primal barrier problem (P_θ) is feasible; that is, there exists a vector $x \in \mathbf{R}^n_{++}$ such that $Ax = b$. Under this assumption, the optimal value of (P_θ) is finite, and Slater's condition implies that the dual problem (D_θ) has the same optimal value, and there exists a dual solution λ^* that achieves this optimal value [3]. Furthermore, because (D_θ) is an unconstrained maximization problem with a strictly concave objective function, the optimal solution λ^* is unique.

2.1. Main result. The main result of this paper is a fully distributed algorithm for solving approximately the primal problem (P). A precise statement of the performance guarantees of the algorithm requires some notation that we will introduce in the description of the algorithm. The following informal theorem statement suppresses the dependence of the running time of the algorithm on some parameters.

Theorem (informal). There exists a fully distributed algorithm for the problem (P) that, for any $\epsilon > 0$ and $\theta > 0$, produces a dual vector λ and a primal solution x such that $\|Ax - b\| \leq \epsilon \|b\|$ and $f(x) \leq \mathrm{OPT} + \epsilon\|b\|\|\lambda\| + n\theta$ with high probability. The running time of the algorithm is polynomial in the number of constraints and ϵ^{-1}.

2.2. Preliminaries. For a vector of dual variables $\lambda \in \mathbf{R}^m$, let $x(\lambda) \in \mathbf{R}^n_{++}$ denote the corresponding primal minimizer in the Lagrange dual function: for $i = 1, \ldots, n$,

$$x_i(\lambda) = \arg \inf_{x_i \in \mathbf{R}_{++}} \left(f_i(x_i) - \theta \ln x_i + a_i^T \lambda x_i \right). \tag{1}$$

We can solve for each $x_i(\lambda)$ explicitly. As $f_i(x_i) - \theta \ln x_i + a_i^T \lambda x_i$ is convex in x_i,

$$f_i'(x_i(\lambda)) - \frac{\theta}{x_i(\lambda)} + a_i^T \lambda = 0. \tag{2}$$

Define $h_i : \mathbf{R}_{++} \to \mathbf{R}$ by $h_i(x_i) = f_i'(x_i) - \theta/x_i$; since f_i is convex, h_i is strictly increasing and hence has a well-defined and strictly increasing inverse. Since $\lim_{x_i \downarrow 0} f_i'(x_i) < \infty$ and $\lim_{x_i \uparrow \infty} f_i'(x_i) = \infty$, the inverse function $h_i^{-1}(y)$ is defined for all $y \in \mathbf{R}$. We now have $x_i(\lambda) = h_i^{-1}\left(-a_i^T \lambda\right)$.

Also, we assume that, given a vector λ, a node i can compute $x_i(\lambda)$. This is reasonable since computing $x_i(\lambda)$ is simply an unconstrained convex optimization problem in a single variable (1), which can be done by several methods, such as Newton's method.

Next, in our convergence analysis, we will argue about the gradient of the Lagrange dual function g_θ. A calculation shows that

$$\begin{aligned} \nabla g_\theta(\lambda) &= -b + \sum_{i=1}^n a_i x_i(\lambda) \\ &= Ax(\lambda) - b. \end{aligned} \tag{3}$$

We will use $p(\lambda)$ to denote $\|\nabla g_\theta(\lambda)\| = \|Ax(\lambda) - b\|$ for a vector $\lambda \in \mathbf{R}^m$. We note that at the optimal dual solution λ^*, we have $p(\lambda^*) = 0$ and $Ax(\lambda^*) = b$.

To control the rate of decrease in the gradient norm $p(\lambda)$, we must understand the Hessian of g_θ. For $j_1, j_2 \in \{1, \ldots, m\}$, component (j_1, j_2) of the Hessian $\nabla^2 g_\theta(\lambda)$

of g_θ at a point λ is

$$\begin{aligned}\frac{\partial g_\theta(\lambda)}{\partial \lambda_{j_1} \partial \lambda_{j_2}} &= \sum_{i=1}^{n} A_{j_1 i} \frac{\partial x_i(\lambda)}{\partial \lambda_{j_2}} \\ &= -\sum_{i=1}^{n} A_{j_1 i} A_{j_2 i} \left(h_i^{-1}\right)' \left(-a_i^T \lambda\right). \qquad (4)\end{aligned}$$

As the functions h_i^{-1} are strictly increasing, $\min_{\ell=1,\dots,n}((h_\ell^{-1})'(-a_\ell^T\lambda)) > 0$. Hence, for any $\mu \in \mathbf{R}^m$ other than the zero vector,

$$\begin{aligned}\mu^T \nabla^2 g_\theta(\lambda) \mu &= \sum_{j_1=1}^{m} \mu_{j_1} \sum_{j_2=1}^{m} \frac{\partial g_\theta(\lambda)}{\partial \lambda_{j_1} \partial \lambda_{j_2}} \mu_{j_2} \\ &= -\sum_{j_1=1}^{m} \mu_{j_1} \sum_{j_2=1}^{m} \sum_{i=1}^{n} A_{j_1 i} A_{j_2 i} \left(h_i^{-1}\right)' \left(-a_i^T \lambda\right) \mu_{j_2} \\ &= -\sum_{i=1}^{n} \left(h_i^{-1}\right)' \left(-a_i^T \lambda\right) \sum_{j_1=1}^{m} A_{j_1 i} \mu_{j_1} \sum_{j_2=1}^{m} A_{j_2 i} \mu_{j_2} \\ &= -\sum_{i=1}^{n} \left(h_i^{-1}\right)' \left(-a_i^T \lambda\right) \left(a_i^T \mu\right)^2 \\ &\le -\min_{\ell=1,\dots,n} \left(\left(h_\ell^{-1}\right)' \left(-a_\ell^T \lambda\right)\right) \left(A^T \mu\right)^T \left(A^T \mu\right) \qquad (5) \\ &< 0,\end{aligned}$$

and g_θ is a strictly concave function.

3. Algorithm description.

3.1. The basic algorithm.

3.1. The basic algorithm. We consider an iterative algorithm for obtaining an approximate solution to (P), which uses gradient ascent for the dual barrier problem (D_θ). The algorithm generates a sequence of feasible solutions λ^0, λ^1, $\lambda^2, \dots$ for (D_θ), where λ^0 is the initial vector. To update λ^{k-1} to λ^k in an iteration k, the algorithm uses the gradient $\nabla g_\theta(\lambda^{k-1})$ to determine the direction of the difference $\lambda^k - \lambda^{k-1}$. We assume that the algorithm is given as inputs the initial point λ^0, and an accuracy parameter ϵ such that $0 < \epsilon \le 1$. The goal of the algorithm is to find a point $x \in \mathbf{R}_+^n$ that is nearly feasible in the sense that $\|Ax - b\| \le \epsilon \|b\|$, and that has objective function value close to that of an optimal feasible point.

In this section, we describe the operation of the algorithm under the assumption that it has knowledge of certain parameters that affect its execution and performance. We refer to an execution of the algorithm with a particular set of parameters as an *inner run* of the algorithm. To address the fact that these parameters are not available to the algorithm at the outset, we add an *outer loop* to the algorithm. The outer loop uses binary search to find appropriate values for the parameters, and performs an inner run for each set of parameters encountered during the search. Section 5 discusses the operation of the outer loop of the algorithm.

An inner run of the algorithm consists of a sequence of iterations. Iteration k, for $k = 1, 2, \dots$, begins with a current vector of dual variables λ^{k-1}, from which each node i computes $x_i(\lambda^{k-1})$. Let $s^{k-1} = Ax\left(\lambda^{k-1}\right)$, so that, by (3), $\nabla g_\theta\left(\lambda^{k-1}\right) = s^{k-1} - b$.

In order for the algorithm to perform gradient ascent, each node must compute the vector s^{k-1}. A component $s_j^{k-1} = \sum_{i=1}^n A_{ji} x_i\left(\lambda^{k-1}\right)$ of s^{k-1} is the sum of the values $y_i = A_{ji} x_i \left(\lambda^{k-1}\right)$ for those nodes i such that $A_{ji} > 0$. As such, any algorithm for computing sums of this form that is fully distributed with respect to the underlying network G_N can be used as a subroutine for the gradient ascent. In this algorithm, the nodes apply the distributed gossip algorithm from [8] (m times, one for each component) to compute a vector $\hat{s}^{k-1}$, where $\hat{s}_j^{k-1}$ is an estimate of s_j^{k-1} for $j = 1, \ldots, m$. (The appendix recapitulates this subroutine, which is fully distributed with respect to G_N, in more detail; in the notation used there, $D = \{i \in V \mid A_{ji} > 0\}$.)

The summation subroutine takes as input parameters an accuracy ϵ_1 and an error probability δ. When used to compute s_j^{k-1} for a particular value of j, the estimate $\hat{s}_j^{k-1}$ it produces will satisfy

$$(1 - \epsilon_1)\, s_j^{k-1} \leq \hat{s}_j^{k-1} \leq (1 + \epsilon_1)\, s_j^{k-1} \tag{6}$$

with probability at least $1 - \delta$. (We discuss how to choose ϵ_1 and δ in the next section and in section 5, respectively.) In the analysis of an inner run, we assume that each invocation of the summation routine succeeds, so that (6) is satisfied. Provided we choose δ sufficiently small (see section 5), this assumption will hold with high probability.

A description of an iteration k of an inner run of the algorithm is shown in Figure 2. We specify values for the step size t and the error tolerance ϵ_1 in the next section. An inner run is essentially standard gradient ascent, where the stopping criterion (sufficiently small gradient norm) is modified to reflect the potential error in nodes' estimates of the gradient. Note that (8) does not imply (7); the nodes must check both of the two conditions, because the error tolerance ϵ_1 for the summation subroutine could be much smaller than ϵ. The summation subroutine ensures that all nodes obtain a common estimate of the sum, and as a consequence either all or no nodes will determine that both stopping conditions are met in a given iteration.

3.2. Choosing the parameters. The step size t and the convergence rate of our algorithm are governed by the variation in curvature of the Lagrange dual function.

Iteration k

1. For $j = 1, \ldots, m$, the nodes compute an estimate $\hat{s}_j^{k-1}$ of $s_j^{k-1} = \sum_{i=1}^{n} A_{ji} x_i(\lambda^{k-1})$.
2. The nodes check the following two stopping conditions:

$$(1 - \epsilon_1)\left(1 - \frac{2}{3}\epsilon\right)\|b\| \leq \left\|\hat{s}^{k-1}\right\| \leq (1 + \epsilon_1)\left(1 + \frac{2}{3}\epsilon\right)\|b\| \tag{7}$$

and

$$\left\|\hat{s}^{k-1} - b\right\| \leq \left(\frac{2}{3}\epsilon + \epsilon_1\left(\frac{1+\epsilon_1}{1-\epsilon_1}\right)\left(1 + \frac{2}{3}\epsilon\right)\right)\|b\|. \tag{8}$$

If both conditions (7) and (8) are satisfied, the inner run terminates, producing as output the vector $x\left(\lambda^{k-1}\right)$.

3. The nodes update the dual vector by setting $\Delta\lambda^{k-1} = \hat{s}^{k-1} - b$ and $\lambda^k = \lambda^{k-1} + t\Delta\lambda^{k-1}$.

FIG. 2. *The kth iteration of an inner run.*

(This is standard in a dual ascent context; intuitively, regions of large curvature necessitate a small step size to guarantee convergence, and if small steps are taken in regions with small curvature, then progress toward an optimal solution is slow.) Examining the Hessian of the Lagrange dual function in (4), we see that the curvature variation depends both on variation in $(h_i^{-1})'$, which roughly corresponds to variation in the curvature of the f_i's, and on the variation in the singular values of A^T. Precisely, note that

$$\begin{aligned}\left(h_i^{-1}\right)'\left(-a_i^T\lambda\right) &= \frac{1}{h_i'\left(h_i^{-1}\left(-a_i^T\lambda\right)\right)}\\ &= \frac{1}{f_i''\left(h_i^{-1}\left(-a_i^T\lambda\right)\right)+\frac{\theta}{\left(h_i^{-1}\left(-a_i^T\lambda\right)\right)^2}}.\end{aligned}$$

The fact that each function f_i has a continuous second derivative implies that the derivative of h_i^{-1} is continuous as well. For a distance $r > 0$, define

$$\begin{aligned}q_f(r) &= \min_{\ell=1,\ldots,n}\min_{\lambda\in B(\lambda^*,r)}\left(h_\ell^{-1}\right)'\left(-a_\ell^T\lambda\right),\\ Q_f(r) &= \max_{\ell=1,\ldots,n}\max_{\lambda\in B(\lambda^*,r)}\left(h_\ell^{-1}\right)'\left(-a_\ell^T\lambda\right).\end{aligned}$$

Our step size and convergence rate will depend on the parameters defined as follows:

$$\begin{aligned}q &= q_f\left(\left\|\lambda^0-\lambda^*\right\|\right)\sigma_{\min}\left(A^T\right)^2,\\ Q &= Q_f\left(\left\|\lambda^0-\lambda^*\right\|\right)\sigma_{\max}\left(A^T\right)^2,\\ R &= \frac{Q}{q}.\end{aligned}$$

For simplicity of notation, we have suppressed the dependence of these parameters on $\|\lambda^0-\lambda^*\|$ and the matrix A. Note that $R \geq 1$. These parameters measure the minimum and maximum curvature variation of the Lagrange dual function only in a ball of radius $\|\lambda^0-\lambda^*\|$ around the optimal dual solution λ^*; this is because the sequence of dual solutions generated by our algorithm grows monotonically closer to λ^*, as shown below in Lemma 4, and we are concerned only with variation in the region in which our algorithm executes (as opposed to the entire feasible region, which is all of $\mathbf{R}^m$). Thus a better initial estimate of the optimal dual solution yields a tighter bound on curvature variation and a better convergence result.

When we analyze the inner run, we assume that both q and Q are known to the algorithm. We discharge this assumption in section 5 using standard binary search techniques.

We define $\alpha = 1/(6R) = q/(6Q)$. For the summation subroutine, nodes use the accuracy parameter $\epsilon_1 = \epsilon\alpha/3$, where ϵ is the error tolerance given to the distributed algorithm. For gradient ascent, nodes compute and employ the following step size:

$$t = \frac{\left(1-\alpha\left(\frac{1}{2}+\frac{\epsilon}{3}\right)\right)^2-\frac{1}{6}\left(\frac{1}{2}+\frac{\epsilon}{3}\right)\left(1+\alpha\left(\frac{1}{2}+\frac{\epsilon}{3}\right)\right)}{\left(1+\alpha\left(\frac{1}{2}+\frac{\epsilon}{3}\right)\right)^2QR}. \tag{9}$$

We have $t > 0$ since $\alpha \leq 1/6$ and $\epsilon \leq 1$. Note that $t = \Theta(q/Q^2)$. An inner run continues to execute iterations for increasing values of k until both stopping conditions are satisfied, or the outer loop of the algorithm terminates the inner run as described in section 5.

4. Convergence analysis.

In this section, we provide an analysis of the number of iterations required for an inner run of the algorithm to obtain a solution $x(\lambda^k)$ such that $\|Ax(\lambda^k)-b\| \leq \epsilon\|b\|$, and we also prove an approximation bound on the objective function value of the final solution. We assume in this analysis that the summation subroutine used by an inner run is always successful; that is, (6) holds for every sum computation. Furthermore, we assume that an inner run executes until both stopping conditions are satisfied. The possibility of an inner run being terminated by the outer loop is addressed in section 5.

First, we consider the extent to which $\Delta\lambda^{k-1}$ deviates from the correct gradient $\nabla g_\theta(\lambda^{k-1})$, provided that the inner run does not terminate in iteration k. To this end, let $u^{k-1} = \hat{s}^{k-1} - s^{k-1}$ be a vector representing the error in the computation of s^{k-1}. Note that $\Delta\lambda^{k-1} = \nabla g_\theta(\lambda^{k-1}) + u^{k-1}$. The following lemma shows that the error introduced by the approximate summation subroutine is small relative to our key measure of progress, the gradient norm.

LEMMA 1. *If the stopping conditions* (7) *and* (8) *are not both satisfied in iteration k, then*

$$\left\|u^{k-1}\right\| \leq \alpha\left(\frac{1}{2}+\frac{\epsilon}{3}\right)\left\|\nabla g_\theta\left(\lambda^{k-1}\right)\right\| \tag{10}$$

and

$$\left(1-\alpha\left(\frac{1}{2}+\frac{\epsilon}{3}\right)\right)\left\|\nabla g_\theta\left(\lambda^{k-1}\right)\right\| \leq \left\|\Delta\lambda^{k-1}\right\| \leq \left(1+\alpha\left(\frac{1}{2}+\frac{\epsilon}{3}\right)\right)\left\|\nabla g_\theta\left(\lambda^{k-1}\right)\right\|. \tag{11}$$

Proof. As the inequalities in (11) follow from (10) and the triangle inequality, we focus on proving the inequality in (10). If (7) is not satisfied, then

$$\left\|\hat{s}^{k-1}\right\| < (1-\epsilon_1)\left(1-\frac{2}{3}\epsilon\right)\|b\| \text{ or } \left\|\hat{s}^{k-1}\right\| > (1+\epsilon_1)\left(1+\frac{2}{3}\epsilon\right)\|b\|,$$

and so, by (6),

$$\left\|s^{k-1}\right\| < \left(1-\frac{2}{3}\epsilon\right)\|b\| \text{ or } \left\|s^{k-1}\right\| > \left(1+\frac{2}{3}\epsilon\right)\|b\|.$$

By the triangle inequality, this implies that

$$\begin{aligned}\left\|\nabla g_\theta\left(\lambda^{k-1}\right)\right\| &= \left\|s^{k-1}-b\right\| \\ &\geq \left|\left\|s^{k-1}\right\| - \|b\|\right| \\ &> \frac{2}{3}\epsilon\|b\|. \qquad (12)\end{aligned}$$

Suppose that (7) is satisfied and (8) is not satisfied. Note that (6) implies that $\|u^{k-1}\| \leq \epsilon_1\|s^{k-1}\|$, and so (7) and (6) yield

$$\left\|u^{k-1}\right\| \leq \epsilon_1\left\|s^{k-1}\right\| \leq \epsilon_1\left(\frac{1+\epsilon_1}{1-\epsilon_1}\right)\left(1+\frac{2}{3}\epsilon\right)\|b\|. \tag{13}$$

By the triangle inequality and (13),

$$\begin{aligned}\left\|\Delta\lambda^{k-1}\right\| = \left\|\hat{s}^{k-1}-b\right\| &= \left\|\nabla g_\theta\left(\lambda^{k-1}\right)+u^{k-1}\right\| \\ &\leq \left\|\nabla g_\theta\left(\lambda^{k-1}\right)\right\| + \epsilon_1\left(\frac{1+\epsilon_1}{1-\epsilon_1}\right)\left(1+\frac{2}{3}\epsilon\right)\|b\|,\end{aligned}$$

and so the fact that (8) is not satisfied implies that

$$\begin{aligned}\left\|\nabla g_\theta\left(\lambda^{k-1}\right)\right\| &\geq \left\|\hat{s}^{k-1}-b\right\| - \epsilon_1\left(\frac{1+\epsilon_1}{1-\epsilon_1}\right)\left(1+\frac{2}{3}\epsilon\right)\|b\| \\ &> \left(\frac{2}{3}\epsilon + \epsilon_1\left(\frac{1+\epsilon_1}{1-\epsilon_1}\right)\left(1+\frac{2}{3}\epsilon\right)\right)\|b\| - \epsilon_1\left(\frac{1+\epsilon_1}{1-\epsilon_1}\right)\left(1+\frac{2}{3}\epsilon\right)\|b\| \\ &= \frac{2}{3}\epsilon\|b\|. \qquad (14)\end{aligned}$$

Combining (12) and (14), it follows that if the two stopping conditions are not both satisfied, then

$$\left\|\nabla g_\theta\left(\lambda^{k-1}\right)\right\| > \frac{2}{3}\epsilon\|b\|.$$

Now, applying the triangle inequality yields

$$\begin{aligned}\left\|u^{k-1}\right\| \leq \epsilon_1\left\|s^{k-1}\right\| \leq \epsilon_1\left(\left\|\nabla g_\theta\left(\lambda^{k-1}\right)\right\| + \|b\|\right) \leq \epsilon_1\left(1+\frac{3}{2\epsilon}\right)\left\|\nabla g_\theta\left(\lambda^{k-1}\right)\right\| \\ = \alpha\left(\frac{1}{2}+\frac{\epsilon}{3}\right)\left\|\nabla g_\theta\left(\lambda^{k-1}\right)\right\|,\end{aligned}$$

where the last equality follows from the fact that $\epsilon_1 = \epsilon\alpha/3$. This proves the inequality in (10), and completes the proof of the lemma. □

Next, we develop some inequalities that will be useful in understanding the evolution of an inner run from one iteration to the next. The following lemma translates the parameters q and Q of section 3.2 to inequalities that bound the variation in the gradient at different dual points.

LEMMA 2. *For any two points $\rho^1, \rho^2 \in B(\lambda^*, \|\lambda^0-\lambda^*\|)$,*

$$\left\|Ax\left(\rho^2\right) - Ax\left(\rho^1\right)\right\| \leq Q\left\|\rho^2-\rho^1\right\| \qquad (15)$$

and

$$\left(\nabla g_\theta\left(\rho^2\right) - \nabla g_\theta\left(\rho^1\right)\right)^T\left(\rho^2-\rho^1\right) \leq -q\left\|\rho^2-\rho^1\right\|^2. \qquad (16)$$

Proof. Let $[\rho^1,\rho^2]$ denote the line segment joining ρ^1 and ρ^2. Since $B(\lambda^*, \|\lambda^0 - \lambda^*\|)$ is a convex set, for any $i = 1,\ldots,n$ and any $\lambda \in [\rho^1,\rho^2]$, $(h_i^{-1})'(-a_i^T\lambda) \leq Q_f(\|\lambda^0-\lambda^*\|)$. As a result,

$$\begin{aligned}\left|x_i\left(\rho^2\right) - x_i\left(\rho^1\right)\right| &= \left|h_i^{-1}\left(-a_i^T\rho^2\right) - h_i^{-1}\left(-a_i^T\rho^1\right)\right| \\ &\leq Q_f\left(\left\|\lambda^0-\lambda^*\right\|\right)\left|a_i^T\left(\rho^2-\rho^1\right)\right| \\ &= Q_f\left(\left\|\lambda^0-\lambda^*\right\|\right)a_i^T\rho,\end{aligned}$$

where $\rho \in \mathbf{R}^m$ is defined by $\rho_j = |\rho_j^2 - \rho_j^1|$ for $j = 1,\ldots,m$. This implies that

$$\begin{aligned}\left\|Ax\left(\rho^2\right) - Ax\left(\rho^1\right)\right\| &= \left\|A\left(x\left(\rho^2\right) - x\left(\rho^1\right)\right)\right\| \\ &\leq Q_f\left(\left\|\lambda^0-\lambda^*\right\|\right)\left\|AA^T\rho\right\| \\ &\leq Q_f\left(\left\|\lambda^0-\lambda^*\right\|\right)\sigma_{\max}\left(AA^T\right)\|\rho\| \\ &= Q_f\left(\left\|\lambda^0-\lambda^*\right\|\right)\sigma_{\max}\left(A^T\right)^2\left\|\rho^2-\rho^1\right\|,\end{aligned}$$

and the inequality in (15) is proved.

For any $\lambda \in \left[\rho^1, \rho^2\right]$ and any $\mu \in \mathbf{R}^m$, a calculation analogous to the one in (5) yields

$$
\begin{aligned}
\mu^T \nabla^2 g_\theta(\lambda) \mu &= -\sum_{j_1=1}^{m} \mu_{j_1} \sum_{j_2=1}^{m} \sum_{i=1}^{n} A_{j_1 i} A_{j_2 i} \left(h_i^{-1}\right)' \left(-a_i^T \lambda\right) \mu_{j_2} \\
&\leq -q_f \left(\left\|\lambda^0 - \lambda^*\right\|\right) \mu^T A A^T \mu \\
&\leq -q_f \left(\left\|\lambda^0 - \lambda^*\right\|\right) \sigma_{\min}\left(A A^T\right) \mu^T \mu \\
&= -q_f \left(\left\|\lambda^0 - \lambda^*\right\|\right) \sigma_{\min}\left(A^T\right)^2 \|\mu\|^2. \qquad (17)
\end{aligned}
$$

From the second-order expansion of the function g_θ, there exist vectors $\mu^1, \mu^2 \in \left[\rho^1, \rho^2\right]$ such that

$$
g_\theta\left(\rho^2\right) = g_\theta\left(\rho^1\right) + \nabla g_\theta\left(\rho^1\right)^T \left(\rho^2 - \rho^1\right) + \frac{1}{2}\left(\rho^2 - \rho^1\right)^T \nabla^2 g_\theta\left(\mu^1\right)\left(\rho^2 - \rho^1\right)
$$

and

$$
g_\theta\left(\rho^1\right) = g_\theta\left(\rho^2\right) + \nabla g_\theta\left(\rho^2\right)^T \left(\rho^1 - \rho^2\right) + \frac{1}{2}\left(\rho^1 - \rho^2\right)^T \nabla^2 g_\theta\left(\mu^2\right)\left(\rho^1 - \rho^2\right).
$$

Adding the two above equations and applying (17) yields

$$
\begin{aligned}
&\left(\nabla g_\theta\left(\rho^2\right) - \nabla g_\theta\left(\rho^1\right)\right)^T \left(\rho^2 - \rho^1\right) \\
&= \frac{1}{2}\left(\rho^2 - \rho^1\right)^T \nabla^2 g_\theta\left(\mu^1\right)\left(\rho^2 - \rho^1\right) + \frac{1}{2}\left(\rho^1 - \rho^2\right)^T \nabla^2 g_\theta\left(\mu^2\right)\left(\rho^1 - \rho^2\right) \\
&\leq -q_f\left(\left\|\lambda^0 - \lambda^*\right\|\right) \sigma_{\min}\left(A^T\right)^2 \left\|\rho^2 - \rho^1\right\|^2.
\end{aligned}
$$

This establishes the inequality in (16) and completes the proof of the lemma. □

COROLLARY 3. *For any* $\lambda \in B(\lambda^*, \|\lambda^0 - \lambda^*\|)$, $\|\nabla g_\theta(\lambda)\| \leq Q\left\|\lambda - \lambda^*\right\|$ *and* $\nabla g_\theta(\lambda)^T\left(\lambda - \lambda^*\right) \leq -q\left\|\lambda - \lambda^*\right\|^2$.

Proof. This follows from an application of Lemma 2 with $\rho^1 = \lambda^*$ and $\rho^2 = \lambda$, using the additional observations that $\nabla g_\theta(\lambda) = Ax(\lambda) - b = Ax(\lambda) - Ax\left(\lambda^*\right)$, and $\nabla g_\theta\left(\lambda^*\right) = 0$ because λ^* is an optimal solution to (D_θ). □

We now show that the dual vector λ^k at the end of any iteration k executed by an inner run in which the stopping conditions are not satisfied is as close to the optimal solution λ^* as the vector λ^{k-1} at the start of the iteration. While this guarantee is too weak to imply directly a convergence analysis, it is necessary to justify our use of the fixed parameters q and Q throughout the entire course of the algorithm.

LEMMA 4. *For each iteration k executed by an inner run, if $\lambda^{k-1} \in B(\lambda^*, \|\lambda^0 - \lambda^*\|)$ and the stopping conditions are not satisfied, then $\|\lambda^k - \lambda^*\| \leq \|\lambda^{k-1} - \lambda^*\|$.*

Proof. Suppose the stopping conditions are not satisfied in iteration k, and so $\lambda^k - \lambda^{k-1} = t\Delta\lambda^{k-1}$. The square of the distance from λ^k to λ^* can be expressed as

$$
\begin{aligned}
\left\|\lambda^k - \lambda^*\right\|^2 &= \left\|\left(\lambda^k - \lambda^{k-1}\right) + \left(\lambda^{k-1} - \lambda^*\right)\right\|^2 \\
&= \left\|\lambda^{k-1} - \lambda^*\right\|^2 + \left\|\lambda^k - \lambda^{k-1}\right\|^2 + 2\left(\lambda^k - \lambda^{k-1}\right)^T\left(\lambda^{k-1} - \lambda^*\right) \\
&= \left\|\lambda^{k-1} - \lambda^*\right\|^2 + t^2\left\|\Delta\lambda^{k-1}\right\|^2 + 2t\left(\Delta\lambda^{k-1}\right)^T\left(\lambda^{k-1} - \lambda^*\right). \qquad (18)
\end{aligned}
$$

The third term on the right-hand side of (18) can be bounded from above by applying Corollary 3, the Cauchy–Schwarz inequality, and Lemma 1 to obtain

$$\begin{aligned}\left(\Delta\lambda^{k-1}\right)^T\left(\lambda^{k-1}-\lambda^*\right) &= \left(\nabla g_\theta\left(\lambda^{k-1}\right)+u^{k-1}\right)^T\left(\lambda^{k-1}-\lambda^*\right)\\ &\le -q\left\|\lambda^{k-1}-\lambda^*\right\|^2+\left\|u^{k-1}\right\|\left\|\lambda^{k-1}-\lambda^*\right\|\\ &\le \left\|\lambda^{k-1}-\lambda^*\right\|^2\left(\alpha\left(\frac{1}{2}+\frac{\epsilon}{3}\right)Q-q\right).\end{aligned}$$

Substituting this inequality into (18), and again applying Lemma 1 and Corollary 3 yields

$$\begin{aligned}&\left\|\lambda^k-\lambda^*\right\|^2\\ &\le \left\|\lambda^{k-1}-\lambda^*\right\|^2\left(1+t^2\left(1+\alpha\left(\frac{1}{2}+\frac{\epsilon}{3}\right)\right)^2Q^2+2t\left(\alpha\left(\frac{1}{2}+\frac{\epsilon}{3}\right)Q-q\right)\right).\end{aligned}$$

As $\alpha Q = q/6$, we will have $\|\lambda^k-\lambda^*\|\le\|\lambda^{k-1}-\lambda^*\|$ provided that

$$t \le \frac{\left(2-\frac{1}{3}\left(\frac{1}{2}+\frac{\epsilon}{3}\right)\right)q}{\left(1+\alpha\left(\frac{1}{2}+\frac{\epsilon}{3}\right)\right)^2Q^2}.$$

The step size in (9) used by an inner run satisfies this inequality because $\epsilon\le 1$. This completes the proof of the lemma. □

To establish that an inner run makes progress as it executes iterations, we show that the norm of the gradient of $g_\theta(\lambda^k)$, $p(\lambda^k)=\|Ax(\lambda^k)-b\|$, decreases by a multiplicative factor in each iteration.

Lemma 5. *For each iteration k executed by an inner run in which the stopping conditions are not satisfied,*

$$\left\|\nabla g_\theta\left(\lambda^k\right)\right\|\le\left(\sqrt{1-\frac{1}{4R^2}}\right)\left\|\nabla g_\theta\left(\lambda^{k-1}\right)\right\|.$$

Proof. If the stopping conditions are not satisfied in iteration k, then Lemma 4 implies that $\lambda^{k-1},\lambda^k\in B\left(\lambda^*,\left\|\lambda^0-\lambda^*\right\|\right)$. The squared norm of the gradient of g_θ at λ^k can be expressed as

$$\begin{aligned}\left\|\nabla g_\theta\left(\lambda^k\right)\right\|^2 &= \left\|\left(\nabla g_\theta\left(\lambda^k\right)-\nabla g_\theta\left(\lambda^{k-1}\right)\right)+\nabla g_\theta\left(\lambda^{k-1}\right)\right\|^2\\ &= \left\|\nabla g_\theta\left(\lambda^{k-1}\right)\right\|^2+\left\|\nabla g_\theta\left(\lambda^k\right)-\nabla g_\theta\left(\lambda^{k-1}\right)\right\|^2\\ &\quad+2\left(\nabla g_\theta\left(\lambda^k\right)-\nabla g_\theta\left(\lambda^{k-1}\right)\right)^T\nabla g_\theta\left(\lambda^{k-1}\right).\end{aligned}\tag{19}$$

By Lemmas 2 and 1, the second term on the right-hand side of (19) can be bounded from above by

$$\begin{aligned}\left\|\nabla g_\theta\left(\lambda^k\right)-\nabla g_\theta\left(\lambda^{k-1}\right)\right\| &= \left\|\left(Ax\left(\lambda^k\right)-b\right)-\left(Ax\left(\lambda^{k-1}\right)-b\right)\right\|\\ &= \left\|Ax\left(\lambda^k\right)-Ax\left(\lambda^{k-1}\right)\right\|\\ &\le Q\left\|\lambda^k-\lambda^{k-1}\right\|\\ &= tQ\left\|\Delta\lambda^{k-1}\right\|\\ &\le t\left(1+\alpha\left(\frac{1}{2}+\frac{\epsilon}{3}\right)\right)Q\left\|\nabla g_\theta\left(\lambda^{k-1}\right)\right\|.\end{aligned}\tag{20}$$

To bound the third term on the right-hand side of (19), we again apply Lemmas 2 and 1 to obtain

$$\begin{aligned}
&\left(\nabla g_\theta\left(\lambda^k\right) - \nabla g_\theta\left(\lambda^{k-1}\right)\right)^T \nabla g_\theta\left(\lambda^{k-1}\right) \\
&= \left(\nabla g_\theta\left(\lambda^k\right) - \nabla g_\theta\left(\lambda^{k-1}\right)\right)^T \left(\Delta\lambda^{k-1} - u^{k-1}\right) \\
&\le -tq\left\|\Delta\lambda^{k-1}\right\|^2 + \left\|u^{k-1}\right\|\left\|\nabla g_\theta\left(\lambda^k\right) - \nabla g_\theta\left(\lambda^{k-1}\right)\right\| \\
&\le -t\left(1-\alpha\left(\frac{1}{2}+\frac{\epsilon}{3}\right)\right)^2 q\left\|\nabla g_\theta\left(\lambda^{k-1}\right)\right\|^2 \\
&\qquad + t\alpha\left(\frac{1}{2}+\frac{\epsilon}{3}\right)\left(1+\alpha\left(\frac{1}{2}+\frac{\epsilon}{3}\right)\right) Q\left\|\nabla g_\theta\left(\lambda^{k-1}\right)\right\|^2 .
\end{aligned} \tag{21}$$

Substituting (20) and (21) into (19) yields

$$\begin{aligned}
&\left\|\nabla g_\theta\left(\lambda^k\right)\right\|^2 \\
&\le \left\|\nabla g_\theta\left(\lambda^{k-1}\right)\right\|^2 \left(1 + t^2\left(1+\alpha\left(\frac{1}{2}+\frac{\epsilon}{3}\right)\right)^2 Q^2\right. \\
&\qquad\qquad + 2t\left(\frac{1}{6}\left(\frac{1}{2}+\frac{\epsilon}{3}\right)\left(1+\alpha\left(\frac{1}{2}+\frac{\epsilon}{3}\right)\right)\right. \\
&\qquad\qquad\qquad \left.\left. - \left(1-\alpha\left(\frac{1}{2}+\frac{\epsilon}{3}\right)\right)^2\right) q\right),
\end{aligned}$$

where we have used the fact that $\alpha Q = q/6$. For the step size t in (9), we have

$$\begin{aligned}
&\left\|\nabla g_\theta\left(\lambda^k\right)\right\|^2 \\
&\le \left\|\nabla g_\theta\left(\lambda^{k-1}\right)\right\|^2 \left(1 - \left(\frac{\left(\left(1-\alpha\left(\frac{1}{2}+\frac{\epsilon}{3}\right)\right)^2 - \frac{1}{6}\left(\frac{1}{2}+\frac{\epsilon}{3}\right)\left(1+\alpha\left(\frac{1}{2}+\frac{\epsilon}{3}\right)\right)\right) q}{\left(1+\alpha\left(\frac{1}{2}+\frac{\epsilon}{3}\right)\right) Q}\right)^2\right).
\end{aligned}$$

Since $\alpha \le 1/6$ and $\epsilon \le 1$, it follows that

$$\left\|\nabla g_\theta\left(\lambda^k\right)\right\|^2 \le \left\|\nabla g_\theta\left(\lambda^{k-1}\right)\right\|^2 \left(1 - \left(\frac{q}{2Q}\right)^2\right),$$

and the proof is complete. □

Lemma 5 implies an upper bound on the number of iterations executed by an inner run.

THEOREM 6. *An inner run terminates after*

$$O\left(R^2 \log\left(\frac{p\left(\lambda^0\right)}{\epsilon\|b\|}\right)\right)$$

iterations with a solution $x(\lambda)$ *such that* $\|Ax(\lambda) - b\| \le \epsilon\|b\|$.

Proof. If an inner run terminates with a solution $x(\lambda)$, then the stopping conditions (7) and (8) are both satisfied for the estimate $\hat{s} = s + u$ of the vector $s = Ax(\lambda)$.

Applying (6) and the triangle inequality yields

$$\begin{aligned}\|Ax(\lambda) - b\| &= \|s - b\| \\ &\leq \|\hat{s} - b\| + \|u\| \\ &\leq \|\hat{s} - b\| + \epsilon_1 \|s\| \\ &\leq \left(\frac{2}{3}\epsilon + \epsilon_1 \left(\frac{1+\epsilon_1}{1-\epsilon_1}\right)\left(1 + \frac{2}{3}\epsilon\right)\right)\|b\| + \epsilon_1 \left(\frac{1+\epsilon_1}{1-\epsilon_1}\right)\left(1 + \frac{2}{3}\epsilon\right)\|b\| \\ &= \left(\frac{2}{3}\epsilon + \frac{2\epsilon\alpha}{3}\left(\frac{1+\epsilon_1}{1-\epsilon_1}\right)\left(1 + \frac{2}{3}\epsilon\right)\right)\|b\|.\end{aligned}$$

Because $\epsilon \leq 1$ and $\alpha \leq 1/6$, $\epsilon_1 = \epsilon\alpha/3 \leq 1/18$, and so we obtain $\|Ax(\lambda) - b\| \leq \epsilon\|b\|$.

Now, we show that if $\|s^{k-1} - b\| \leq (2/3)\epsilon\|b\|$ at the start of an iteration k, then the inner run will terminate in that iteration. Since $|\|s^{k-1}\| - \|b\|| \leq \|s^{k-1} - b\|$, (6) implies that

$$(1 - \epsilon_1)\left(1 - \frac{2}{3}\epsilon\right)\|b\| \leq \left\|\hat{s}^{k-1}\right\| \leq (1 + \epsilon_1)\left(1 + \frac{2}{3}\epsilon\right)\|b\|,$$

and (7) is satisfied. Moreover,

$$\begin{aligned}\left\|\hat{s}^{k-1} - b\right\| &\leq \left\|s^{k-1} - b\right\| + \left\|u^{k-1}\right\| \\ &\leq \frac{2}{3}\epsilon\|b\| + \epsilon_1 \left\|s^{k-1}\right\| \\ &\leq \left(\frac{2}{3}\epsilon + \epsilon_1\left(1 + \frac{2}{3}\epsilon\right)\right)\|b\| \\ &\leq \left(\frac{2}{3}\epsilon + \epsilon_1\left(\frac{1+\epsilon_1}{1-\epsilon_1}\right)\left(1 + \frac{2}{3}\epsilon\right)\right)\|b\|,\end{aligned}$$

and so (8) is satisfied as well, and the inner run terminates.

Repeated application of Lemma 5 implies that, if an inner run does not terminate in or before an iteration k, then

$$\left\|\nabla g_\theta\left(\lambda^k\right)\right\| \leq \left(1 - \frac{1}{4R^2}\right)^{\frac{k}{2}} p\left(\lambda^0\right).$$

For

$$k \geq 8R^2 \ln\left(\frac{3p\left(\lambda^0\right)}{2\epsilon\|b\|}\right),$$

we have $\|\nabla g_\theta(\lambda^k)\| \leq (2/3)\epsilon\|b\|$, and hence the stopping conditions will be satisfied and an inner run will terminate in the claimed number of iterations. □

Finally, we bound the difference between the objective function value of the solution produced by an inner run and the optimal value of the primal problem.

Corollary 7. *The objective function value of the solution $x(\lambda)$ produced by an inner run satisfies*

$$f(x(\lambda)) \leq \mathrm{OPT} + \epsilon\|b\|\|\lambda\| + n\theta.$$

Proof. Given the solution $x(\lambda)$ produced by an inner run, define a vector $\nu(\lambda) \in \mathbf{R}^n_{++}$ by, for all $i = 1, \ldots, n$,

$$\nu_i(\lambda) = \frac{\theta}{x_i(\lambda)}.$$

The pair $(\lambda, \nu(\lambda))$ is a feasible solution to the dual problem (D) with objective function value

$$\begin{aligned} g(\lambda, \nu(\lambda)) &= \inf_{x \in \mathbf{R}^n_+} L(x, \lambda, \nu(\lambda)) \\ &= -b^T\lambda + \sum_{i=1}^n \inf_{x_i \in \mathbf{R}_+} \left(f_i(x_i) + \left(a_i^T\lambda - \frac{\theta}{x_i(\lambda)} \right) x_i \right). \end{aligned}$$

As the components of the vector $x(\lambda)$ satisfy (2), we have $L(x(\lambda), \lambda, \nu(\lambda)) = g(\lambda, \nu(\lambda))$.

From the definition of the Lagrangian and the fact that $(\lambda, \nu(\lambda))$ is feasible for (D),

$$\begin{aligned} f(x(\lambda)) + \lambda^T(Ax(\lambda) - b) - \nu(\lambda)^T x(\lambda) &= L(x(\lambda), \lambda, \nu(\lambda)) \\ &= g(\lambda, \nu(\lambda)) \\ &\leq \mathrm{OPT}. \end{aligned}$$

Applying the Cauchy–Schwarz inequality and Theorem 6 yields

$$\begin{aligned} f(x(\lambda)) &\leq \mathrm{OPT} - \lambda^T(Ax(\lambda) - b) + \nu(\lambda)^T x(\lambda) \\ &\leq \mathrm{OPT} + \|\lambda\| \, \|Ax(\lambda) - b\| + \sum_{i=1}^n \left(\frac{\theta}{x_i(\lambda)} \right) x_i(\lambda) \\ &\leq \mathrm{OPT} + \epsilon \|b\| \|\lambda\| + n\theta, \end{aligned}$$

which is the claimed upper bound on the objective function value of the vector $x(\lambda)$. □

Since the dual solution λ produced by the algorithm satisfies $\|\lambda\| \leq \|\lambda^0\| + 2\|\lambda^0 - \lambda^*\|$, by choosing the parameters ϵ and θ appropriately, the approximation error can be made as small as desired (though, of course, the convergence time increases as each of these parameters decreases).

5. Setting parameters.

In this section, we consider the setting of some parameters that were assumed known by an inner run in section 4. First, we describe the outer loop of the algorithm. The purpose of the outer loop is to invoke inner runs with various parameter values, and to terminate runs if they do not end in the allotted number of iterations.

As the outer loop does not know the values q and Q, it uses binary search to choose the parameter values for the inner runs. Note that the analysis in section 4 remains valid if we replace the former quantity with a lower bound on it, and the latter quantity with an upper bound on it. Let $U > 0$ be an upper bound on the ratio between the largest and smallest possible values of these two quantities.

The outer loop enumerates $\log U$ possible values $q_1, q_2, \ldots, q_{\log U}$ for q, with $q_{\ell+1} = 2q_\ell$ for each ℓ. Similarly, it considers values $Q_1, Q_2, \ldots, Q_{\log U}$ for Q. For each pair of values (q_{ℓ_1}, Q_{ℓ_2}) such that $q_{\ell_1} \leq Q_{\ell_2}$, it computes an upper bound $T(q_{\ell_1}, Q_{\ell_2})$ on the number of iterations required for an inner run with these parameter values, using Theorem 6.

Now, the outer loop sorts the $T(q_{\ell_1}, Q_{\ell_2})$ values and executes inner runs according to this sorted order. When an inner run is executed with parameter values (q_{ℓ_1}, Q_{ℓ_2}), the outer loop lets it execute for $T(q_{\ell_1}, Q_{\ell_2})$ iterations. If it terminates due to the stopping conditions being satisfied within this number of iterations, then by Theorem 6 the solution $x(\lambda)$ produced satisfies $\|Ax(\lambda)-b\| \leq \epsilon\|b\|$, and so the outer loop outputs this solution. On the other hand, if the stopping conditions for the inner run are not satisfied within the allotted number of iterations, the outer loop terminates the inner run, and then executes the next inner run with new parameter values according to the order induced by $T(q_{\ell_1}, Q_{\ell_2})$.

By the choice of q_{ℓ_1} and Q_{ℓ_2}, there exist $q_{\ell_1^*}$ and $Q_{\ell_2^*}$ such that $q/2 \leq q_{\ell_1^*} \leq q$ and $Q \leq Q_{\ell_2^*} \leq 2Q$. For the parameter pair $(q_{\ell_1^*}, Q_{\ell_2^*})$, $T(q_{\ell_1^*}, Q_{\ell_2^*})$ is, up to constant factors, the bound in Theorem 6. Hence, when the outer loop reaches the pair $(q_{\ell_1^*}, Q_{\ell_2^*})$, the corresponding inner run will terminate with the stopping conditions satisfied in the number of iterations specified in Theorem 6. Since the inner runs executed prior to this one will also be terminated in at most this number of iterations, and there are at most $\log^2 U$ such runs, we obtain the following upper bound on the total number of iterations executed by the algorithm.

LEMMA 8. *The total number of iterations executed in all the inner runs initiated by the outer loop is*

$$O\left(R^2 \log\left(\frac{p\left(\lambda^0\right)}{\epsilon\|b\|}\right)\log^2 U\right).$$

In an iteration k of an inner run, the nodes must compute an estimate $\hat{s}_j^{k-1}$ for each of the m components of the vector s^{k-1}. As such, the summation routine must be invoked m times in each iteration. When the error probability δ satisfies $\delta \leq 1/n$, the summation algorithm in [8] computes an estimate satisfying (6) with probability at least $1-\delta$ in $O(\epsilon_1^{-2}\log^2\delta^{-1}/\Phi(P))$ time, where $\Phi(P)$ is the conductance of a doubly stochastic matrix P that determines how nodes communicate with each other. We assume here that the nodes have an upper bound N on the number of nodes in the network, and a lower bound ϕ on $\Phi(P)$. (If necessary, the trivial lower bound $\Omega(1/N^2)$ can be used for the latter parameter.) Using these bounds, the nodes can terminate the summation algorithm in $O(\epsilon_1^{-2}\log^2\delta^{-1}/\phi)$ time with an estimate $\hat{s}_j^{k-1}$ such that the probability that (6) is not satisfied is at most δ.

Given Lemma 8 and the fact that there are m summation computations per iteration, to ensure that every summation computation satisfies (6) with high probability, it suffices to set

$$\delta \leq \left(N^2 m R^2 \log\left(\frac{p\left(\lambda^0\right)}{\epsilon\|b\|}\right)\log^2 U\right)^{-1}.$$

By setting δ within a constant factor of this upper bound, and using the fact that $\epsilon_1 = \epsilon\alpha/3 = \epsilon/(18R)$, we conclude that one invocation of the summation subroutine will run in

$$O\left(\frac{R^2}{\epsilon^2\phi}\left(\log\left(NmR\log\left(\frac{p\left(\lambda^0\right)}{\epsilon\|b\|}\right)\log U\right)\right)^2\right)$$

time. Combining this with Lemma 8 yields the following upper bound on the total running time of the algorithm.

THEOREM 9. *The algorithm produces a solution $x(\lambda)$ that satisfies $\|Ax(\lambda)-b\| \leq \epsilon\|b\|$ and the objective function value bound in Corollary 7 with high probability in a total running time of*

$$O\left(\frac{mR^4}{\epsilon^2\phi}\log\left(\frac{p\left(\lambda^0\right)}{\epsilon\|b\|}\right)\log^2 U\left(\log\left(NmR\log\left(\frac{p\left(\lambda^0\right)}{\epsilon\|b\|}\right)\log U\right)\right)^2\right).$$

6. Extension to linear inequalities.

The algorithm we have presented for the primal problem (P) can be adapted to handle problems with linear inequalities of the form $Ax \leq b$. Our first step is to rewrite the inequalities as equalities by introducing slack variables z_j for the constraints. The slack variables are constrained to be nonnegative, and so the Lagrange dual function now has an additional vector of dual variables corresponding to these constraints on the slack variables. Also, the infimum in the definition of the Lagrange dual function is now taken over the slack variables z as well as the primal variables x. To ensure that the infimum is finite, we introduce an additional term $\psi(z) = \sum_{j=1}^m \psi_j(z_j)$ into the objective function for the slack variables. The functions ψ_j must satisfy the same technical conditions as the functions f_i. An example of a natural choice of $\psi_j(z_j)$ is a quadratic function in z_j.

We introduce logarithmic barriers for the nonnegativity constraints on both the primal and the slack variables to obtain the primal barrier problem

$$\begin{array}{ll}\text{minimize} & f(x) - \theta\sum_{i=1}^n \ln x_i + \psi(z) - \theta\sum_{j=1}^m \ln z_j \\ \text{subject to} & Ax + z = b.\end{array}$$

The corresponding Lagrange dual function g_θ has the gradient

$$\nabla g_\theta(\lambda) = Ax(\lambda) + z(\lambda) - b,$$

where $z(\lambda)$ is the vector defined by

$$z_j(\lambda) = \arg\inf_{z_j\in\mathbf{R}_{++}}\left(\psi_j(z_j) - \theta\ln z_j + \lambda_j z_j\right)$$

for all $j = 1, \ldots, m$.

In the basic algorithm, the nodes now perform gradient ascent for the new dual barrier problem. Computation of the gradient in each iteration requires the nodes to compute the vector $z\left(\lambda^{k-1}\right)$ for the current dual vector λ^{k-1}. This can be accomplished by each node if it is given the functions ψ_j for all j. The stopping conditions that the nodes check to determine whether the approximate feasibility guarantee is satisfied are modified to account for the additional term $z(\lambda)$ in the gradient.

When an inner run terminates with a solution $x(\lambda)$ such that $\|Ax(\lambda)+z(\lambda)-b\| \leq \epsilon\|b\|$, the corresponding approximate feasibility bound for the inequality constraints is $\|v(x(\lambda))\| \leq \epsilon\|b\|$, where the vector v is defined by $v_j(x(\lambda)) = \max((Ax(\lambda))_j - b_j, 0)$ for all $j = 1, \ldots, m$. The addition of the $\psi(z)$ term to the primal objective function leads to further error in the approximate guarantee on the value of $f(x(\lambda))$ relative to OPT, although this error can be made small by appropriate choice of the functions ψ_j. Choosing the functions ψ_j to minimize this error will affect the running time of the algorithm, however, because the definitions of the curvature variation parameters q and Q now have additional terms that account for variation in the curvature of the ψ_j functions.

7. Discussion.

We presented a randomized, distributed algorithm with a polynomial running time for solving a convex minimization (or concave maximization) problem with a separable convex objective function and linear constraints. The key contribution lies in making the existing dual-decomposition-based algorithm distributed with respect to the communication network graph rather than the constraint network graph. The analysis presented here could be of interest in many other contexts. For example, we strongly believe that it can be adapted to recover most of the known results on convergence bounds for a class of stochastic gradient algorithms with varying step size.

The algorithm presented here requires some amount of synchronization to select the correct step size. Removing this synchronization and thus making the algorithm asynchronous could be an interesting direction for future research.

Appendix. Description of summation subroutine.

For completeness, we describe the summation subroutine from [8] used by the algorithm in this paper. Suppose that, for a subset of nodes $D \subseteq V$, each node $i \in D$ has a number $y_i > 0$. The nodes wish to compute $y = \sum_{i \in D} y_i$. Each node $i \in D$ generates a random variable W^i (independently of the other nodes) that is distributed according to the exponential distribution with rate y_i (mean $1/y_i$). The approximate summation algorithm is based on the following property of exponential random variables.

Proposition 10. *The random variable*

$$\bar{W} = \min_{i \in D} W^i$$

is distributed as an exponential random variable of rate y.

This suggests that $1/\bar{W}$ is a reasonable estimate of y. To reduce the variance of the estimate, the nodes repeat this procedure c times, where c is an integer parameter, and average the results. That is, each node $i \in D$ generates independent exponential random variables $W_1^i, \ldots, W_c^i$, the nodes compute $\bar{W}_\ell = \min_{i \in D} W_\ell^i$ for $\ell = 1, \ldots, c$, and each node outputs

$$\hat{y} = \frac{c}{\sum_{\ell=1}^{c} \bar{W}_\ell}$$

as an estimate of y.

To compute the minima $\bar{W}_\ell$, the nodes use a *gossip* algorithm, which proceeds in a sequence of rounds. Each node i maintains a collection of numbers $\hat{W}_1^i, \ldots, \hat{W}_c^i$, where $\hat{W}_\ell^i$ is an estimate of $\bar{W}_\ell$ for $\ell = 1, \ldots, c$. Initially, $\hat{W}_\ell^i = W_\ell^i$ if $i \in D$, and $\hat{W}_\ell^i = \infty$ otherwise. In any round, each node i contacts a neighbor j in the graph with probability P_{ij}, so the communication pattern is described by a stochastic $n \times n$ matrix P. When two nodes i and j communicate, they update their estimates of $\bar{W}_\ell$ according to

$$\hat{W}_\ell^i, \hat{W}_\ell^j \leftarrow \min(\hat{W}_\ell^i, \hat{W}_\ell^j)$$

for $\ell = 1, \ldots, c$. In any round, node i uses its estimates $\hat{W}_\ell^i$ of $\bar{W}_\ell$ to compute an estimate

$$\hat{y}_i = \frac{c}{\sum_{\ell=1}^{c} \hat{W}_\ell^i}$$

of y.

The summation algorithm takes as accuracy parameters two inputs $\epsilon_1 \in (0,1)$ and $\delta \in (0,1)$. When $c = \Theta(\epsilon_1^{-2}(1+\log\delta^{-1}))$ and P is doubly stochastic, the amount of time required for the estimates of the nodes to achieve an accuracy of $(1 \pm \epsilon_1)$ is bounded as follows. Let

$$\Phi(P) = \min_{S\subset V, 0<|S|\le n/2} \frac{\sum_{i\in S, j\notin S} P_{ij}}{|S|}$$

be the conductance of the matrix P.

THEOREM 11 (see [8]). *After*

$$O\left(\epsilon_1^{-2}\left(1+\log\delta^{-1}\right)\left(\frac{\log n + \log\delta^{-1}}{\Phi(P)}\right)\right)$$

time, the probability that the estimate $\hat{y}_i$ *of any node* i *is not in* $[(1-\epsilon_1)y, (1+\epsilon_1)y]$ *is at most* δ*.*

REFERENCES

[1] Y. BARTAL, J. W. BYERS, AND D. RAZ, *Fast, distributed approximation algorithms for positive linear programming with applications to flow control*, SIAM J. Comput., 33 (2004), pp. 1261–1279.

[2] D. P. BERTSEKAS AND J. N. TSITSIKLIS, *Parallel and Distributed Computation: Numerical Methods*, 2nd ed., Prentice-Hall, Upper Saddle River, NJ, 1989.

[3] S. BOYD AND L. VANDENBERGHE, *Convex Optimization*, Cambridge University Press, Cambridge, UK, 2004.

[4] R. G. GALLAGER, *A minimum delay routing algorithm using distributed computation*, IEEE Trans. Comm., 25 (1977), pp. 73–85.

[5] N. GARG AND N. E. YOUNG, *On-line end-to-end congestion control*, in Proceedings of the 43rd Annual Symposium on Foundations of Computer Science, IEEE Computer Society, Los Alamitos, CA, IEEE, Washington, DC, 2002, pp. 303–312.

[6] F. P. KELLY, A. K. MAULLOO, AND D. K. H. TAN, *Rate control for communication networks: Shadow prices, proportional fairness and stability*, J. Oper. Res. Soc., 49 (1998), pp. 237–252.

[7] M. LUBY AND N. NISAN, *A parallel approximation algorithm for positive linear programming*, in Proceedings of the 25th Annual ACM Symposium on Theory of Computing, ACM, New York, 1993, pp. 448–457.

[8] D. MOSK-AOYAMA AND D. SHAH, *Fast distributed algorithms for computing separable functions*, IEEE Trans. Inform. Theory, 54 (2008), pp. 2997–3007.

[9] C. H. PAPADIMITRIOU AND M. YANNAKAKIS, *Linear programming without the matrix*, in Proceedings of the 25th Annual ACM Symposium on Theory of Computing, ACM, New York, 1993, pp. 121–129.

[10] R. T. ROCKAFELLAR, *Network Flows and Monotropic Optimization*, John Wiley & Sons, New York, 1984.

[11] R. SRIKANT, *The Mathematics of Internet Congestion Control (Systems and Control: Foundations and Applications)*, Birkhäuser Boston, Boston, 2004.