Distributed and Secure Computation of Convex Programs over a Network of Connected Processors

Michael J. Neely
University of Southern California
http://www-rcf.usc.edu/∼mjneely

***Abstract*— We consider the fundamental problem of optimizing a convex function subject to a collection of convex inequality constraints and set constraints. An iterative algorithm is developed that solves the problem to within any desired accuracy using a network of distributed processors. The network is assumed to form a connected graph. Each processing node of the graph takes charge of a portion of the original constraints and solves a correspondingly less complex problem, passing key values to neighboring nodes. The constraints can be assigned to nodes arbitrarily, and individual nodes do not require knowledge of the network topology or the constraints assigned to other nodes. Further, we assume that each node has a set of *private optimization variables* that participate in the global optimization problem but are unknown to other nodes of the graph. This establishes a general framework for computational load sharing and secure optimization over a network.**

***Index Terms*— Distributed Computing, Optimization, Privacy, Network Security**

I. INTRODUCTION

We consider the fundamental problem of finding the minimum value of a multi-variable convex function subject to a set of convex inequality constraints. Such *convex programs* have numerous applications, particularly in the area of data networks, and a variety of computational algorithms exist for solving them [1][2][3][4][5]. In this paper, we develop a novel technique for distributing the computation over a connected graph of network processors. Each processing node takes charge of a subset of the original problem constraints, and iteratively solves a simplified problem involving only this subset. Problem parameters are updated at every iteration based on message passing between neighboring nodes. Further, we assume each processing node has a set of *private optimization variables* that participate in the global optimization problem but must be kept hidden from other nodes of the graph. The resulting algorithm yields a solution that is arbitrarily close to the global optimal solution, where proximity to optimality is controlled by a parameter V that affects a tradeoff in the required computation time.

These results contribute to the growing field of *distributed computing*. This field has received a considerable amount of attention in recent years due to the inherent inter-networking capabilities of modern computers and the numerous computationally intensive experiments performed by the scientific community. Recent theoretical results consider sorting, ordering, and averaging over graphs [6] [7], and distributed computation of matrix eigenvalues is considered in [8]. Related work considers implementation of parallel algorithms over mesh networks [9] [10] [11] and distributed computation with asynchronous updates [12].

Algorithms for solving convex programs over multiple parallel processors have been developed previously in [5] [3] using a primal-dual methodology, and distributed algorithms for solving linear programs on a network have been recently considered in [2] in the case when each linear inequality constraint involves only local variables. Distributed nonlinear optimization for stochastic network flow problems is considered in [13] [14] [15]. In this paper, we develop distributed solutions to convex optimization problems with any number of variables and with general constraint sets. Our contributions are threefold: First, we develop a distributed algorithm that operates over any connected graph of processors. Inequality constraints can be assigned to processors arbitrarily to ensure an equitable sharing of system resources, and individual processors do not require knowledge of the constraints of other processors. Second, we consider the issue of *secure optimization*, where a global optimum is attained without requiring individual processors to reveal their private optimization variables. Third, we present our results in terms of *Lyapunov drift theory*, which deviates significantly from the traditional primal-dual approach to convex optimization and simplifies the analysis.

The outline of this paper is as follows. In the next section we introduce the optimization problem, and in Section III we present the distributed optimization algorithm. In Section IV we introduce Lyapunov drift theory and prove the performance bounds.

II. Problem Formulation

Consider an undirected graph with K nodes and L links, where nodes represent processors and links represent inter-processor communication channels. We assume the graph is *connected*, so that there is a path from any node to any other node. For each node $k \in \{1, \ldots, K\}$, let N_k represent the set of *neighboring nodes*, that is, N_k consists of all nodes j such that there is a link between nodes k and j. We desire to use this graph of processors to compute the solution to the following convex optimization problem:

Problem A:

$$\begin{aligned} \text{Minimize:} \quad & \textstyle\sum_{k=1}^{K} g_k(\vec{x}, \vec{p}_k) \\ \text{Subject to:} \quad & \vec{f}_k(\vec{x}, \vec{p}_k) \leq \vec{b}_k \quad \text{for } k \in \{1, \ldots, K\} \\ & \vec{x} \in \Omega \cap \Omega_1 \cap \ldots \cap \Omega_K \qquad (1) \\ & \vec{p}_k \in \Theta_k \quad \text{for } k \in \{1, \ldots, K\} \qquad (2) \end{aligned}$$

where $\vec{x}$ is a vector of *public variables* in $\mathbb{R}^M$ for some integer M, $\vec{p}_k$ is a vector of *private variables* in $\mathbb{R}^{M_k}$ for some integer M_k for each k, $g_k(\cdot), \vec{f}_k(\cdot)$ are convex functions of their multi-variable arguments (where we define a vector valued function to be convex if each component function is convex), $\Omega, \Omega_1, \ldots, \Omega_K$ are convex subsets of $\mathbb{R}^M$, and $\Theta_1, \ldots, \Theta_K$ are convex subsets of $\mathbb{R}^{M_1}$, ..., $\mathbb{R}^{M_K}$, respectively.

To prevent infinite solutions to the above optimization, we assume that all sets are compact and all functions are bounded. Further, we make the following *non-negativity assumptions*: We assume that $\vec{b}_k > 0$ and $g_k(\cdot) \geq 0, \vec{f}_k(\cdot) \geq 0$ for all $k \in \{1, \ldots, K\}$ (where the inequalities are taken entrywise), and that the set Ω restricts the public variables $\vec{x}$ to having non-negative entries. It is not difficult to show that general convex optimization problems with bounded functions over compact sets can be written as optimizations that conform to the non-negativity assumptions.[1]

Note that we have defined K sets of inequalities so that we can assign each set to a particular processor. In the case when there are no private optimization variables, the particular assignment of inequalities to processors is arbitrary. However, the private optimization variables $\vec{p}_k$ and their constraint sets Θ_k represent decision variables and constraints that participate in the global optimization, but which must be kept hidden from all other processors. Such a problem arises, for example, when the private variables represent prices or consumption levels that a particular individual does not wish to reveal. We now transform the optimization problem into a form that is more conducive to distributed implementation.

We first designate node 1 as the *root node* of the graph, and form the *shortest path tree* from all nodes to this root node 1. Specifically, each node $i \geq 2$ is assigned a *parent node* $Par(i)$ from its set of neighbors, and the sequence of successive parents of a given node i terminates at the root node 1 and forms a shortest hop path from node i to node 1. Such trees always exist in connected graphs, and simple distributed algorithms for constructing them are given in [16]. We let $Child(i)$ represent the set of all "children" nodes of a given node i, that is, $Child(i)$ is the set of all nodes j such that $Par(j) = i$.

Problem B:

$$\begin{aligned} \text{Minimize:} \quad & \textstyle\sum_{k=1}^{K} g_k(\vec{x}_k, \vec{p}_k) \\ \text{Subject to:} \quad & \vec{f}_k(\vec{x}_k, \vec{p}_k) \leq \vec{b}_k \quad \text{for } k \in \{1, \ldots, K\} \quad (3) \\ & \vec{x}_k \in \Omega \cap \Omega_k \quad \text{for } k \in \{1, \ldots, K\} \\ & \vec{p}_k \in \Theta_k \quad \text{for } k \in \{1, \ldots, K\} \\ & \vec{x}_k \geq \vec{x}_{Par(k)} \quad \text{for } k \in \{2, \ldots, K\} \quad (4) \\ & \vec{x}_k \leq \vec{x}_{Par(k)} \quad \text{for } k \in \{2, \ldots, K\} \quad (5) \end{aligned}$$

The constraints (4) and (5) imply that children and parents have the same $\vec{x}_k$ values. Because the graph is connected, this implies that $\vec{x}_1 = \vec{x}_2 = \ldots = \vec{x}_K$. We thus have the following simple lemma, the proof of which is straightforward and omitted for brevity.

Lemma 1: The vector $(\vec{x}^*, \vec{p}_1^*, \ldots, \vec{p}_K^*)$ is an optimal solution to Problem A if and only if $(\vec{x}^*, \ldots, \vec{x}^*, \vec{p}_1^*, \ldots, \vec{p}_K^*)$ is an optimal solution to Problem B. □

A. The Interior Point Assumption

To facilitate analysis, it is useful to assume that there exists a vector $(\vec{x}, \vec{p}_1, \ldots, \vec{p}_K)$ satisfying the set constraints (1) and (2), and such that $\vec{f}_k(\vec{x}, \vec{p}_k) < \vec{b}_k$ for all k. That is, there exists a point that satisfies all inequalities with strict inequality (note that we do not require the *optimal* point to have this property). We define ϵ_{max} as the maximum value of ϵ such that there exists a vector $(\vec{x}, \vec{p}_1, \ldots, \vec{p}_K)$ satisfying (1) and (2) and additionally satisfying $\vec{f}_k(\vec{x}, \vec{p}_k) \leq \vec{b}_k - \vec{\epsilon}$ for all k (where $\vec{\epsilon}$ is a vector with all entries equal to ϵ). It is not difficult to show that, given the existence of a positive value of ϵ_{max}, there must exist a sequence of vectors $\vec{x}^{(\epsilon)}, \vec{p}_k^{(\epsilon)}$ parameterized by positive values $\epsilon \leq \epsilon_{max}$ that satisfy the set constraints (1) and (2) and such that:

$$\vec{f}_k(\vec{x}^{(\epsilon)}, \vec{p}_k^{(\epsilon)}) \leq \vec{b}_k - \vec{\epsilon} \quad \text{for } 0 < \epsilon \leq \epsilon_{max} \qquad (6)$$

while $\vec{x}^{(\epsilon)} \to \vec{x}^*$, $\vec{p}_k^{(\epsilon)} \to \vec{p}_k^*$ as $\epsilon \to 0$, where $\vec{x}^*$ and $\vec{p}_k^*$ represent the optimal solution vectors for Problem A.

[1] Indeed, all that is required to modify the general problem to meet the non-negativity assumptions is to add a sufficiently large positive constant to both sides of every inequality and make an appropriate change of variables.

III. Distributed and Secure Optimization

We now present a distributed algorithm for computing a solution that is arbitrarily close to the optimal solution of Problem B (and hence, Problem A). In particular, each network node k takes charge of the inequality constraints $\vec{f}_k(\vec{x}_k, \vec{p}_k) \leq \vec{b}_k$ and the set constraints $\vec{x}_k \in \Omega \cap \Omega_k$ and $\vec{p}_k \in \Theta_k$. A sequence of vector values $\{\vec{x}_k[0], \vec{x}_k[1], \ldots, \vec{x}_k[t]\}$ and $\{\vec{p}_k[0], \ldots, \vec{p}_k[t]\}$ is computed over t iterations, the average of which approaches the desired solution. We note that the vector $\vec{x}_k[t]$ can be viewed as the estimate of the public variables at node k at time t.

To define the algorithm, let $V > 0$ be a control parameter that affects algorithm performance, and define the sequence $\delta[t] \triangleq 1/\sqrt{1+t}$ for $t \in \{0, 1, \ldots\}$. Furthermore, we define *violation sequences* $\vec{U}_k[t]$, $\vec{Y}_k[t]$, and $\vec{Z}_k[t]$ as follows: Let $\vec{U}_k[0] = \vec{0}$ for $k \in \{1, \ldots, K\}$, and let $\vec{Y}_k[0] = \vec{Z}_k[0] = 0$ for $k \in \{2, \ldots, K\}$. On every iteration t we update the $\vec{U}_k[t]$ sequences for each node $k \in \{1, \ldots, K\}$ as follows:

$$\vec{U}_k[t+1] = \max[\vec{U}_k[t] - \vec{b}_k, 0] + \vec{f}_k(\vec{x}_k[t], \vec{p}_k[t]) \quad (7)$$

Likewise, for nodes $k \in \{2, \ldots, K\}$ we have:

$$\vec{Y}_k[t+1] = \max[\vec{Y}_k[t] - \vec{x}_k[t] - \vec{\delta}[t], 0] + \vec{x}_{Par(k)}[t] \quad (8)$$

$$\vec{Z}_k[t+1] = \max[\vec{Z}_k[t] - \vec{x}_{Par(k)}[t] - \vec{\delta}[t], 0] + \vec{x}_k[t] \quad (9)$$

where the values of $\vec{p}_k[t]$ and $\vec{x}_k[t]$ are computed as defined below, and where $\vec{\delta}[t]$ represents a vector with all entries equal to $\delta[t]$. The $\vec{U}_k[t]$, $\vec{Y}_k[t]$, and $\vec{Z}_k[t]$ vectors are analogous to a sequence of *slack variables* in a dual solution to the convex optimization problem of interest, but can intuitively be viewed as *queue backlogs* in a slotted queueing system with arrivals and departures determined by the control decision variables $\vec{x}_k[t], \vec{p}_k[t]$. Indeed, our algorithm below is inspired by the stable queue control policies of [17] [14] [18] [15]. Specifically, if the $\vec{U}_k[t]$, $\vec{Y}_k[t]$, and $\vec{Z}_k[t]$ queue backlogs are kept bounded, then it must be the case that the time average "input rate" to each queue is less than or equal to the time average "service rate," so that inequality constraints (3), (4), and (5) are satisfied. We note that the positive $\delta[t]$ sequence is defined to allow the server rate of the $\vec{Y}_k[t]$ and $\vec{Z}_k[t]$ queues of (8) and (9) to be slightly larger than the input rate to allow for stability, although this margin decreases to zero with increased iterations.

For the following iterative algorithm, it is useful to define the vector $\vec{H}_k[t]$ for each node $k \geq 2$ as follows:

$$\vec{H}_k[t] = \vec{Y}_k[t] - \vec{Z}_k[t] \quad (10)$$

The Iterative Algorithm: On iteration t, every node $k \geq 2$ transmits its $\vec{H}_k[t]$ vector to its parent. Each node $k \in \{1, \ldots, K\}$ then computes $\vec{x}_k[t]$ and $\vec{p}_k[t]$ as solutions to the following optimization:

$$\begin{aligned} \text{Minimize:} \quad & V g_k(\vec{x}_k, \vec{p}_k) + 2\vec{U}_k[t] \cdot \vec{f}_k(\vec{x}_k, \vec{p}_k) \\ & -2\vec{x}_k \cdot \left(\vec{H}_k[t] - \textstyle\sum_{j \in Child(k)} \vec{H}_j[t]\right) \\ \text{Subject to:} \quad & \vec{p}_k \in \Theta_k \\ & \vec{x}_k \in \Omega \cap \Omega_k \end{aligned}$$

where the vector multiplication represents the standard dot product (i.e., a sum of the products of each entry of the two vectors being multiplied). Each node k then transmits its vector $\vec{x}_k[t]$ to all of its children (defined as the set $Child(k)$). Note that $Child(k)$ is defined as the empty set if a node has no children. The violation sequences $\vec{U}_k[t]$, $\vec{Y}_k[t]$, $\vec{Z}_k[t]$ are then updated according to (7)-(9).

Note that each node only requires knowledge of its own constraints and constraint sets, and that the private variables $\vec{p}_k[t]$ are known only to their corresponding nodes k. It is not difficult to show that these private variables cannot be inferred by other nodes if these nodes do not have exact knowledge of the individual $\vec{f}_k(\vec{x}_k, \vec{p}_k)$ functions. Indeed, if a particular node j replaces $\vec{f}_j(\vec{x}_j, \vec{p}_j)$ and Θ_j by a new function $\vec{f}_j(\vec{x}_j, \frac{\vec{p}_j}{2})$ and a new constraint set $2\Theta_j$, the resulting message passing between neighbors will be exactly the same, although the magnitudes of the private variables $\vec{p}_j[t]$ will be doubled.

It is interesting to note that the above iterative algorithm is similar to a classical subgradient search algorithm on the dual optimization of Problem B (see, for example, [1]), where the stepsize is normalized to 1 unit and the cost function is scaled by the control parameter V. However, the algorithm is inspired by minimizing the drift of a quadratic Lyapunov function of the violation sequences $\vec{U}_k[t], \vec{Y}_k[t], \vec{Z}_k[t]$, rather than by the classical primal-dual methodology. One advantage of the Lyapunov approach is that it yields a sequence of improving solution estimates obtained by *time averages* of the $\vec{x}_k[t], \vec{p}_k[t]$ variables, and does not require a global evaluation of the cost function.

Specifically, we define empirical averages of the $\vec{x}_k[t]$ and $\vec{p}_k[t]$ sequences as follows:

$$\vec{x}_k^{av}[t] \triangleq \frac{1}{t}\sum_{\tau=0}^{t-1} \vec{x}_k[\tau] \quad , \quad \vec{p}_k^{av}[t] \triangleq \frac{1}{t}\sum_{\tau=0}^{t-1} \vec{p}_k[\tau] \quad (11)$$

Define $(\vec{x}^*, \vec{p}_1^*, \ldots, \vec{p}_K^*)$ as the optimal solution vector of Problem A, and define g^* as the corresponding optimal cost. Let G_{max} represent the maximum value of $\sum_{k=1}^K g_k(\vec{x}, \vec{p}_k)$ over the feasible vectors satisfying the constraints of Problem A. Further define F_{max} as

the maximum value of $||\vec{f}_k(\vec{x}_k, \vec{p}_k)||^2$ over all $k \in \{1, \ldots, K\}$ and over all vectors $(\vec{x}_k, \vec{p}_k)$ satisfying $\vec{x}_k \in \Omega \cap \Omega_k$, $\vec{p}_k \in \Theta_k$. Define $B_{max} \triangleq \max_k ||\vec{b}_k||^2$.

Theorem 1: (Algorithm Performance) If the optimization Problem A satisfies the *interior point assumption* of Section II-A, then for all iterations t, the above iterative algorithm yields vectors $(\vec{x}_k^{av}[t], \vec{p}_k^{av}[t])$ with a cost that satisfies:

$$\sum_{k=1}^{K} g_k(\vec{x}_k^{av}[t], \vec{p}_k^{av}[t]) \leq g^* + \frac{C}{V} \quad (12)$$

Further, $\vec{x}_k^{av}[t] \in \Omega \cap \Omega_k$ and $\vec{p}_k^{av}[t] \in \Theta_k$ for all k and all t, and the following inequality constraints are satisfied:

$$\vec{f}_k(\vec{x}_k^{av}[t], \vec{p}_k^{av}[t]) \leq \vec{b}_k + \frac{C+VKG_{max}}{2\epsilon_{max} t} + \frac{\epsilon_{max}}{t} \quad (13)$$

$$\left| \vec{x}_k^{av}[t] - \vec{x}_{Par(k)}^{av}[t] \right| \leq \frac{2}{\sqrt{t}} + \frac{(C+VKG_{max})\sqrt{t+1}}{2t} \quad (14)$$

where C is defined:

$$C \triangleq K \left(2 \max_{\vec{x} \in \Omega} ||\vec{1} + \vec{x}||^2 + 2 \max_{\vec{x} \in \Omega} ||\vec{x}||^2 + B_{max} + F_{max} \right)$$

Hence, a value of V can be selected so that the vectors $\vec{x}_k^{av}[t], \vec{p}_k^{av}[t]$ have a resulting cost that is arbitrarily close to the minimizing cost g^*. The algorithm can be run for a number of iterations t until the constraints (13) and (14) are arbitrarily close to the constraints (3)-(5). The proof of this theorem is provided in the next section.

A. Discussion

Note that the above algorithm is inherently distributed, where each processor communicates only with its neighboring processor, as specified by the underlying graph structure. Indeed, each node k maintains its own "estimate" $\vec{x}_k[t]$ of the public variables $\vec{x}$, and the set constraint $\vec{x} \in \Omega \cap \Omega_1 \cap \ldots \cap \Omega_K$ is enforced locally by restricting each estimate $\vec{x}_k[t]$ to the local set constraint $\Omega \cap \Omega_k$ without requiring knowledge of the other set constraints. The algorithm yields a sequence of solutions $\vec{x}_k^{av}[t]$, $\vec{p}_k^{av}[t]$ with improved error bounds at each timestep t. This is an important feature for distributed applications, and a significant departure from the primal-dual optimization results of [1] which involve maintaining a running "best" solution for the global problem as computations proceed. Evaluating the "best" solution computed so far is not always possible in distributed settings, as it involves complete knowledge of the global problem and usually requires all processors to share all of their state variables and constraint sets with all other processors. In our algorithm, each node passes vectors $\vec{H}_k[t]$ only to neighboring nodes, and this information is sufficient to ensure that local estimates of the public variables get progressively closer and closer to satisfying the global constraints.

We note that the interior point assumption leads to an inequality constraint (13) that converges to (3) like $O(1/t)$. In the case when no interior point exists, the update equation (7) can be modified by the sequence $\vec{\delta}[t]$ as in (8) and (9). However, this would yield a convergence of $O(1/\sqrt{t})$.

IV. Performance Analysis

To prove the theorem, we first present a fundamental result concerning *Lyapunov drift*. Lyapunov drift theory has been useful in developing stable control policies for queueing systems [17] [19] [20] [21] [14], and the theory has recently been extended to allow for performance optimization of stochastic networks [15] [18]. Here, we consider a deterministic variant of the Lyapunov result in [15] applied to the violation sequences of the previous section. Specifically, let $\vec{U}[t]$ represent a vector sequence of non-negative variables indexed by time $t \in \{0, 1, 2, \ldots\}$. Define the *Lyapunov function* $L(\vec{U}[t]) = ||\vec{U}[t]||^2$. Here we use the Euclidean norm, so that $||\vec{U}||^2$ represents the sum of squares of the individual entries of vector $\vec{U}$. Define the *single-step Lyapunov drift* as follows:

$$\Delta(\vec{U}[t]) = L(\vec{U}[t+1]) - L(\vec{U}[t])$$

At any time t, the $\vec{U}[t]$ vector can be viewed as the "current state" of a dynamic system. Let $\vec{x}[t], \vec{p}[t]$ be vector sequences representing control decision variables that effect the evolution of $\vec{U}[t]$. Let $g(\vec{x}, \vec{p})$ be a non-negative *cost function*, assumed to be convex in the composite vector $(\vec{x}, \vec{p})$. We assume the cost function is bounded and define G_{max} as an upper bound on the maximum cost over all possible vectors $\vec{x}$ and $\vec{p}$. Consider any particular *target vectors* $\vec{x}^*, \vec{p}^*$ that yield a desired cost.

Theorem 2: (Optimization via Lyapunov Drift) If $\vec{U}[0] = \vec{0}$ and if there are positive constants V, ϵ, C such that for all timeslots t and all sequences $\vec{U}[t]$ we have:

$$\Delta(\vec{U}[t]) \quad \leq \quad C - \vec{\epsilon} \cdot \vec{U}[t] + Vg(\vec{x}^*, \vec{p}^*) - Vg(\vec{x}[t], \vec{p}[t])$$

then for all $t \in \{1, 2, \ldots\}$ we have:

(a) $g(\vec{x}^{av}[t], \vec{p}^{av}[t]) \leq g(\vec{x}^*, \vec{p}^*) + C/V$

(b) $||\vec{U}[t]|| \leq \frac{C+VG_{max}}{\epsilon} + \frac{\epsilon}{2}$

where $\vec{x}^{av}[t]$ and $\vec{p}^{av}[t]$ are empirical averages of the control variables, defined as in (11).

Proof: To prove part (a), note that the drift condition of the theorem together with non-negativity of the $\vec{U}[t]$ values imply that for all t:

$$\Delta(\vec{U}[t]) + Vg(\vec{x}[t], \vec{p}[t]) \leq C + Vg(\vec{x}^*, \vec{p}^*)$$

Summing over all times $\tau \in \{0, \ldots, t-1\}$ and dividing by t, we have:

$$\frac{L(\vec{U}[t])}{t} + V\frac{1}{t}\sum_{\tau=0}^{t-1} g(\vec{x}[\tau], \vec{p}[\tau]) \leq C + Vg(\vec{x}^*, \vec{p}^*) \quad (15)$$

Because $g(\vec{x}, \vec{p})$ is convex in $(\vec{x}, \vec{p})$, by Jensen's inequality we have $\frac{1}{t}\sum_{\tau=0}^{t-1} g(\vec{x}[\tau], \vec{p}[\tau]) \geq g(\vec{x}^{av}[t], \vec{p}^{av}[t])$. Using this bound together with non-negativity of the Lyapunov function in the inequality (15) yields part (a) of the theorem.

To prove (b), note that the drift condition implies:

$$\Delta(\vec{U}[t]) \leq C + VG_{max} - \vec{\epsilon} \cdot \vec{U}[t]$$

Because the $\vec{U}[t]$ vectors have non-negative entries, it follows that the maximum increment in $L(\vec{U}[t])$ is $C + VG_{max}$. Further, $L(\vec{U}[t])$ cannot increase if $\vec{\epsilon} \cdot \vec{U}_k[t] \geq C + VG_{max}$. Therefore, we have for all times t:

$$L(\vec{U}[t]) \leq L^* + C + VG_{max} \quad (16)$$

where the value of L^* is the maximum value of $||\vec{U}||^2$ subject to $\vec{\epsilon} \cdot \vec{U} \leq C + VG_{max}$. The maximum is achieved when all weight is placed on a single entry of $\vec{U}$, and hence $L^* = (C + VG_{max})^2/\epsilon^2$. It follows from (16) that $||\vec{U}[t]|| \leq \sqrt{\frac{(C+VG_{max})^2}{\epsilon^2}} + C + VG_{max}$. It is not difficult to show that $\sqrt{\frac{A^2}{\epsilon^2} + A} \leq \frac{A}{\epsilon} + \frac{\epsilon}{2}$ for all positive values A, ϵ, and the result of part (b) follows. □

A. Proof of Theorem 1

We now compute the Lyapunov drift associated with the iterative algorithm of the previous section. First, let $\vec{U}[t], \vec{Y}[t]$, and $\vec{Z}[t]$ represent composite vectors consisting of concatenated $\vec{U}_k[t], \vec{Y}_k[t]$, and $\vec{Z}_k[t]$ vectors for $k \in \{1, \ldots, K\}$. We define the Lyapunov function $L(\vec{U}, \vec{Y}, \vec{Z}) \triangleq \sum_k ||\vec{U}_k||^2 + \sum_k ||\vec{Y}_k||^2 + \sum_k ||\vec{Z}_k||^2$. The one-step Lyapunov drift is thus:

$$\Delta(\vec{U}[t], \vec{Y}[t], \vec{Z}[t]) \triangleq L(\vec{U}[t+1], \vec{Y}[t+1], \vec{Z}[t+1]) - L(\vec{U}[t], \vec{Y}[t], \vec{Z}[t])) \quad (17)$$

Consider the update equations (7)-(9) for the violation sequences. Taking the squared norm of both sides of (7) yields:

$$||\vec{U}_k[t+1]||^2 \leq ||\vec{U}_k[t]||^2 + ||\vec{b}_k||^2 + F_{max} - 2\vec{U}_k[t] \cdot (\vec{b}_k - \vec{f}_k(\vec{x}_k[t], \vec{p}_k[t]))$$

Likewise, taking the squared norm of both sides of (8) yields:

$$||\vec{Y}_k[t+1]||^2 \leq ||\vec{Y}_k[t]||^2 + ||\vec{x}_k[t] + \vec{\delta}[t]||^2 + ||\vec{x}_{Par(k)}[t]||^2 - 2\sum_k \vec{Y}_k[t] \cdot (\vec{x}_k[t] + \vec{\delta}[t] - \vec{x}_{Par(k)}[t])$$

Similarly, squaring (9) yields:

$$||\vec{Z}_k[t+1]||^2 \leq ||\vec{Z}_k[t]||^2 + ||\vec{x}_{Par(k)}[t] + \vec{\delta}[t]||^2 + ||\vec{x}_k[t]||^2 - 2\sum_k \vec{Z}_k[t] \cdot (\vec{x}_{Par(k)}[t] + \vec{\delta}[t] - \vec{x}_k[t])$$

Using the abbreviated notation Δ to represent the Lyapunov drift defined in (17), it follows that the drift satisfies:

$$\begin{aligned}\Delta \leq C &- 2\sum_k \vec{U}_k[t] \cdot (\vec{b}_k - \vec{f}_k(\vec{x}_k[t], \vec{p}_k[t])) \\ &-2\sum_k \vec{Y}_k[t] \cdot \left(\vec{x}_k[t] + \vec{\delta}[t] - \vec{x}_{Par(k)}[t]\right) \\ &-2\sum_k \vec{Z}_k[t] \cdot \left(\vec{x}_{Par(k)}[t] + \vec{\delta}[t] - \vec{x}_k[t]\right) \\ &+V\sum_k g_k(\vec{x}_k[t], \vec{p}_k[t]) - V\sum_k g_k(\vec{x}_k[t], \vec{p}_k[t]) \quad (18)\end{aligned}$$

where C is defined in Theorem 1, and where we have added and subtracted the optimization metric. By shifting the sums, using the definition $\vec{H}_k[t] \triangleq \vec{Y}_k[t] - \vec{Z}_k[t]$, and recalling that $Child(k)$ is the set of all nodes j such that $Par(j) = k$, we have:

$$\begin{aligned}\Delta \leq C &- 2\sum_k \vec{U}_k[t] \cdot (\vec{b}_k - \vec{f}_k(\vec{x}_k[t], \vec{p}_k[t])) \\ &-2\sum_k \vec{x}_k[t] \cdot (\vec{H}_k[t] - \textstyle\sum_{j \in Child(k)} \vec{H}_j[t]) \\ &-2\sum_k \vec{\delta}[t] \cdot (\vec{Y}_k[t] + \vec{Z}_k[t]) \\ &+V\sum_k g_k(\vec{x}_k[t], \vec{p}_k[t]) - V\sum_k g_k(\vec{x}_k[t], \vec{p}_k[t]) \quad (19)\end{aligned}$$

The right hand sides of (18) and (19) are identical. However, from the latter expression it is clear that the iterative algorithm defined in the previous section is precisely designed to minimize the sum of the second, third, fourth, and fifth terms on the right hand side of the above inequality (19) over all feasible control vectors $\vec{x}_k, \vec{p}_k$. Hence, the drift is less than or equal to the resulting right hand side if a particular set of feasible control vectors are plugged into the second, third, fourth, and fifth terms. Now recall from the *interior point assumption* of Section II-A that feasible vectors $\vec{x}^{(\epsilon)}, \vec{p}_k^{(\epsilon)}$ exist and satisfy $\vec{f}_k(\vec{x}^{(\epsilon)}, \vec{p}_k^{(\epsilon)}) \leq \vec{b}_k - \vec{\epsilon}$ for all k and for all ϵ such that $0 < \epsilon \leq \epsilon_{max}$. Hence, using the right hand side as expressed in (18), we have:

$$\begin{aligned}\Delta \leq C &- 2\sum_k \vec{U}_k[t] \cdot \vec{\epsilon} - 2\sum_k \vec{\delta}[t] \cdot (\vec{Y}_k[t] + \vec{Z}_k[t]) \\ &+V\sum_k g_k(\vec{x}^{(\epsilon)}, \vec{p}_k^{(\epsilon)}) - V\sum_k g_k(\vec{x}_k[t], \vec{p}_k[t])\end{aligned}$$

The above drift expression is in the form specified by the Lyapunov drift theorem (Theorem 2). Hence, we have the following for all times t:

$$\sum_k g_k(\vec{x}_k^{av}[t], \vec{p}_k^{av}[t]) \leq \sum_k g_k(\vec{x}^{(\epsilon)}, \vec{p}_k^{(\epsilon)}) + C/V \quad (20)$$

$$||\vec{U}[t]|| \leq \frac{C+VKG_{max}}{2\epsilon} + \epsilon \quad (21)$$

$$||\vec{Y}[t] + \vec{Z}[t]|| \leq \frac{C+VKG_{max}}{2\delta[t]} + \delta[t] \quad (22)$$

where we have used the fact that $\delta[t]$ decreases with t, and where the norm of the composite vectors $\vec{U}[t]$ and $\vec{Y}[t] + \vec{Z}[t]$ is given by:

$$||\vec{U}[t]|| = \sqrt{\sum_{k=1}^{K} ||\vec{U}_k[t]||^2}$$

$$||\vec{Y}[t] + \vec{Z}[t]|| = \sqrt{\sum_{k=2}^{K} ||\vec{Y}_k[t] + \vec{Z}_k[t]||^2}$$

The inequalities (20)-(22) hold for any value ϵ such that $0 < \epsilon \leq \epsilon_{max}$. Hence, they can be optimized separately by choosing the best ϵ value. Recall that $\vec{x}^{(\epsilon)}$ and $\vec{p}_k^{(\epsilon)}$ converge to the optimal operating point as $\epsilon \to 0$. Hence, taking a limit in (20) as $\epsilon \to 0$ proves the performance bound (12). Conversely, the bound in (21) is minimized by setting $\epsilon = \epsilon_{max}$, proving that:

$$||\vec{U}[t]|| \leq \frac{C + VKG_{max}}{2\epsilon_{max}} + \epsilon_{max} \qquad (23)$$

Recall that the $\vec{U}_k[t]$ values represent queue backlogs (see (7)). It follows that the accumulated "arrivals" to the queue during the first t slots are less than or equal to the maximum possible "departures" plus the backlog at time $t-1$:

$$\sum_{\tau=0}^{t-1} \vec{f}_k(\vec{x}_k[\tau], \vec{p}_k[\tau]) \leq t\vec{b}_k + \vec{U}_k[t-1]$$

Dividing the above inequality by t and using Jensen's inequality to push the time average summation inside the convex function $\vec{f}_k(\cdot)$ yields:

$$\vec{f}_k(\vec{x}_k^{av}[t], \vec{p}_k^{av}[t]) \leq \vec{b}_k + \frac{\vec{U}_k[t-1]}{t} \qquad (24)$$

Combining (24) and (23) proves the performance bound (13). Similarly, the performance bound (14) can be proven directly from (22), using the fact that $\frac{1}{t}\sum_{\tau=0}^{t-1}\delta[\tau] \leq 2/\sqrt{t} - 1/t$ for $t \geq 1$, as well as the fact that the accumulated "arrivals" to the $\vec{Y}_k[t]$ queues over the first t slots are bounded by the total service opportunities over this interval plus the backlog at time $t-1$. This proves Theorem 1.

V. Conclusions

We have developed a framework for distributed and secure computation of convex programs using an arbitrary connected graph of processors. Each node of the graph participates in the optimization by solving a simplified problem involving both public and private variables, and the resulting algorithm maintains privacy while achieving global optimality. Our analysis uses a Lyapunov drift technique that transforms the optimization into a corresponding queue control strategy. The technique is quite general and can be extended to treat stochastic versions of this problem.

References

[1] D. P. Bertsekas, A. Nedic, and A. E. Ozdaglar. *Convex Analysis and Optimization*. Boston: Athena Scientific, 2003.

[2] Y. Bartal, J. W. Byers, and D. Raz. Fast, distributed approximation algorithms for positive linear programming with applications to flow control. *Siam Journal of Computing*, vol. 33, no. 6, pp. 1261-1279, 2004.

[3] D. P. Bertsekas and P. Tseng. Partial proximal minimization algorithms for convex programming. *Massachusetts Institute of Technology Technical Report*, 1995.

[4] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Prentice-Hall, Englewood Cliffs, NJ, 1989.

[5] M. C. Ferris and O. L. Mangasarian. Parallel constraint distribution. *SIAM Journal on Optimization*, vol. 1, pp. 487-500, 1991.

[6] D. Kempe, A. Dobra, and J. Gehrke. Gossip-based computation of aggregate information. *Proceedings of FOCS*, 2003.

[7] J. K. Bordim, K. Nakano, and H. Shen. Sorting on single-channel wireless sensor networks. *International Symposium on Parallel Architectures, Algorithms, and Networks*, May 2002.

[8] D. Kempe and F. McSherry. A decentralized algorithm for spectral analysis. *Proc. of STOC*, 2004.

[9] M. Singh, V. K. Prasanna, and J. D. P. Rolim. Collaborative and distributed computation in mesh-like wireless sensor arrays. *Personal Wireless Communications*, Sept. 2003.

[10] R. Miller and Q. F. Stout. Parallel algorithms for regular architectures: Meshes and pyramids. *MIT Press*, 1996.

[11] D. Nassimi and S. Sahni. Bitonic sort on mesh-connected computers. *IEEE Trans. on Computers*, vol. c-27, Jan. 1979.

[12] J. N. Tsitsiklis, D. P. Bertsekas, and M. Athens. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE Transactions on Automatic Control*, Vol. AC-31, no. 9, September 1986.

[13] M. J. Neely. *Dynamic Power Allocation and Routing for Satellite and Wireless Networks with Time Varying Channels*. PhD thesis, Massachusetts Institute of Technology, LIDS, 2003.

[14] M. J. Neely, E. Modiano, and C. E Rohrs. Dynamic power allocation and routing for time varying wireless networks. *IEEE Journal on Selected Areas in Communications*, January 2005.

[15] M. J. Neely, E. Modiano, and C. Li. Fairness and optimal stochastic control for heterogeneous networks. *Proceedings of IEEE INFOCOM*, March 2005.

[16] D. P. Bertsekas and R. Gallager. *Data Networks*. New Jersey: Prentice-Hall, Inc., 1992.

[17] L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Transacations on Automatic Control*, Vol. 37, no. 12, Dec. 1992.

[18] M. J. Neely. Energy optimal control for time varying wireless networks. *Proceedings of IEEE INFOCOM*, March 2005.

[19] L. Tassiulas and A. Ephremides. Dynamic server allocation to parallel queues with randomly varying connectivity. *IEEE Trans. on Inform. Theory*, vol. 39, pp. 466-478, March 1993.

[20] N. McKeown, V. Anantharam, and J. Walrand. Achieving 100% throughput in an input-queued switch. *Proc. INFOCOM*, 1996.

[21] E. Leonardi, M. Melia, F. Neri, and M. Ajmone Marson. Bounds on average delays and queue size averages and variances in input-queued cell-based switches. *Proc. INFOCOM*, 2001.