# Cloud-Based Centralized/Decentralized Multi-Agent Optimization with Communication Delays

Matthew T. Hale,[†] Angelia Nedić,[⋆] and Magnus Egerstedt[†]

*Abstract*— **We present and analyze a hybrid computational architecture for performing multi-agent optimization. The optimization problems under consideration have convex objective and constraint functions with mild smoothness conditions imposed on them. For such problems, we provide a primal-dual algorithm implemented in the hybrid architecture, which consists of a decentralized network of agents into which an updated dual vector is occasionally injected, and we establish its convergence properties. In this setting, a central cloud computer is responsible for aggregating information, computing dual variable updates, and distributing these updates to the agents. The agents update their (primal) state variables and also communicate among themselves with each agent sharing and receiving state information with some number of its neighbors. Throughout, communications with the cloud are not assumed to be synchronous or instantaneous, and communication delays are explicitly accounted for in the modeling and analysis of the system. Experimental results for a team of robots are presented to support the theoretical developments made.**

## I. Introduction

Algorithms for multi-agent and distributed optimization have been considered for a variety of problems in part because of the varied collection of application domains in which such problems arise. Applications of multi-agent optimization can be found in robotics [4], [2], [18], power systems [14], sensor networks [19], [8], [17], and communications [1], [7], [13]. These diverse applications lead to optimization problems of many different formulations. Correspondingly, algorithms have been developed that allow for a broad range of problem characteristics. For example, in [15] problems with constraints on network connectivity and memory are considered. In [3] a distributed method for minimizing a sum of convex functions over a digraph is devised. Problems with time-varying communication graphs, non-differentiable objective functions, and noisy communication links are considered in [12].

The development of decentralized methods in optimization has been motivated in part by the fact that centralized methods may not scale well for very large networks of agents [20, Section 1.1]. At the same time, centralized methods can more efficiently solve some problems, like the Credit Assignment Problem in multi-agent robotics, than decentralized methods [16, Section 3.1]. Such examples indicate that centralized information may be rich in a way that could be useful in networks which would otherwise be purely decentralized. In adding a centralized component to a decentralized network, it seems likely that the centralized component would operate slower than the decentralized components. Nevertheless, one may ask whether it would be useful to occasionally inject global information into a multi-agent network where such information would otherwise be absent.

Towards answering this question, we present here a multi-agent optimization architecture in which a cloud computer is used to occasionally provide centralized information to a network of agents solving a nonlinear programming problem. This cloud-based optimization architecture was introduced in [5], though here we substantially broaden the class of problems to be solved and allow for communications delays when communicating with the cloud. The cloud carries out computations based on information sent to it by agents in the network and intermittently disseminates these results to the agents for use in their own computations. At the same time, each agent shares its state with some number of neighboring agents at each timestep. In [5], the assumption was made that all information in the network was synchronized at each time, so that all computations were relying on the same information. Here we eliminate this assumption and, as a consequence, delays occur which give rise to various kinds of errors. We present explicit bounds on these errors in terms of known constants.

To solve nonlinear programs in a distributed manner across many agents, we cast such problems as variational inequalities and then use a Tikhonov regularization, which endows the resulting variational inequality with certain properties that let us draw from existing results. In particular we consider a fixed regularization as was done in [10]. A fixed regularization is desirable for multi-agent problems because it may be difficult to synchronize the timing of the changes in regularization parameters across large networks. Accordingly, we use the approach of [10] as a starting point, though the problem and approach there are quite different from the current paper. In [10], the need for these results stems from reducing computation times in Lagrangian subproblems associated with a dual optimization scheme by allowing inexactness in some computations. Here we use a different architecture and different model for delays to operate as fast as possible by using the most recent information available to the agents. Doing so results in delays in communication, and computations that rely on information of different ages. The resulting structure of delays will be detailed below.

[†]School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA. Email: {matthale, magnus}@gatech.edu. Research supported in part by the NSF under Grant CNS-1239225.

[⋆]Department of Industrial and Enterprise Systems Engineering, University of Illinois, Urbana IL, 61801, USA. Email: angelia@illinois.edu.

The rest of the paper is organized as follows. Section II will cover the background concerning the problem to be solved and a centralized method for solving it. Then, Section III will cover the cloud architecture and modify the centralized solution method to fit with the hybrid centralized/decentralized system. Section IV will present the convergence results and error bounds of the partially decentralized algorithm, and Section V will present experimental results from an implementation of this algorithm on a team of mobile robots. Section VI concludes the paper.

## II. Problem Formulation and Centralized Solution

In this section we formulate the problem to be solved. This section states global results that will be modified later in Section III to fit with a hybrid architecture described therein.

### A. Variational Inequality Setup

Consider a multi-agent optimization problem comprised of $N$ agents indexed by $i \in I := \{1, \dots, N\}$. Suppose that agent $i$ has state $x_i \in \mathbb{R}^{n_i}$ with $n_i \in \mathbb{N}$, and let the vector $x$ denote the column vector of all states, namely

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \in \mathbb{R}^n,$$

where $n = \sum_{i=1}^{N} n_i$. Let each agent have a local objective function depending only on its own state, $f_i$. We assume that $f_i : \mathbb{R}^{n_i} \to \mathbb{R}$ is $C^1$ and convex. We also consider a global cost that is not necessarily separable, $c(x)$, and assume that $c : \mathbb{R}^n \to \mathbb{R}$ is both $C^1$ and convex as well. We assume that agent $i$ knows $c$ and $f_i$, but not $f_\ell$ for any $\ell \neq i$; that is, each agent knows the non-separable cost and its own local cost, but not the local cost function of any other agent. We assume further that the cloud does not know $f_i$ for any $i \in I$.

The agents are collectively subject to global inequality constraints of the form

$$g(x) = \begin{pmatrix} g_1(x) \\ g_2(x) \\ \vdots \\ g_m(x) \end{pmatrix} \leq 0,$$

where $g : \mathbb{R}^n \to \mathbb{R}^m$ with $m \geq 1$. The constraint functions $g_j : \mathbb{R}^n \to \mathbb{R}$ are assumed to be convex and $C^1$ for all $j \in J := \{1, \dots, m\}$. Each agent's state is also constrained to lie in a non-empty set $X_i \subset \mathbb{R}^{n_i}$, i.e., we require

$$x_i \in X_i$$

for every $i$, where $X_i$ is compact and convex. Letting

$$X = X_1 \times X_2 \times \cdots \times X_N,$$

we encapsulate each set constraint by requiring

$$x \in X.$$

Regarding the class of optimization problems under consideration, we summarize the conditions that we have imposed in the following assumption.

*Assumption 1:* The set $X$ is non-empty, compact, and convex. The functions $\{g_j\}_{j \in J}$ and $c$ are convex and $C^1$ in $x$, and $f_i$ is convex and $C^1$ in $x_i$ for all $i \in I$. ▲

For notational convenience, define the function

$$f(x) = \sum_{i=1}^{N} f_i(x_i) + c(x).$$

Let $\nabla_{x_i}$ denote the operator $\frac{\partial}{\partial x_i}$ and define the map

$$\nabla f(x) = \Big(\nabla_{x_1}\big(f_1(x_1)+c(x)\big), \dots, \nabla_{x_N}\big(f_N(x_N)+c(x)\big)\Big).$$

We enforce the following assumption on $\nabla f$.

*Assumption 2:* The map $\nabla f$ is Lipschitz continuous with Lipschitz constant $L_f$. ▲

Note that any collection of functions $f_i$ and $c$ which are all $C^2$ comprise an $f$ that automatically satisfies Assumption 2 whenever $X$ is compact (cf. Assumption 1). Concerning the constraints $g$, we have the following assumptions.

*Assumption 3:* (Slater's Condition) There exists a vector $\bar{x} \in X$ such that $g(\bar{x}) < 0$, i.e., the constraints are strictly feasible at $\bar{x}$. ▲

*Assumption 4:* The gradient of each constraint, $\nabla g_j$, $j \in J$, is Lipschitz with constant $L_j$ and hence $\nabla g$ is Lipschitz with constant

$$L_g = \sqrt{\sum_{j=1}^{m} L_j^2}.$$

▲

A global formulation of the multi-agent optimization problem under consideration is given by:

*Problem 1:*

$$\begin{aligned} &\text{minimize } f(x) \\ &\text{subject to } g(x) \leq 0 \\ &\qquad\qquad\quad x \in X. \end{aligned}$$

◊

Assumption 1 guarantees that Problem 1 has a solution and Assumption 3 guarantees that a dual solution exists with no duality gap. Denote an optimal primal-dual pair for Problem 1 by $(\hat{x}, \hat{\mu})$. We now define the Lagrangian associated with Problem 1 as

$$L(x, \mu) = f(x) + \mu^T g(x),$$

where $x \in X$ is as defined above and $\mu$ is a vector of Kuhn-Tucker multipliers in the non-negative orthant of $\mathbb{R}^m$, denoted $\mathbb{R}^m_+$. By definition, $L(\cdot, \mu)$ is convex for all $\mu \in \mathbb{R}^m_+$ and $L(x, \cdot)$ is concave for every $x$. Seminal work by Kuhn and Tucker [11] showed that optimal primal-dual pairs for Problem 1 are saddle points of $L$ which maximize $L(\hat{x}, \cdot)$ and minimize $L(\cdot, \hat{\mu})$. This condition can be expressed concisely as: for all $x \in X$ and $\mu \in \mathbb{R}^m_+$,

$$L(\hat{x}, \mu) \leq L(\hat{x}, \hat{\mu}) \leq L(x, \hat{\mu}).$$

The problem of finding Lagrangian saddle points can be restated as a variational inequality with an identical solution set (e.g., [9, Section 11.1]). Let $\nabla_x$ and $\nabla_\mu$ denote the operators $\frac{\partial}{\partial x}$ and $\frac{\partial}{\partial \mu}$, respectively. In the variational inequality setting, we wish to find a point $(\hat{x}, \hat{\mu}) \in X \times \mathbb{R}^m_+$ such that

$$\left[\begin{pmatrix} x \\ \mu \end{pmatrix} - \begin{pmatrix} \hat{x} \\ \hat{\mu} \end{pmatrix}\right]^T \begin{bmatrix} \nabla_x L(\hat{x}, \hat{\mu}) \\ -\nabla_\mu L(\hat{x}, \hat{\mu}) \end{bmatrix} \geq 0$$

for all $(x, \mu) \in X \times \mathbb{R}^m_+$. In order to make use of certain established results concerning variational inequalities, we take two further theoretical steps: first we regularize the map $(\nabla_x L \ \ -\nabla_\mu L)^T$ to make it strongly monotone, and second we find a (non-empty) compact, convex set containing the optimal primal-dual vectors. Errors introduced by the regularization of $(\nabla_x L \ \ -\nabla_\mu L)^T$ are discussed in Section IV.

## B. Tikhonov Regularization

By the convex-concave property of $L$, the gradient map

$$\begin{pmatrix} \nabla_x L(x, \mu) \\ -\nabla_\mu L(x, \mu) \end{pmatrix}$$

is monotone, and we use a fixed Tikhonov regularization in order to work with a strongly monotone map. This is done by regularizing the Lagrangian function as follows:

$$L_{\nu,\epsilon}(x, \mu) = f(x) + \frac{\nu}{2}\|x\|^2 + \mu^T g(x) - \frac{\epsilon}{2}\|\mu\|^2,$$

where $\nu > 0$ and $\epsilon > 0$, and these values are kept fixed to avoid the need to synchronize changes in parameter values across many agents.

Under this regularization, we see that $L_{\nu,\epsilon}(\cdot, \mu)$ is strongly convex for all $\mu \in \mathbb{R}^m_+$ and $L_{\nu,\epsilon}(x, \cdot)$ is strongly concave for all $x$. These properties imply that the map $(\nabla_x L_{\nu,\epsilon} \ \ -\nabla_\mu L_{\nu,\epsilon})^T$ is strongly monotone. In addition, the strongly convex-strongly concave property of $L_{\nu,\epsilon}$, together with Assumption 1 and Assumption 3, guarantee the existence of a unique saddle point, $(\hat{x}_{\nu,\epsilon}, \hat{\mu}_{\nu,\epsilon}) \in X \times \mathbb{R}^m_+$. Using the regularized Lagrangian, we now state the variational inequality of interest.

*Problem 2:* Find the point $(\hat{x}_{\nu,\epsilon}, \hat{\mu}_{\nu,\epsilon}) \in X \times \mathbb{R}^m_+$ such that for all $(x, \mu) \in X \times \mathbb{R}^m_+$,

$$\left[\begin{pmatrix} x \\ \mu \end{pmatrix} - \begin{pmatrix} \hat{x}_{\nu,\epsilon} \\ \hat{\mu}_{\nu,\epsilon} \end{pmatrix}\right]^T \begin{bmatrix} \nabla_x L_{\nu,\epsilon}(\hat{x}_{\nu,\epsilon}, \hat{\mu}_{\nu,\epsilon}) \\ -\nabla_\mu L_{\nu,\epsilon}(\hat{x}_{\nu,\epsilon}, \hat{\mu}_{\nu,\epsilon}) \end{bmatrix} \geq 0.$$

$\diamondsuit$

We note that a solution to the above variational inequality is also a saddle-point of the regularized Lagrangian function $L_{\nu,\epsilon}$, i.e., $(\hat{x}_{\nu,\epsilon}, \hat{\mu}_{\nu,\epsilon}) \in X \times \mathbb{R}^m_+$ solves the above variational inequality problem if and only if for all $(x, \mu) \in X \times \mathbb{R}^m_+$,

$$L_{\nu,\epsilon}(\hat{x}_{\nu,\epsilon}, \mu) \leq L_{\nu,\epsilon}(\hat{x}_{\nu,\epsilon}, \hat{\mu}_{\nu,\epsilon}) \leq L_{\nu,\epsilon}(x, \hat{\mu}_{\nu,\epsilon}). \quad (1)$$

## C. Bounds on Dual Variables

We proceed along the lines of [21] and derive a bound on $\hat{\mu}_{\nu,\epsilon}$. Letting $\bar{x}$ denote a Slater point for the constraints $g$, we define the dual function associated with $L_{\nu,\epsilon}$ as

$$q_{\nu,\epsilon}(\mu) := \min_{x \in X} L_{\nu,\epsilon}(x, \mu).$$

Now consider an arbitrary multiplier $\tilde{\mu} \in \mathbb{R}^m_+$ and let $\tilde{x} \in X$ be such that $\tilde{x} = \min_{x \in X} L_{\nu,\epsilon}(x, \tilde{\mu})$. By the definition of $q_{\nu,\epsilon}$ we then have

$$q_{\nu,\epsilon}(\tilde{\mu}) = L_{\nu,\epsilon}(\tilde{x}, \tilde{\mu}) \leq L_{\nu,\epsilon}(\hat{x}_{\nu,\epsilon}, \tilde{\mu}) \leq L_{\nu,\epsilon}(\hat{x}_{\nu,\epsilon}, \hat{\mu}_{\nu,\epsilon}) \leq L_{\nu,\epsilon}(\bar{x}, \hat{\mu}_{\nu,\epsilon}),$$

where the two right-most inequalities follow from the saddle-point property of $(\hat{x}_{\nu,\epsilon}, \hat{\mu}_{\nu,\epsilon})$ in Equation (1). Expanding the regularized Lagrangian expression, we have

$$\begin{aligned} q_{\nu,\epsilon}(\tilde{\mu}) &\leq f(\bar{x}) + \hat{\mu}_{\nu,\epsilon}^T g(\bar{x}) + \frac{\nu}{2}\|\bar{x}\|^2 - \frac{\epsilon}{2}\|\hat{\mu}_{\nu,\epsilon}\|^2 \\ &\leq f(\bar{x}) + \hat{\mu}_{\nu,\epsilon}^T g(\bar{x}) + \frac{\nu}{2}\|\bar{x}\|^2. \end{aligned}$$

Rearranging terms then gives

$$\sum_{j=1}^m \hat{\mu}_{\nu,\epsilon,j} \leq \frac{f(\bar{x}) + \frac{\nu}{2}\|\bar{x}\|^2 - q_{\nu,\epsilon}(\tilde{\mu})}{\min\limits_{1 \leq j \leq m}\{-g_j(\bar{x})\}}. \quad (2)$$

For any $\nu > 0$ and $\epsilon > 0$, we certainly have

$$q_{\nu,\epsilon}(\tilde{\mu}) = L_{\nu,\epsilon}(\tilde{x}, \tilde{\mu}) \geq L_{0,\epsilon}(\tilde{x}, \tilde{\mu}) = f(\tilde{x}) + \tilde{\mu}^T g(\tilde{x}) - \frac{\epsilon}{2}\|\tilde{\mu}\|^2.$$

Selecting $\tilde{\mu} = 0$, observe that

$$L_{0,\epsilon}(\tilde{x}, 0) = f(\tilde{x}).$$

Letting $f^*_X = \min_{x \in X} f(x)$, the bound in Equation (2) can be simplified to

$$\sum_{j=1}^m \hat{\mu}_{\nu,\epsilon,j} \leq \frac{f(\bar{x}) + \frac{\nu}{2}\|\bar{x}\|^2 - f^*_X}{\min\limits_{1 \leq j \leq m}\{-g_j(\bar{x})\}}.$$

Using that $\hat{\mu}_{\nu,\epsilon,j} \geq 0$ for all $j \in J$ and defining

$$\mathcal{D}_\nu = \left\{\mu \in \mathbb{R}^m_+ : \|\mu\|_1 \leq \frac{f(\bar{x}) + \frac{\nu}{2}\|\bar{x}\|^2 - f^*_X}{\min\limits_{1 \leq j \leq m}\{-g_j(\bar{x})\}}\right\},$$

we see that $\mathcal{D}_\nu$ is non-empty, compact, and convex, and we are guaranteed that $\hat{\mu}_{\nu,\epsilon} \in \mathcal{D}_\nu$. Using $\mathcal{D}_\nu$, we can now define the algorithm used to solve Problem 2.

*Algorithm 1:* Given an initial point $(x(0), \mu(0)) \in X \times \mathcal{D}_\nu$, execute the update law

$$x(k+1) = \Pi_X\Big[x(k) - \alpha \nabla_x L_{\nu,\epsilon}(x(k), \mu(k))\Big] \quad (3)$$

$$\mu(k+1) = \Pi_{\mathcal{D}_\nu}\Big[\mu(k) + \tau \nabla_\mu L_{\nu,\epsilon}(x(k), \mu(k))\Big], \quad (4)$$

until some stopping criterion is reached. $\blacklozenge$

Here $\Pi_X[\cdot]$ and $\Pi_{\mathcal{D}_\nu}[\cdot]$ are the projections onto the sets $X$ and $\mathcal{D}_\nu$, respectively, with respect to the standard Euclidean norm. In the next section we will explicitly reformulate Algorithm 1 for the cloud-based multi-agent case.

## III. Cloud Architecture and Hybrid Solution

We now cover the architecture that will be used to implement a modified form of Algorithm 1.

### A. Overview

Let agent $i$ have neighborhood set $N_i$ containing the indices of all agents it is *directly* coupled to by $g$ and $c$. That is, if the computation of $\frac{\partial L_{\nu,\epsilon}}{\partial x_i}$ requires $x_j$, then $j \in N_i$ and agent $j$ sends its state to agent $i$ at each time. This structure of communications necessitates that $i \in N_j$ if and only if $j \in N_i$.

Let the agents share their states with each of their neighbors at each timestep. In this framework, each agent stores and manipulates a local copy of Problem 2 onboard and updates its own state within that local copy based on computations it performs onboard. Within each timestep, agent $i$ computes an updated value of its own state, $x_i$, and then shares the new value of $x_i$ with agent $j$ for all $j \in N_i$. In many cases, including in very large networks of agents, we expect that the neighborhood set of each agent will be a small subset of total collection of agents so that $|N_i| < N$ for all $i$, where $| \cdot |$ denotes cardinality. Computing values of dual variables using Algorithm 1 will require all states in the system (see Equation (4)), and given that $|N_i| < N$, we see that no agent will be able to perform these computations. Furthermore, there is no assumption that the communication graph of the system is connected, nor is it even assumed that each agent is coupled to any other agent. In such cases, there is not any way to aggregate all states in the network onboard a single agent, even after long periods of time.

To fill this gap, we use a cloud computer as was done in [5]. The cloud computer is assumed to be capable of executing computationally intensive calculations quickly as would be the case with a computer cluster or server farm. Occasionally every agent sends its state to the cloud and after some time each agent receives back an updated dual vector that it stores onboard and incorporates into its own local calculations of state updates. Because the cloud must take the time to aggregate all states in the network, it is assumed that there are delays in communicating with the cloud and, due to these delays, Algorithm 1 will be modified.

The precise update law used by each agent will be detailed below, though for the current discussion it is sufficient to note that each agent executes some onboard update law using its own state information, information it receives from its neighbors, and the most recent dual vector it has received from the cloud, regardless of how long ago that dual vector was received. Suppose the agents send their states to the cloud at some timestep $k_0$ and suppose they all have some dual vector $\mu_0$ onboard which was received just prior to sending their states to the cloud (the states sent at time $k_0$ were not computed using $\mu_0$). While the agents are waiting to receive an updated dual vector, $\mu_1$, they continue to communicate with each other as before and continue to use $\mu_0$, which is held constant onboard each agent, in their computations of state updates.

Suppose the agents' states from $k_0$ arrive at the cloud[1] at time $k_0 + p_0$ for some $p_0 \in \mathbb{N}$. With all states received, the cloud computes the next dual update using a rule similar to that in Algorithm 1. Suppose that computing the next dual vector takes some number of timesteps $q_0 \in \mathbb{N}$ so that $\mu_1$ has been computed at time $k_0 + p_0 + q_0$. Then the cloud sends $\mu_1$ to each agent and it takes $r_0 \in \mathbb{N}$ timesteps to reach the agents, arriving at time $k_0 + p_0 + q_0 + r_0$. Before agent $i$ computes a primal update of $x_i$ using $\mu_1$, it again sends its state to the cloud and then uses $\mu_1$ in its subsequent computations. Then this process of the agents sharing states with their neighbors, receiving a delayed dual update, and sending their states to the cloud is repeated. Note that the delays $p_0$, $q_0$, and $r_0$ are not assumed to be constant but instead are associated with $k_0$ and are allowed to vary with each communications cycle, i.e., if the agents again send their states to the cloud at time $k_1$, there is no need for $p_1 = p_0$, $q_1 = q_0$, or $r_1 = r_0$. We denote the number of primal steps taken between receiving $\mu(t)$ and $\mu(t+1)$ by $d(t)$.

### B. Multi-agent Implementation

To compactly express this cycle of communication and computation, we implement a change in notation. Let $\mu$ be indexed by the time variable $t \in \mathbb{N}$. The state of each agent will be indexed both over timesteps at which the agents compute primal updates and also over which dual vector is currently being used in its computations. The time index of the agents' computations will be $k \in \mathbb{N}$ and agent $i$'s state will have a superscript to denote the time index of the dual variable agent $i$ currently has onboard. The results of agent $i$'s $k^{th}$ state update using $\mu(t)$ will be denoted $x_i^t(k)$. Using this notation, we restate Algorithm 1 to explicitly specify the update law for agent $i$ and to account for the delays in dual vectors seen above.

*Algorithm 2:* Let agent $i$ have initial state $x_i^0(0)$, stepsize $\alpha$, and initial dual vector $\mu(0)$. Let the cloud have initial multiplier vector $\mu(0)$ and stepsize $\tau$. Execute for each agent $i$, the following two steps: for all $k = 0, \ldots, d(t)-1,\ t \geq 0,$

$$x_i^t(k+1) = \Pi_{X_i}\Big[x_i^t(k) - \alpha \nabla_{x_i} L_{\nu,\epsilon}\big(x^t(k), \mu(t)\big)\Big] \tag{5}$$

and for all $t \geq 0$,

$$\mu(t+1) = \Pi_{\mathcal{D}_\nu}\Big[\mu(t) + \tau \nabla_\mu L_{\nu,\epsilon}\big(x^{t-1}\big(d(t-1)\big), \mu(t)\big)\Big] \tag{6}$$

until a certain stopping criterion is reached by each agent and the cloud. ♦

In the setting of Algorithm 2, we define $x^{-1}\big(d(-1)\big) = x^0(0)$.

## IV. Convergence Analysis

We now show that Algorithm 2 "nearly" converges to the saddle point of $L_{\nu,\epsilon}$ and bound the quantities $\|x^t(k) - \hat{x}_{\nu,\epsilon}\|$ and $\|\mu(t) - \hat{\mu}_{\nu,\epsilon}\|$, where $x^t(k) := (x_1^t(k), \ldots, x_N^t(k))^T$.

[1]The agents' states can arrive at the cloud at different times in which case the cloud can wait to compute an updated dual vector until it has received all agents' states. Here, we assume all states arrive simultaneously for simplicity, though no generality is lost.

*Theorem 1:* Let Assumptions 1–4 hold. Suppose that each agent uses regularization parameter $\nu > 0$ and the cloud uses regularization parameter $\epsilon > 0$. Let the primal stepsize $\alpha$ satisfy $0 < \alpha < C_f := L_f + \nu + M_\nu L_g$ and let the dual stepsize $\tau$ be bounded according to

$$\tau < \min\left\{ \frac{2\nu}{M_g^2 + 2\epsilon\nu}, \frac{2\epsilon}{1+\epsilon^2} \right\}.$$

Define the constant

$$q_d := (1-\tau\epsilon)^2 + \tau^2,$$

which is in the set $(0,1)$ by the definition of $\tau$. Then for all $t \in \mathbb{N}$ we have

$$\|\mu(t+1) - \hat{\mu}_{\nu,\epsilon}\|^2 \leq q_d^{t+1}\|\mu(0) - \hat{\mu}_{\nu,\epsilon}\|^2 + \sum_{i=1}^{t} q_d^{i-1}\Big(q_d M_g^2 M_x^2 q_p^{d(t-i)} + 2\tau^2 M_g^2 M_x^2 q_p^{d(t-i)/2}\Big), \quad (7)$$

where $M_x = \max_{x,y\in X}\|x-y\|$ and $q_p := 1 - 2\alpha\nu + \alpha^2\nu C_f$.
*Proof:* See [6], Theorem 1. ■

The interpretation of Theorem 1 is that vectors $\mu(t)$ computed by the cloud will eventually become close to $\hat{\mu}_{\nu,\epsilon}$, though the distance between them will never become zero. In fact, after a long time this distance is dominated by the first few terms of the summation in Equation (7) because the term containing $q_d^{t+1}$ goes to zero and because $q_d^{i-1}$ becomes negligible for large $i$. We now bound the distance between primal vectors and their optima.

*Theorem 2:* Let Assumptions 1-4 hold. Then for the sequence of primal vectors $\{x^t\big(d(t)\big)\}_{t\in\mathbb{N}}$ generated by Algorithm 2 we have

$$\|x^t\big(d(t)\big) - \hat{x}_{\nu,\epsilon}\| \leq q_p^{d(t)/2} M_x + \frac{M_g}{\nu}\|\mu(t) - \hat{\mu}_{\nu,\epsilon}\|$$

along with

$$\max\{0, g_j(x^t\big(d(t)\big)\} \leq M_g\left(q_p^{d(t)/2}M_x + \frac{M_g}{\nu}\|\mu(t) - \hat{\mu}_{\nu,\epsilon}\|\right).$$

*Proof*: See [6], Theorem 2. ■

The first half of Theorem 2 says that $x_i^t$ will eventually become close to $\hat{x}_{\nu,\epsilon,i}$, with the degree of closeness determined in part by the distance between $\mu(t)$ and $\hat{\mu}_{\nu,\epsilon}$. The second half makes a similar statement about the extent of any constraint violations, namely that the degree of any constraint violation depends upon the distance from $\mu(t)$ to $\hat{\mu}_{\nu,\epsilon}$. Both statements in Theorem 2 say that the length of delays between dual updates can be beneficial when it is long, though naturally longer delays also require more time for convergence, and thus there is a tradeoff between rate of convergence on the one hand and both proximity to $\hat{x}_{\nu,\epsilon}$ and feasibility of the final primal point on the other.

While the point $(\hat{x}_{\nu,\epsilon}, \hat{\mu}_{\nu,\epsilon})$ is not necessarily a saddle point of the original (unregularized) Lagrangian, it is guaranteed to be sufficiently close to one when $\nu$ and $\epsilon$ are small enough; an extended discussion of this matter is in [10, Section 3.2]. In effect, Algorithm 2 lets the agents approach $\hat{x}_{\nu,\epsilon}$ which itself it not far from an optimal solution to Problem 1 in terms of the optimal function value and small feasibility violation of the functional constraints.

## V. Experimental Results

Algorithm 2 was simulated for and then run with 8 robots. We outline the problem and cover the simulation results, and then present the experimental results. All agents are planar so that $x_i \in \mathbb{R}^2$ for all $i$ and $x \in \mathbb{R}^{16}$. The sum of the per-agent objective functions is

$$\sum_{i=1}^{8} f_i(x_i) = \|x_1\|^2 + \left\|x_2 - \begin{pmatrix}-1\\1\end{pmatrix}\right\|^2 + \left\|x_3 - \begin{pmatrix}0.2\\-0.6\end{pmatrix}\right\|^2 + \left\|x_4 - \begin{pmatrix}-1.4\\1.4\end{pmatrix}\right\|^2 + \left\|x_5 - \begin{pmatrix}-0.1\\0.5\end{pmatrix}\right\|^2 + \left\|x_6 - \begin{pmatrix}-0.7\\0.7\end{pmatrix}\right\|^2 + (x_{7,1} - 0.5)^2 + x_{7,2} - 1.1 + (x_{8,1} + 0.3)^2 + x_{8,2}^4.$$

The non-separable term in the cost is

$$c(x) = \frac{1}{200}\big(\|x_1 - x_4\|^2 + \|x_1 - x_8\|^2 + \|x_4 - x_8\|^2\big).$$

As before the total cost used was $f(x) = \sum_{i=1}^{8} f_i(x_i) + c(x)$. The functional and set constraints were

$$g(x) = \begin{pmatrix} \|x_1 - x_2\|^2 - 0.6 \\ \|x_1 - x_5\|^2 - 1.2 \\ \|x_7 - x_8\|^2 - 1.8 \\ \|x_1 - x_3\|^2 - 0.4 \\ \|x_4 - x_6\|^2 - 0.9 \end{pmatrix} \leq 0$$

$$\text{and } X = \prod_{i=1}^{8}[-1.5, 1.5] \times [-1.0, 1.5].$$

The constants needed to solve this problem were computed (approximately) numerically to be

$$L_f = 26.9982,\ L_g = 4.2426,\ M_\nu = 8.7430,\ M_g = 13.5277, \text{ and } M_x = \sqrt{122}.$$

The regularization parameters were chosen to be $\nu = \epsilon = 0.1$, giving $C_f = 64.191$. The primal and dual stepsizes were chosen to be $0.9$ times their upper bounds in Theorem 1, giving

$$\alpha = 0.02804 \text{ and } \tau = 9.835 \cdot 10^{-4}.$$

All delays had length determined by a random integer drawn from a uniform distribution on the integers between 10 and 100, inclusive. Algorithm 2 was run until the agents had computed $250,000$ state updates, during which time the cloud computed $4,539$ dual updates.

In simulation, the initial total distance between the agents' positions and their regularized optima, $\|x^0(0) - \hat{x}_{\nu,\epsilon}\|$, was $2.6024$ and their final total distance, $\|x^{4539}(250000) - \hat{x}_{\nu,\epsilon}\|$, was $0.0843$. In addition, after only $25,000$ iterations the total distance of the agents to $\hat{x}_{\nu,\epsilon}$ was $0.1392$, indicating that fewer steps can be taken while still achieving an acceptable ending state. In the dual space, the final distance to the regularized optimum was $\|\mu(4,539) - \hat{\mu}_{\nu,\epsilon}\| = 0.0393$, indicating close convergence in the dual space as well.

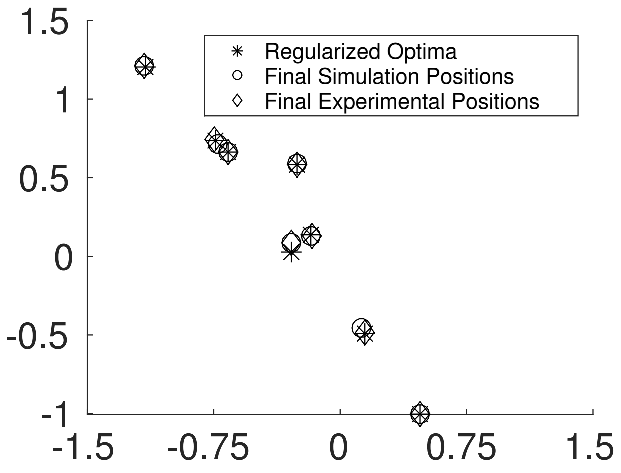

Fig. 1. A plot of $\hat{x}_{\nu,\epsilon}$, the final simulation positions, and the final robot positions, shown as asterisks, circles, and diamonds, respectively.

This problem was executed on the team of 8 Khepera III robots pictured in Figure 2. Position data was gathered using an OptiTrack motion capture system and the cloud-based algorithm was used to generate position waypoints for the agents. The experiment was run until the robots and cloud completed $75,000$ total updates; this point was reached in the middle of a communications cycle, and that cycle was allowed to finish, giving $73,720$ total state updates by each agent and $1,340$ dual updates by the cloud. The final error in the primal space was $\|x^{1340}(73720) - \hat{x}_{\nu,\epsilon}\| = 0.0689$ and the final error in the dual space was $\|\mu(1340) - \hat{\mu}_{\nu,\epsilon}\| = 0.0587$, indicating close convergence of the robots to $(\hat{x}_{\nu,\epsilon}, \hat{\mu}_{\nu,\epsilon})$ and close agreement with the simulation. A plot of the regularized optima, simulation results, and experimental results is shown in Figure 1, representing close agreement among the three sets of data plotted there.

## VI. Conclusion

We presented a hybrid centralized/decentralized algorithm for solving multi-agent nonlinear programs with inequality constraints. To do this, we used a Tikhonov regularization of the problem and a computing regime that spread computations across the agents and a cloud computer. The architectural model incorporated communications delays in the system and approximate convergence of the algorithm was proven. Experimental results were provided to show the applicability of these results.

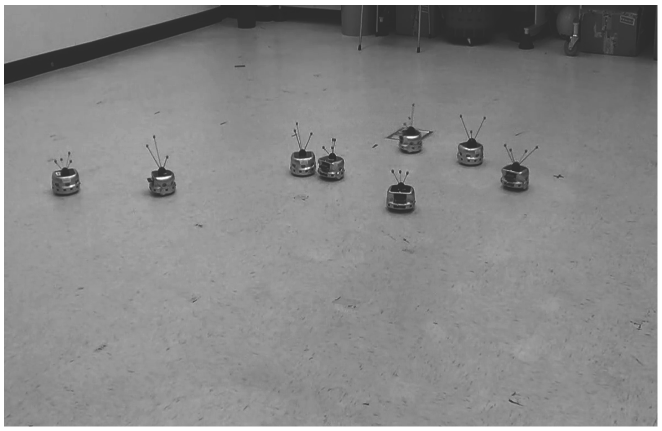

Fig. 2. Team of 8 Khepera III robots executing the cloud-based algorithm.

## References

[1] M. Chiang, S.H. Low, A.R. Calderbank, and J.C. Doyle. Layering as optimization decomposition: A mathematical theory of network architectures. *Proceedings of the IEEE*, 95(1):255–312, Jan 2007.

[2] J. Cortés, S. Martnez, and F. Bullo. Spatially-distributed coverage optimization and control with limited-range interactions. *ESAIM: Control, Optimisation and Calculus of Variations*, 11:691–719, 2005.

[3] B. Gharesifard and J. Cortes. Distributed continuous-time convex optimization on weight-balanced digraphs. *Automatic Control, IEEE Transactions on*, 59(3):781–786, March 2014.

[4] Y. Guo and L.E. Parker. A distributed and optimal motion planning approach for multiple mobile robots. In *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, volume 3, pages 2612–2619, 2002.

[5] M.T. Hale and M. Egerstedt. Cloud-based optimization: A quasi-decentralized approach to multi-agent coordination. In *Decision and Control (CDC), IEEE 53rd Annual Conference on*, pages 6635–6640, 2014.

[6] M.T. Hale, A. Nedić, and M. Egerstedt. Cloud-based centralized/decentralized multi-agent optimization with communication delays. arXiv:math.OC/1508.06230, 2015.

[7] F. Kelly, A. Maulloo, and D. Tan. Rate control in communication networks: shadow prices, proportional fairness and stability. In *Journal of the Operational Research Society*, volume 49, 1998.

[8] M. Khan, G. Pandurangan, and V.S.A. Kumar. Distributed algorithms for constructing approximate minimum spanning trees in wireless sensor networks. *Parallel and Distributed Systems, IEEE Transactions on*, 20(1):124–139, Jan 2009.

[9] I. Konnov. *Equilibrium models and variational inequalities*, volume 210. Elsevier, 2007.

[10] J. Koshal, A. Nedić, and U. Shanbhag. Multiuser optimization: Distributed algorithms and error analysis. *SIAM Journal on Optimization*, 21(3):1046–1081, 2011.

[11] H. W. Kuhn and A. W. Tucker. Nonlinear programming. In *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, pages 481–492, Berkeley, Calif., 1951.

[12] I. Lobel and A. Ozdaglar. Distributed subgradient methods for convex optimization over random networks. *Automatic Control, IEEE Transactions on*, 56(6):1291–1306, June 2011.

[13] D. Mitra. An asynchronous distributed algorithm for power control in cellular radio systems. In *Wireless and Mobile Communications*, pages 177–186. Springer, 1994.

[14] M.H. Nazari, Z. Costello, M.J. Feizollahi, S. Grijalva, and M. Egerstedt. Distributed frequency control of prosumer-based electric energy systems. *Power Systems, IEEE Transactions on*, 29, November 2014.

[15] G. Notarstefano and F. Bullo. Network abstract linear programming with application to cooperative target localization. In *Modelling, Estimation and Control of Networked Complex Systems*, Understanding Complex Systems, pages 177–190. 2009.

[16] L. Panait and S. Luke. Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, 11(3):387–434, 2005.

[17] M. Rabbat and R. Nowak. Distributed optimization in sensor networks. In *Information Processing in Sensor Networks, 2004. IPSN 2004. Third International Symposium on*, pages 20–27, April 2004.

[18] D.E. Soltero, M. Schwager, and D. Rus. Decentralized path planning for coverage tasks using gradient descent adaptive control. *The International Journal of Robotics Research*, 2013.

[19] N. Trigoni and B. Krishnamachari. Sensor network algorithms and applications: Introduction. *Philosophical Transactions of the Royal Scoeity A - Mathematical, Physical, and Engineering Sciences*, 370(1958, SI):5–10, JAN 13 2012.

[20] J. Tsitsiklis. *Problems in Decentralized Decision making and Computation*. PhD thesis, Massachusetts Institute of Technology, 1984.

[21] H. Uzawa. Iterative methods in concave programming. *Studies in Linear and Non-Linear Programming*, 1958.