Joint 48th IEEE Conference on Decision and Control and
28th Chinese Control Conference
Shanghai, P.R. China, December 16-18, 2009

Asynchronous Distributed Optimization with Minimal Communication and Connectivity Preservation

Minyi Zhong and **Christos G. Cassandras**
Division of Systems Engineering
and Center for Information and Systems Engineering
Boston University, mzhong@bu.edu, cgc@bu.edu

Abstract— **We consider problems where multiple agents cooperate to control their individual state so as to optimize a common objective while communicating with each other to exchange state information. Since communication costs can be significant, we seek conditions under which communication of state information among nodes can be minimized while still ensuring that the optimization process converges. In prior work, an asynchronous (event-driven) optimization scheme was proposed that limits communication to instants when some state estimation error function at a node exceeds a threshold. It was shown that convergence is guaranteed under no communication delays. In this paper, we first prove that convergence is still guaranteed under bounded communication delays. Next, we propose a decentralized mechanism which provably preserves network connectivity by using each node's routing information. We apply the optimization scheme to a sensor network coverage control problem where the objective is to maximize the probability of detecting events occurring in a region and show that the proposed asynchronous approach significantly reduces communication costs, hence also prolonging the system's lifetime, without any performance degradation.**

Keywords: Cooperative Control, Distributed Systems, Distributed Optimization, Sensor Networks

I. Introduction

The need for distributed optimization arises in settings which involve multiple controllable agents cooperating toward a common objective without a central controller to coordinate their actions. The cooperating agents define a dynamic system which may be thought of as a network with each agent corresponding to a node maintaining its own state s_i, $i = 1, \ldots, N$. The goal of each node is to control its state so as to optimize some system-wide objective expressed as a function of $\mathbf{s} = [s_1, \ldots, s_N]$ and possibly the state of the environment. Clearly, to achieve such a goal, the nodes must share, at least partially, their state information. However, this may require a large amount of information exchange and becomes a critical issue when the system consists of wirelessly communicating nodes which are often small, inexpensive devices with limited resources (e.g., a sensor network). Aside from energy required to move (if nodes are mobile), communication is known to be by far the largest consumer of the limited energy of a node [1], compared to other functions such as sensing and computation. Therefore, it is crucial to reduce communication between nodes to the minimum possible. Standard synchronization schemes require that nodes frequently exchange state information, which is clearly inefficient and, in fact, unnecessary since often the state of a node may not have changed much or may have only changed in a predictable way. This motivated the work in [2] to seek *asynchronous* optimization mechanisms in which a node communicates with others only when it considers it indispensable and only as a last resort.

The general setting described above applies to problems where the nodes may be vehicles seeking to maintain some desirable formation [3]. The system may also be a sensor network whose nodes must be placed so as to ensures high-quality monitoring in a given region, e.g., [4],[5]; this is often referred to as a "coverage control" problem. In some cases, the state of a node may be its perception of the environment which changes based on data directly collected by that node or communicated to it by other nodes, e.g., [6],[7]; consensus problems fall in this category.

In [2] we considered a network of N cooperating nodes, whose goal is to minimize an objective function $H(\mathbf{s})$ known to all nodes with every node controlling its individual state $s_i \in \mathbb{R}^{n_i}$, $i = 1, \ldots, N$. The synchronous state update scheme employed by the ith node is of the general form

$$s_i(k+1) = s_i(k) + \alpha_i d_i(\mathbf{s}(k)), \ k = 0, 1, \ldots \quad (1)$$

where α_i is a constant positive step size and $d_i(\mathbf{s}(k))$ is an *update direction* evaluated at the kth *update event* (see also [8]). We view $s_i(k+1)$ as the desired state determined at the kth update event and assume that local control is capable of reaching $s_i(k+1)$ from $s_i(k)$ within a time interval shorter than the time between update events.

A key difficulty in (1) is that $\mathbf{s}(k)$ is in fact not fully known to node i. Thus, $d_i(\mathbf{s}(k))$ has to be evaluated by synchronizing all nodes to provide their states to node i at the time its kth update event takes place. This is extremely costly in terms of communication and assumes no delays so that the state information is accurate. Alternatively, node i can evaluate $d_i(\cdot)$ using estimates of s_j for all $j \neq i$ relying on prior information from node j and possibly knowledge of its dynamics. Our concern is with determining instants when a node j may communicate its state to other nodes through what we term *communication events*. We note that such communication events occur at different times for each node,

The authors' work is supported in part by NSF under Grants DMI-0330171 and EFRI-0735974, by AFOSR under grant FA9550-07-1-0361 and FA9550-09-1-0095, and by DOE under grant DE-FG52-06NA27490.

as do each node's state update events, so that the resulting mechanism is fully asynchronous. The scheme proposed in [2] is based on each node j maintaining an *error function* of its actual state relative to its state as estimated by other nodes (which node j can evaluate). The node then transmits its actual state at time t only if this error function at t exceeds a given *threshold* δ_j. Assuming negligible communication delays, it was proved in [2] that by varying this threshold appropriately the resulting optimization scheme converges and leads to a local minimum of $H(\mathbf{s})$.

This paper contains two contributions that extend the results of [2]. First, we allow communication delays to be non-negligible but bounded and show that the resulting optimization mechanism still converges to a minimum of $H(\mathbf{s})$. Our analysis is based on the distributed optimization framework in [8], but our emphasis is on controlling the asynchronous occurrence of communication events through the threshold-based scheme. Second, we study how mobile nodes carrying out distributed optimization algorithms can preserve connectivity of the underlying communication network. Since connectivity is a global network property, distributed connectivity preservation which allows both link addition and deletion is a challenging task. In [9], decentralized power iteration is used to estimate the algebraic connectivity of a graph. In [10], each node locally estimates the network graph and a market-based auction mechanism is proposed so that deletion of a communication link will not disconnect the graph. The approach we take utilizes the information from a wireless routing algorithm [11] running in parallel with the optimization process. Finally, in case the network is disconnected, we include a simple scheme for re-establishing connectivity.

II. Asynchronous Distributed Optimization Framework With Communication Delays

We first review the asynchronous distributed optimization framework developed in [2]. In a setting where N cooperating nodes seek to optimize a common objective function, there are two processes associated with each node: a *state update process* and a *state communication process*. We begin with the former.

Let t_k, $k = 1, 2, \ldots$, denote the time when any one node performs a state update, e.g., using (1). Such update events can occur at a node either periodically or according to some local node-based policy. However, we will assume that every node performs an update with sufficient frequency relative to the updates of other nodes (this assumption will be stated precisely later).

Let $\mathcal{C}^i$ be the set of indices in $\{t_k\}$ corresponding to update events at node i. As an example, in Fig. 1, where nodes 1 and 2 perform state updates at $\{t_1, t_4, t_5\}$ and $\{t_2, t_3, t_6\}$ respectively, we have $\mathcal{C}^1 = \{1, 4, 5\}$, and $\mathcal{C}^2 = \{2, 3, 6\}$. We will set $d_i(\mathbf{s}(k)) = 0$ in (1) for all $k \notin \mathcal{C}^i$, i.e., $s_i(k+1) = s_i(k)$ if $k \notin \mathcal{C}^i$. We refer to any state update at such $k \notin \mathcal{C}^i$ as a *null step* at node i. Consider some t_k with $k \in \mathcal{C}^i$, we let $\mathbf{s}^i(k)$ be a vector with node i's estimates of all node states at that time, i.e., an estimate of $\mathbf{s}(k)$. With

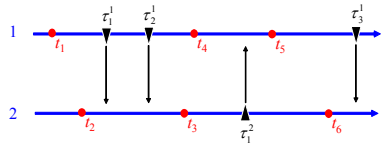

Fig. 1. State update and communication processes for two nodes. Red dots represent state update events at a node and black triangles represent state communication events.

$\mathbf{s}^i(k)$ available at each node, it no longer has to perform the state synchronization as required by (1) and can use the following asynchronous distributed state update scheme,

$$s_i(k+1) = s_i(k) + \alpha d_i\left(\mathbf{s}^i(k)\right) \text{ for } k = 0, 1, \ldots \quad (2)$$

Next, let us discuss the state communication process. Let τ_n^{ij} be the nth time when node i sends its true state to node j, $n = 1, 2, \ldots$ Let us also set $\tau_0^{ij} = 0$ for all i, j. We assume that at all communication event times the state information sent by node i can reach its destination node with a bounded delay. For the range of applications that motivate our work, we view it as a system requirement that the communication network is connected and we will discuss mechanisms to ensure such connectivity in Section IV.

Now let us consider what criterion a node i might use to generate its communication events, recalling that we aim to reduce communication costs. If node i knows that node j uses a specific method to estimate its state, then node i can evaluate that estimate and hence the error in it. We use $x_i(t)$ to denote a node state at any time t and observe that $s_i(k) = x_i(t_k)$. Let $x_i^j(t)$ be the estimate of $x_i(t)$ evaluated by node $j \neq i$ at time t, we can define an estimation *error function* $g(x_i(t), x_i^j(t))$, which measures the quality of the state estimate of node i with the requirement that

$$g(x_i(t), x_i^j(t)) = 0 \quad \text{if } x_i(t) = x_i^j(t) \quad (3)$$

Examples of $g(x_i(t), x_i^j(t))$ include $\|x_i(t) - x_i^j(t)\|_1$ and $\|x_i(t) - x_i^j(t)\|_2$. Let $\delta_i(k)$ be an error *threshold*, determined by node i after the kth state update event such that $k \in \mathcal{C}^i$. Thus, $\delta_i(k) = \delta_i(k-1)$ if $k \notin \mathcal{C}^i$. Let $\tilde{k}_t^i$ be the index of the most recent state update time of node i up to t, i.e.,

$$\tilde{k}_t^i = \max\left\{n : n \in \mathcal{C}^i,\ t_n \leq t\right\} \quad (4)$$

In [2], where negligible communication delay is assumed, the communication event policy at node i with respect to node j is determined by

$$\tau_n^{ij} = \inf\left\{t : g(x_i(t), x_i^j(t)) \geq \delta_i(\tilde{k}_t^i),\ t > \tau_{n-1}^{ij}\right\} \quad (5)$$

In other words, node i communicates its state to another node j only when it detects that its true state deviates from that node's estimate of it by at least the threshold $\delta_i(\tilde{k}_t^i)$. The black curve in Fig. 2 illustrates what the trajectory of $g(x_i(t), x_i^j(t))$ would look like when communication delay is negligible and (5) is applied (for clarity of presentation, the error bound $\delta_i(k)$ is treated as constant and the error function is continuous over all intervals $(\tau_{n-1}^{ij}, \tau_n^{ij})$, $n = 1, 2, \ldots$)

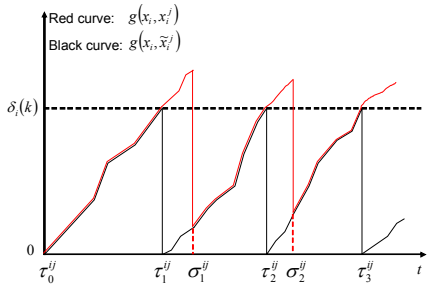

Fig. 2. Example of error functions trajectories with non-negligible communication delays.

Note that the negligible communication delay assumption allows the value of $g(x_i(t), x_i^j(t))$ to be instantaneously reset to 0 at τ_n^{ij}, $n = 1, 2, \ldots$

When the communication delay is non-negligible, after a communication event originates from node i to node j at τ_n^{ij}, node j will not receive it immediately. Therefore, the error function $g(x_i(t), x_i^j(t))$ continues to grow until the message containing $x_i(\tau_n^{ij})$ is received by node j. Let σ_n^{ij} denote the time when a message sent by node i at τ_n^{ij} is received at node j. The communication delay is assumed to be bounded. In particular, $\sigma_n^{ij} - \tau_n^{ij}$ for $n = 1, 2, \ldots$ is bounded, which is crucial to limiting the state estimation error $g(x_i(t), x_i^j(t))$.

If we still use the policy in (5), then between τ_n^{ij} and σ_n^{ij} the value of $g(x_i(t), x_i^j(t))$ generally remains above δ_i (as shown in Fig. 2). As a result, node i will continuously trigger communication events until σ_n^{ij}. To suppress this redundancy over the interval $[\tau_n^{ij}, \sigma_n^{ij}]$, we need to modify (5). Doing so requires carefully differentiating between the way in which node j estimates the state of node i in the presence of communication delays as opposed to no delays. Thus, let us express such a general-purpose estimate as follows:

$$x_i^j(t) = \begin{cases} h_i^j(t; \tau_n^{ij}) & \rho_n < t < \rho_{n+1}, \quad n = 0, 1, 2, \ldots \\ x_n & t = \rho_n, \quad n = 0, 1, 2, \ldots \end{cases} \tag{6}$$

where $h_i^j(\cdot)$ denotes the specific functional form of the estimator used by node j to estimate the state of node i and τ_n^{ij} is viewed as a parameter of the estimator. The estimate is defined over an interval (ρ_{n-1}, ρ_n) and ρ_n is the nth time when the estimate is reset to a given value x_n. This provides a formal representation of the estimation process in our optimization framework: At instants ρ_n, $n = 1, 2, \ldots$, node j resets its estimate of node i's state to some value x_n and then repeats the estimation process with this new initial condition over the next interval (ρ_n, ρ_{n+1}). In the case of negligible communication delays ([2]), we have

$$\rho_n = \tau_n^{ij}, \quad x_n = x_i(\tau_n^{ij}) \tag{7}$$

that is, the nth reset time is the instant when a communication event is generated from node i to j, and the reset value is the corresponding actual state of i at that time. However, in the presence of communication delays, we must set

$$\rho_n = \sigma_n^{ij}, \quad x_n = h_i^j(\sigma_n^{ij}; \tau_n^{ij}) \tag{8}$$

where $\sigma_n^{ij} > \tau_n^{ij}$ and the reset value depends on the estimation method used by node j. In view of this discussion, we define the new communication event policy

$$\tau_n^{ij} = \inf \left\{ t : g(x_i(t), h_i^j(t; \tau_{n-1}^{ij})) \geq \delta_i(\tilde{k}_t^i), \ t > \tau_{n-1}^{ij} \right\} \tag{9}$$

with the initial condition $\tau_0^{ij} = 0$. Next we introduce a new variable $\tilde{x}_i^j(t)$, defined as:

$$\tilde{x}_i^j(t) = \begin{cases} h_i^j(t; \tau_{n-1}^{ij}) & \tau_{n-1}^{ij} < t < \tau_n^{ij}, \quad n = 0, 1, 2, \ldots \\ x_i(\tau_n^{ij}) & t = \tau_n^{ij}, \quad n = 0, 1, 2, \ldots \end{cases} \tag{10}$$

In other words, $\tilde{x}_i^j(t)$ is the estimate of node i's state used by node j as if there were no communication delays, consistent with (7). This variable is maintained by node i.

Figure 2 illustrates the difference between a trajectory of $g(x_i(t), \tilde{x}_i^j(t))$ (black) and a trajectory of $g(x_i(t), x_i^j(t))$ (red) under (9). Note that if the communication delay is zero, then $g(x_i(t), \tilde{x}_i^j(t))$ and $g(x_i(t), x_i^j(t))$ will obviously follow the exact same trajectory.

Regarding the possible forms that $h_i^j(t; \tau_n^{ij})$ can take, we will discuss two cases. First, node j can use a static state estimation method, i.e., $h_i^j(t; \tau_n^{ij}) = x_i(\tau_n^{ij})$. At σ_n^{ij}, it simply sets its estimate of i to the value contained in the received message:

$$x_i^j(\sigma_n^{ij}) = x_i(\tau_n^{ij}) \tag{11}$$

In this case, $\tilde{x}_i^j(t) = x_i(\tau_n^{ij})$ for all $t \in [\tau_n^{ij}, \tau_{n+1}^{ij})$, in accordance with (10). On the other hand, if node j uses a dynamic state estimation method, the value of $x_i^j(\sigma_n^{ij})$ depends on τ_n^{ij} as well. For example, node j could uses a linear projection, i.e., $h_i^j(t; \tau_n^{ij}) = x_i(\tau_n^{ij}) + (t - \tau_n^{ij}) \cdot \alpha_i \cdot d_{i,\tau_n^{ij}} / \Delta_i$, where Δ_i is an estimate of the average time between state updates at node i (e.g., a known constant if node i performs periodic updates) and $d_{i,\tau_n^{ij}}$ is the update direction communicated by node i at time τ_n^{ij} along with its state. Then we have

$$x_i^j(\sigma_n^{ij}) = x_i(\tau_n^{ij}) + (\sigma_n^{ij} - \tau_n^{ij}) \cdot \alpha_i \cdot d_{i,\tau_n^{ij}} / \Delta_i \tag{12}$$

Note that in this case evaluating the state estimate in (12) requires knowledge of τ_n^{ij}, i.e., the message sent by node i must be time-stamped (otherwise, in the presence of random delays, node j cannot infer the value of τ_n^{ij} when it receives a message at σ_n^{ij}). This, in turn, requires a clock synchronization mechanism across nodes.

Finally, we discuss the way in which the threshold $\delta_i(k)$ should be selected. The basic idea is to use a large value at the initial stages of the optimization process and later reduce it to ultimately ensure convergence. One of the difficulties is in selecting an appropriate initial value for $\delta_i(k)$ which, if too large, may prevent any communication. The approach we follow is to control $\delta_i(k)$ in a manner which is proportional to $\|d_i(\mathbf{s}^i(k))\|_2$, the Euclidean norm of the update direction at the kth update event henceforth denoted by $\|\cdot\|$. Thus, let

$$\delta_i(k) = \begin{cases} K_\delta \|d_i(\mathbf{s}^i(k))\| & \text{if } k \in \mathcal{C}^i \\ \delta_i(k-1) & \text{otherwise} \end{cases} \tag{13}$$

where K_δ is a positive constant. We also impose an initial condition such that

$$\delta_i(0) = K_\delta \left\| d_i\left(\mathbf{s}^i(0)\right)\right\|, \quad i = 1, \ldots, N \tag{14}$$

where $s_j^i(0) = x_j(0)$. Clearly, the computation in (13) requires only local information.

III. Convergence Analysis

In this section, we study the convergence properties of (2). For simplicity, a common step size α is used, but each node may easily adjust its step size by incorporating a scaling factor into its own $d_i\left(\mathbf{s}^i(k)\right)$. Assumptions 1-4 below were discussed in greater detail in [2], while Assumption 5 is an additional one regarding communication delays.

Assumption 1. There exists a positive integer B such that for every $i = 1, \ldots, N$ and $k \geq 0$ at least one of the elements of the set $\{k - B + 1, k - B + 2, ..., k\}$ belongs to $\mathcal{C}^i$.

This assumption ensures that each node updates at least once in a period where B state update events take place.

Assumption 2. The objective function $H(\mathbf{s})$, where $\mathbf{s} \in \mathbb{R}^m$, $m = \sum_{i=1}^N n_i$, satisfies the following:

(a) $H(\mathbf{s}) \geq 0$ for all $\mathbf{s} \in \mathbb{R}^m$

(b) $H(\cdot)$ is continuously differentiable and $\nabla H(\cdot)$ is Lipschitz continuous, i.e., there exists a constant K_1 such that for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^m$, $\|\nabla H(\mathbf{x}) - \nabla H(\mathbf{y})\| \leq K_1 \|\mathbf{x} - \mathbf{y}\|$.

In what follows, we shall take all vectors to be column vectors and use $'$ to denote a transpose. For simplicity, we will henceforth write $d_i(k)$ instead of $d_i(\mathbf{s}^i(k))$ and let $d(k) = [d_1(k)', \ldots, d_N(k)']'$.

Assumption 3. There exist positive constants K_2 and K_3 such that for all $i = 1, \ldots, N$ and $k \in \mathcal{C}^i$, we have

(a) $d_i(k)' \nabla_i H\left(\mathbf{s}^i(k)\right) \leq -\|d_i(k)\|^2 / K_3$

(b) $K_2 \left\|\nabla_i H\left(\mathbf{s}^i(k)\right)\right\| \leq \|d_i(k)\|$

Here $\nabla_i H\left(\mathbf{s}^i(k)\right)$ denotes a vector with dimension n_i. This assumption is immediately satisfied with $K_2 = K_3 = 1$ when we use an update direction $d_i(k) = -\nabla_i H\left(\mathbf{s}^i(k)\right)$.

Assumption 4. There exists a positive constant K_4 such that the error function satisfies $\|x_i(t) - x_i^j(t)\| \leq K_4 g(x_i(t), x_i^j(t))$ for all i, j, t

In the common case where $g(x_i(t), x_i^j(t)) = \left\|x_i(t) - x_i^j(t)\right\|$, this is obviously satisfied with $K_4 = 1$.

Assumption 5. There exists a non-negative integer D such that for all i, j, n and k, if $\tau_n^{ij} < t_{k-D}$, then $\sigma_n^{ij} < t_k$.

In other words, we assume that at most D state update events can occur between a node sending a message and all destination nodes receiving this message.

In Theorem 1, we consider only the static estimation case which is the least costly in terms of estimation complexity. Moreover, static estimation does not require a time synchronization mechanism which (12), for example, does. Due to space limitations, all proofs in this paper are omitted.

Theorem 1: Under Assumptions 1-5, the communication event policy (9), the state update scheme (2), and the static estimation method (11), if the error threshold $\delta_i(k)$ controlling communication events is set by (13)-(14), then there exist positive constants α and K_δ such that $\lim_{k\to\infty} \nabla H(\mathbf{s}(k)) = 0$.

IV. Connectivity Preservation

We have thus far assumed that nodes communicate over a connected network. However, with wireless mobile nodes, this assumption is often violated unless connectivity is explicitly preserved. For example, in coverage control missions, sensor nodes tend to spread out to cover remote regions; this increases hop distance and often disconnects critical links. In this section, we address the issue of connectivity preservation in conjunction with the proposed distributed optimization framework. In particular, we seek to ensure the following two requirements: (i) The distance between two communicating nodes is less than a certain threshold C. This is to maintain link quality (i.e., a sufficiently high SNR). We define the following Boolean variable to indicate whether two nodes i and j satisfy this requirement (s_i denotes node i's 2D location):

$$c_1(s_i, s_j) = \begin{cases} 1, & \|s_i - s_j\| \leq C \\ 0, & \text{otherwise} \end{cases}$$

(ii) In the presence of obstacles, line-of-sight between two communicating nodes is maintained. We define:

$$c_2(s_i, s_j) = \begin{cases} 1, & as_i + (1-\alpha)s_j \in F \text{ for all } \alpha \in [0,1] \\ 0, & \text{otherwise} \end{cases}$$

where F denotes the free portion of the mission space unoccupied by obstacles. Links satisfying (i) and (ii) are termed *strong* links and we define $c(s_i, s_j) = c_1(s_i, s_j) \cdot c_2(s_i, s_j)$, so that i and j form a *strong* link if and only if $c(s_i, s_j) = 1$.

Let us represent the network of mobile nodes by a graph $\mathcal{G}(\mathbf{s}) = (\mathcal{N}, \mathcal{E}(\mathbf{s}))$, where $\mathcal{N} = \{0, 1, ..., N\}$ is the set of node indices including the base station denoted by 0, and $\mathcal{E}(\mathbf{s}) = \{(i,j) : i, j \in \mathcal{N},\ i \neq j,\ c(s_i, s_j) = 1\}$, which is the set of all *strong* links. Over $\mathcal{G}(\mathbf{s})$, let $\overline{\Pi}_i$ be the set of all possible loop-free paths from node i to 0. If $\overline{\Pi}_i \neq \emptyset$ for all $i \in \{1, ..., N\}$, the graph $\mathcal{G}(\mathbf{s})$ is *connected*. Our goal is to preserve this property.

We assume that the network operates with a distributed routing algorithm in parallel with our distributed optimization algorithm. The routing algorithm proactively generates and maintains a set of paths, denoted by $\widehat{\Pi}_i$, for each node i to forward data to 0. Note that the routing algorithm is not restricted to use only links in $\mathcal{E}(\mathbf{s})$. Let $\Pi_i = \overline{\Pi}_i \cap \widehat{\Pi}_i$ be the set of paths in $\widehat{\Pi}_i$ which contains only *strong* links. Denote the kth path in Π_i by $\pi_{i,k}$, represented by a *sequence* (*ordered* set) which contains all the nodes on this path from i and 0 (including i and 0) ordered by their hop counts. If we use $\pi_{i,k}^j$ to denote the jth element in $\pi_{i,k}$, node $\pi_{i,k}^j$ is $j-1$ hops away from node i on the path $\pi_{i,k}$. We require that the path be loop-free, thus $\pi_{i,k}$ cannot contain duplicate elements, i.e., $\pi_{i,k}^j \neq \pi_{i,k}^l$, if $j \neq l$. Since $\overline{\Pi}_i$ only uses links in $\mathcal{E}(\mathbf{s})$, $c(\pi_{i,k}^j, \pi_{i,k}^{j+1}) = 1$, for all i, j, k. For example, in Fig. 3, $\overline{\Pi}_5 = \{\{5,4,2,0\}, \{5,3,1,0\}, \{5,3,1,2,0\}, \{5,4,2,1,0\}\}$ and depending on the result of the routing algorithm, Π_5 is some subset of $\overline{\Pi}_5$. The routing algorithm also maintains at node i its *upstream* (further from the base station 0) node

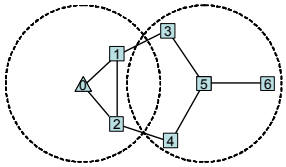

Fig. 3. Base station is labeled 0. Desired communication range is marked for node 0 and 5. *Strong* links are indicated by lines between nodes.

set, denoted by $\mathcal{U}_i = \cup_{j,k} \omega_i(\pi_{j,k})$ where

$$\omega_i(\pi_{j,k}) = \begin{cases} \pi_{j,k}^{l-1}, & \text{if } i \in \pi_{j,k},\ i \neq j \text{ and } i = \pi_{j,k}^{l} \\ \emptyset, & \text{else} \end{cases}$$

In addition, it maintains the *downstream* node set $\mathcal{D}_i = \{j : i \in \mathcal{U}_j,\ j \in \{0, ..., N\}\}$.

The information in Π_i, $\mathcal{U}_i$ and $\mathcal{D}_i$ is essential to our connectivity preservation algorithm. By piggybacking on the routing algorithm, nodes reduce the communication cost incurred for connectivity preservation. Before proceeding, we also define a projection of $x \in \mathbb{R}^2$ on a set $\mathcal{S} \subset \mathbb{R}^2$ as $P_{\mathcal{S}}(x) = \arg\min_{y \in \mathcal{S}} \|x - y\|$. Next, let $\mathcal{X}(s) = \left\{x : x \in \mathbb{R}^2,\ c(x, s) = 1\right\}$ be the region where a *strong* link with s can be established. Finally, for simplicity, henceforth s_j stands for $s_j(k)$ unless expressly specified otherwise, where k indexes state update times t_k.

Algorithm 1: When node i makes a state update at t_k, $k \in \mathcal{C}^i$, using (1), it takes the following steps:

S1) Using (1), generate a candidate of the next state: $\hat{s}_i = s_i(k) + \alpha_i \cdot d_i(\mathbf{s}(k))$.

S2) If for all $j \in \mathcal{D}_i$, $c(\hat{s}_i, s_j) = 0$, $\rightarrow$ S3; else, $\rightarrow$ S5.

S3) If there exists a node $j \neq i$ such that $s_j \in \mathcal{X}(\hat{s}_i)$ and there exists a path $\pi_{j,l} \in \Pi_j$, such that $i \notin \pi_{j,l}$, go to S5; else, go to S4.

S4) Select some $j \in \mathcal{D}_i$ and project $\hat{s}_i$ on $\mathcal{X}(s_j)$. Redefine the result $P_{\mathcal{X}(s_j)}(\hat{s}_i)$ as $\hat{s}_i$.

S5) If for all $j \in \mathcal{U}_i$, $c(\hat{s}_i, s_j) = 1$, $\rightarrow$ S7; else, $\rightarrow$ S6.

S6) If for all $j \in \mathcal{U}_i$ such that $c(\hat{s}_i, s_j) = 0$ there exists a path $\pi_{j,l} \in \Pi_j$, such that either $i \notin \pi_{j,l}$ or $i \in \pi_{j,l}$ and $c(\hat{s}_i, s_r) = 1$ with $r = \omega_i(\pi_{j,l})$, go to S7; else $s_i(k+1) = s_i(k)$ and skip S7.

S7) $s_i(k+1) = \hat{s}_i$.

It should be pointed out that in S3 and S5, node i has to communicate with nodes in $\mathcal{X}(\hat{s}_i)$ and $\mathcal{U}_i$ respectively. In S4, when the set $\mathcal{X}(s_j)$ is a disk, the projection is straightforward. When $\mathcal{X}(s_j)$ is nonconvex due to obstacles, the projection involves finding the closest point to $\hat{s}_i$ on some line segments and some circular arcs.

Theorem 2: Assuming only one node performs a state update at any given time and $\Pi_j \neq \emptyset$ for all $j \in \{1, ..., N\}$ before the state update, an iteration of Algorithm 1 preserves the connectivity of $\mathcal{G}(\mathbf{s})$.

Asynchronous iterations: With the asynchronous method in Section II, node i cannot accurately evaluate $c(\hat{s}_i, s_j)$ because i only has an estimate, s_j^i, of node j's state. However, based on our analysis in previous sections, the error between s_j^i and s_j always has an upper bound known to j. Let us denote this upper bound of $\left\|s_j^i - s_j\right\|$ by λ_j and assume that node j includes λ_j in its communication messages to i. Based on λ_j and s_j^i, node i can conclude that $s_j \in \left\{x : \left\|s_j^i - x\right\| \leq \lambda_j\right\} = \Phi_j^i$ and use

$$c'\left(s_i, \Phi_j^i\right) = \begin{cases} 1, & c(s_i, x) = 1 \text{ for all } x \in \Phi_j^i \\ 0, & \text{otherwise} \end{cases} \tag{15}$$

to replace $c(s_i, s_j)$ in Algorithm 1. The test in (15) is rather conservative as $c'(s_i, \Phi_j^i)$ might be 0 when a *strong* link actually exists between i and j. However, note that in Algorithm 1, node i needs to communicate with neighboring nodes when some Boolean variables are 0. The information obtained from these communications may help node i detect those false positives.

Fault recovery: When two or more nodes update their states at the same time, Algorithm 1 does not guarantee connectivity, thus a fault recovery mechanism is required. Connectivity loss may also be caused by bad initial conditions, excessive communication delays, or node failure. When a node detects that the base station is unreachable, a connection can be re-established by navigating back to the base station (a known position). As soon as that node establishes a communication link with the base station or any other node with a path to the base station, it can use Algorithm 1 to maintain connectivity.

V. Application to Coverage Control

In this section, we apply the asynchronous distributed optimization framework and the connectivity preservation algorithm to the class of coverage control problems mentioned in Section I. We will show how an asynchronous algorithm significantly reduces the energy expended on communication with no adverse effect on performance and the effectiveness of the connectivity preservation algorithm.

Following the formulation in [5], we define an event density function $R(x)$ over the *mission space* $\Omega \subset \mathbb{R}^2$, which captures the frequency of random event occurrences at $x \in \Omega$. $R(x)$ satisfies $R(x) \geq 0$ for all $x \in \Omega$ and $\int_\Omega R(x)\, dx < \infty$. We assume that when an event takes place, it will emit some signal which may be observed by some sensor nodes. The cooperative system consists of N mobile sensor nodes deployed into Ω to detect the random events. Their positions are denoted by $\mathbf{s} = (s_1, ..., s_N)$.

The probability that node i detects an event at $x \in \Omega$, denoted by $p_i(x, s_i)$, is a monotonically decreasing differentiable function of $\|x - s_i\|$. Since multiple independent sensor nodes are deployed, the joint probability that an event occurring at x is detected, denoted by $P(x, \mathbf{s})$, is given by

$$P(x, \mathbf{s}) = 1 - \prod_{i=1}^{N} \left[1 - p_i(x, s_i)\right]$$

and the optimization problem of interest is

$$\max_{\mathbf{s}} H(\mathbf{s}) = \int_\Omega R(x)\, P(x, \mathbf{s})\, dx$$

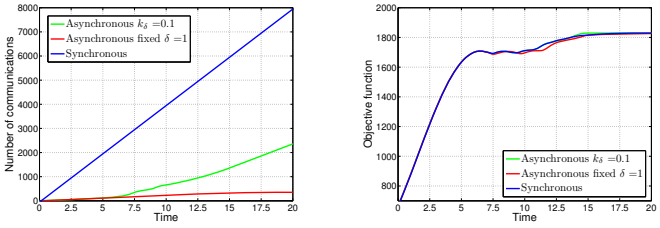

Fig. 4. Communication cost (left) and objective function trajectories (right) of 3 distributed optimization alogrithms with communication delays.

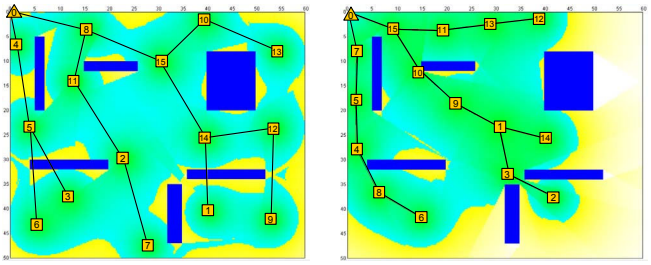

Fig. 5. Final node deployment after applying coverage control without (left) and with (right) connectivity preservation. The lines between nodes indicate communication links. In left, $H(\mathbf{s}) = 1642.1$. In right, Algorithm 1 is applied with $C = 10$ and $H(\mathbf{s}) = 1449.4$.

in which we use the locations of the sensor nodes as decision variables to maximize the event detection rate in Ω.

A *synchronous* gradient-based solution was obtained in [5] in which node i's next way point is determined by

$$s_i(k+1) = s_i(k) + \alpha \frac{\partial H(\mathbf{s})}{\partial s_i}, \quad k = 0, 1, \ldots \tag{16}$$

The evaluation of the gradient $\partial H(\mathbf{s})/\partial s_i$ needs the exact locations of nearby nodes and the communication involved to ensure such state synchronization has a high cost, which is often unnecessary because the locations of neighboring nodes may be accurately estimated, see (11), (12).

Next, we apply the asynchronous method developed in Section II to this problem in a simulated environment (*http://codescolor.bu.edu/simulators.html*) and compare the results with the synchronous approach. The simulation setup is identical to that described in [2] except that non-negligible communication delays are now included.

In Fig. 4, four nodes are deployed in a rectangular empty mission space. Every time a node broadcasts its state, the number of communications is increased by 1. This figure shows that the asynchronous method can substantially reduce the communication cost (hence, the energy consumption at all nodes) while convergence performance is virtually indistinguishable from that of the synchronous method. The asynchronous algorithm with fixed $\delta_i(k)$ has the added advantage that it usually stops incurring communication cost earlier than the other two methods. However, it does not guarantee convergence to stationary points.

In Fig. 5, we compare the results of coverage control with or without connectivity preservation. The base station is located at the upper-left corner and all nodes initially start from that location. Blue rectangles represent obstacles and the navigable areas are color-coded to indicate the quality of coverage (green means good coverage and white means no coverage). When Algorithm 1 is applied to the coverage control problem (right figure), no communication link violates the range limit $C = 10$ or line-of-sight constraints. As a price for guaranteed connectivity, the final objective function value reached is lower.

VI. Conclusions and Future Work

We have addressed the extent to which communication among agents can be minimized in a cooperative optimization setting. Extending the results of [2], we have shown that a scheme limiting communication events to "last resort only" when some state estimation error function at a node exceeds a threshold guarantees the convergence of a gradient-based fully asynchronous distributed algorithm under bounded communication delays. Further, to ensure network connectivity while optimization takes place, we have proposed a mechanism which utilizes each node's local routing information to limit the location a node is allowed to move to. We have applied this approach to a coverage control problem for wireless sensor networks and confirmed through numerical examples that limited asynchronous (event-driven) communication results in substantial energy savings with no compromise in performance. Future work aims at allowing multiple nodes to update at the same time and at generalizing the connectivity condition to include transmission power and angle constraints.

References

[1] V. Shnayder, M. Hempstead, B. R. Chen, G. W. Allen, and M. Welsh, "Simulating the power consumption of large-scale sensor network applications," in *In Sensys*. ACM Press, 2004, pp. 188–200.

[2] M. Zhong and C. G. Cassandras, "Asynchronous distributed optimization with minimal communication," in *Proc. of 47th IEEE Conf. on Decision and Control*, 2008, pp. 363–368.

[3] P. Ögren, E. Fiorelli, and N. E. Leonard, "Cooperative Control of Mobile Sensor Networks: Adaptive Gradient Climbing in a Distributed Environment," *IEEE Trans. Automatic Control*, vol. 49, no. 8, pp. 1292–1302, 2004.

[4] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Trans. on Robotics and Automation*, vol. 20, no. 2, 2004.

[5] C. G. Cassandras and W. Li, "Sensor networks and cooperative control," *European Journal of Control*, vol. 11, no. 4-5, pp. 436–463, 2005.

[6] A. Jadbabaie, J. Lin, and S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Trans. Automatic Control*, vol. 48, no. 6, pp. 988–1001, 2003.

[7] L. Moreau, "Stability of multi-agent systems with time-dependent communication links," *IEEE Trans. Automatic Control*, vol. 50, no. 2, pp. 169–182, 2005.

[8] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, 1997.

[9] P. Yang, R. Freeman, G. Gordon, K. Lynch, S. Srinivasa, and R. Sukthankar, " Decentralized estimation and control of graph connectivity in mobile sensor networks," in *Proc. of American Control Conference*, 2008, pp. 2678 – 2683.

[10] M. Zavlanos and G. Pappas, " Distributed connectivity control of mobile networks," *IEEE TRANSACTIONS ON ROBOTICS*, vol. 24, no. 6, pp. 1416–1428, 2008.

[11] J. Al-Karaki and A. Kamal, "Routing techniques in wireless sensor networks: a survey," *IEEE Wireless Communications*, vol. 11, no. 6, pp. 6 –28, 2004.