Distributed Coordination for Fast Iterative Optimization in Wireless Sensor/Actuator Networks

Rahul Balani, Nabil Hajj Chehade, Supriyo Chakraborty, Mani B. Srivastava
Electrical Engineering
University of California at Los Angeles, USA
{rahulb, nabilhc, supriyo, mbs}@ee.ucla.edu

***Abstract*—Large-scale coordination and control problems in sensor/actuator networks are often expressed within the networked optimization model. While significant advances have taken place in both first- and higher-order optimization techniques, their widespread adoption in practical implementations has been hindered by a lack of adequate programming and evaluation support. This motivates the two major contributions of this paper. First, we extend the distributed programming framework proposed in [1] with a synchronization primitive to implement different versions of the subgradient technique and perform extensive evaluation with varying deployment and algorithmic parameters. Second, the insights - obtained by observing the variability in practical metrics such as response time and incurred message cost - lead us to exploit the spatial locality inherent in these large-scale actuator control applications, and propose a novel consensus algorithm applied to the subgradient method. We show using simulations that there is at least 99% improvement in response time and the message cost is reduced by more than 90% over prior consensus based algorithms.**

I. Introduction

Wireless sensor nodes collect an enormous amount of data over space and time to estimate the state of the environment and control actuators to modify it. Large-scale estimation and control problems in several important applications – such as water-efficient irrigation, cooperative multi-target tracking, and personalized light control – can often be expressed as optimization of a convex cost function involving control inputs to actuators and sensor data. Consequently, programmers implement these applications by selecting an optimization algorithm that satisfies their performance requirements given energy constraints and network characteristics of the deployment. In practice, this choice is usually restricted by available programming and networking support for the underlying inter-node coordination.

In this paper, we identify the key requirements for supporting the subgradient method and expose them in our extended Application Programming Framework (APF). The subgradient method is a popular first-order convex optimization algorithm that iteratively estimates control inputs of actuators to optimize the objective function. Besides being applicable in a wide variety of application domains, its unique characteristic is that it allows both centralized [2] and distributed [3][4] implementations, enabling programmers to select the best version for their requirements and constraints. However, we focus on scenarios where a distributed implementation of the subgradient scheme is favored over its centralized counterpart due to energy constraints as discussed in [4].

In its classical distributed form, the subgradient method can be implemented in an *incremental* fashion where distributed controllers locally update the estimate of the optimizer in a desired sequence. This is the most message-efficient implementation for a deployment characterized by a ring topology. However, researchers in [5] recognized that implementation of a communication scheme to support these sequential iterations, on top of an adhoc wireless mesh network, can often be problematic in practice; more so when the sequence needs to be randomized in every iteration. Hence, they proposed to combine consensus negotiations with the subgradient method to relax the communication restraints by requiring coordination between neighboring nodes only. However, our analysis and simulations confirm that this is achieved at the cost of higher messaging overhead and slower rate of convergence that degrades application performance.

One of our main contributions in the paper is that we exploit *localized* spatio-temporal influence of actuators to improve the convergence rate of subgradient method when combined with consensus negotiations. We call it *Fast Consensus (FC)*. The proposed algorithm successfully demonstrates that local consensus between controllers is sufficient for convergence, rather than global consensus between all the nodes in the network as proposed by [5]. Section III outlines the algorithm and selection of consensus weights based on local subsets of nodes that are defined in terms of the physical properties of sensors and actuators instead of the network topology [5][6]. We formally prove that the subgradient method converges to optimal values of control inputs with FC and analytically demonstrate its faster rate of convergence over global consensus (GC) based method.

The APF supports consensus based algorithms through automatic identification of local node subsets at run-time using the *clique* discovery mechanisms outlined in our prior work [1]. It exposes a concise *Sync* interface to the programmer, described in Section V, that enables synchronization within the subsets via neighborhood-based information exchange mechanisms similar to [7][8]. In addition, support for hierarchical networks in the APF enables efficient execution of FC by splitting computation across actuators and controllers and reducing multi-hop routing delays.

Extensive simulations of a sample personalized light control application in Section VI confirm our analysis of FC and illustrate the consequential 99% improvement in *response time* and over 90% reduction in message cost of the application over GC. In this paper, we measure *rate of convergence* in number of subgradient iterations μ required for the value of objective function $f(x_\mu)$ to reach within an ϵ-ball of the optimal value $f(x^*)$. The time it takes for an algorithm to complete μ iterations for a given ϵ defines its *response time* (or *convergence latency*) in estimating desired control inputs x_μ. Interestingly, our evaluation also demonstrates that FC has a 50.5% faster response time on average than the incremental subgradient method despite its provably slower rate of convergence than the latter.

Although rate of convergence is typically used to evaluate iterative optimization algorithms, it is important to note that in practice, the user is ultimately concerned with the response time of the application. Therefore, an important contribution of our APF is that it provides a common platform to compare the performance of distributed algorithms in terms of a more practical metric like convergence latency. It also enables us to *realistically* analyze the effect of deployment and algorithm specific parameters like communication range, link MTU's and number of consensus iterations, on the performance metric of choice. Prior work could not take this into account as it was seldom accompanied with programming support necessary for this comparison, and hence, aimed for an implementation-agnostic metric that is observed to be often misleading in our evaluation. For instance, Section VI demonstrates that increasing communication range or number of consensus negotiations in GC have a negative effect on the response time of the application despite the improvement in rate of convergence as predicted by prior results [5].

II. Problem Formulation

We consider the following optimization problem

$$\min_{x} \quad f(x) = \sum_{i=1}^{N} f^i(x) \text{ s.t} \quad x \in \mathcal{X} \tag{1}$$

where $f^i : \mathbb{R}^M \to \mathbb{R}$ are convex functions and $\mathcal{X}$ is a nonempty, closed, and convex subset of $\mathbb{R}^M$. This class of optimization problems is found in optimal control for finite-time rendezvous of multiple dynamical agents, cooperative multi-target tracking, source localization, estimation and control in sensor/actuator networks, and resource allocation in computer networks. However, in this paper, we study this class of problems through a simpler example of an intelligent light control application commonly envisioned for huge office spaces, workshops and theatre/multi-media production. It is important to note that the ideas discussed here are applicable to others as well, for instance, through the application of Model Predictive Control to control problems in sprinkler irrigation [9] and building HVAC systems.

A. Intelligent Light Control

Consider a light control application with M light sources and N light sensors in a room. Each light sensor i corresponds to an occupant of the room and has an associated incident light intensity L_i^* desired by the user. The sensors also act as distributed controllers that regulate output intensities $\hat{I} = (I_1, .., I_M)^T$ of the light sources to achieve resultant light intensities such that the error between actual and desired intensities is minimized. This can be expressed as a convex optimization problem where the optimal control inputs $\hat{I}^*$ to the light sources are determined by

$$\hat{I}^* = \arg\min_{\hat{I}\in\mathcal{X}} \sum_{i=1}^{N} \left(L_i(\hat{I}) + \phi_i - L_i^* \right)^2 \tag{2}$$

$$\text{s.t.} \quad L_i(\hat{I}) = \sum_{j=1}^{M} a_{ij} I_j, \quad \forall i \in [1, N]$$

$$\mathcal{X} = \left\{ (I_1, .., I_M)^T \,|0 \le I_j \le I_{max}, \forall j \in [1, M] \right\}$$

where, $L_i : \mathbb{R}^M \to \mathbb{R}$ models the resultant light intensity at sensor i from M controllable light sources given $\hat{I}$, and ϕ_i represents the modelling error and incident light at sensor i from uncontrollable sources in the room like windows. Assuming the location and orientation of all light sources and sensors is fixed, model coefficients $a_{ij} \in [0, 1]$ are constant. I_{max} is the maximum output intensity of all light sources.

B. Subgradient Algorithm

A popular method for solving problems of type (1) is the subgradient algorithm, which consists of the following iterative procedure when applied to problem (2).

$$\hat{I}_k = \mathcal{P}_{\mathcal{X}} \left[\hat{I}_{k-1} - \alpha \sum_{i=1}^{N} g^i(\hat{I}_{k-1}) \right] \tag{3}$$

where $g^i(\hat{I})$ is a subgradient of f^i at $\hat{I}$, α is a constant positive stepsize, and $\mathcal{P}_{\mathcal{X}}$ denotes projection on the set $\mathcal{X} \subset \mathbb{R}^M$.

The subgradient method can also be implemented in an incremental fashion as proposed in [3]. For each k^{th} iteration, this entails changing the variable $\hat{I}_{k-1}$ incrementally through N *subiterations* to obtain $\hat{I}_k$, and in each step using only the subgradient corresponding to a single component function f^i. The procedure is shown below:

$$\psi_{i+1,k} = \mathcal{P}_{\mathcal{X}} \left[\psi_{i,k} - \alpha g^i(\psi_{i,k}) \right], \text{ with } \psi_{0,k} = \hat{I}_{k-1}, \tag{4}$$

where $\psi_{i,k}$ is the estimate at i^{th} subiteration of k^{th} iteration, and at the end of N subiterations, $\hat{I}_k = \psi_{N,k}$. Researchers in [4] observed that this method can be performed in a distributed way by circulating the estimate $\psi_{i,k}$ between the sensor nodes following a logical ring, where they perform a subiteration according to (4) using only a single subgradient corresponding to the node's component function f^i. This incremental subgradient scheme has advantages over the standard (3) in terms of rate of convergence.

C. Consensus Algorithms

Researchers in [5] studied the application of distributed average consensus algorithms to the subgradient method. They demonstrated that the communication requirements imposed by the cyclical subiterations could be relaxed by enabling the nodes to iterate in parallel using their component functions

f^i. Applied to the light control problem, the proposed method combines the subgradient iterations of (3) with φ consensus iterations/negotiations as shown below:

$$\begin{aligned} u_k^i &= \hat{I}_{k-1}^i - \alpha g^i(\hat{I}_{k-1}^i) \\ v_k^i &= \sum_{j=1}^{N} [W^\varphi]_{ij}\, u_k^j,\ i = 1,..,N \\ \hat{I}_k^i &= \mathcal{P}_{\mathcal{X}}\left[v_k^i\right] \end{aligned} \tag{5}$$

where $\hat{I}_k^i$ is the local copy of estimated control input at sensor i and $[W^\varphi]_{ij}$ denotes the $(i,j)^{th}$ element of W^φ. It basically states that starting with an initial estimate $\hat{I}_{k-1}^i$ at the beginning of each k^{th} iteration, all sensor nodes perform local *subgradient updates* in parallel to obtain their respective u_k^i. Then, each node runs φ consensus iterations with its communication neighbors to update its local estimate u_k^i with a weighted average of neighbors' estimates.

Ideally, in each k^{th} iteration, the average consensus negotiations try to drive *global* consensus towards

$$\begin{aligned} v_k^i &= \lim_{\varphi\to\infty} \sum_{j=1}^{N} [W^\varphi]_{ij}\, u_k^j \\ &= \frac{1}{N}\sum_{j=1}^{N} u_k^j = \hat{U}_k, \quad \forall i = 1,..,N \end{aligned} \tag{6}$$

i.e. the mean over all N sensors in the network. Johansson *et al.* [5] show that under these ideal conditions, this method is equivalent to the subgradient method with stepsize α/N.

However, in practical scenarios, the choice of W affects the accuracy and rate of convergence of the algorithm. $W \in \mathbb{R}^{N\times N}$ is the consensus weight matrix that must be stochastic (sum of each row = 1) and symmetric for the subgradient iterations to converge at the optimizer $\hat{I}^*$ [5][6]. It is typically derived from the communication graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ associated with the sensor network. $\mathcal{V} = \{1,..,N\}$ is the set of vertices (nodes) in the network and $\mathcal{E} \subseteq \mathcal{V}\times\mathcal{V}$ is the set of edges depicting 1-hop communication between their end points.

D. Analysis

In this section, we analytically compare GC with the incremental method in terms of message cost and convergence latency - parameters that effect human perceptible response time and battery lifetime. Our analysis, in combination with the simulations in Section VI, demonstrates that the incremental method performs better than GC in all practical scenarios.

Message cost: Ignoring the overhead of routing and ring maintenance in the incremental subgradient method, the total number of messages exchanged per node during the execution of the algorithm is 2μ, i.e. one message each to receive and transmit a token in each iteration. For the consensus-based algorithm, it is given by $\mu_c\varphi d_{c,a}$ where μ_c is the number of subgradient iterations and $d_{c,a}$ is the average size of 1-hop communication neighborhoods in $\mathcal{G}$. This calculation assumes one message to broadcast the local copy of estimates, and $d_{c,a} - 1$ messages to receive estimates from respective neighbors in each consensus iteration.

Prior research [5] has shown that the incremental method has a faster rate of convergence than GC. In fact, our simulations confirm this result and demonstrate that when $\varphi = 1$, μ is at least an order of magnitude lesser than μ_c (Tables II and III). For higher values of φ, μ_c decreases [5] but the product $\mu_c\varphi$ is nonetheless approximately two orders of magnitude greater than μ. Further, it is observed that $d_{c,a} \geq 2$ for connected networks, which leads us to the conclusion that the incremental subgradient algorithm is more message efficient than GC even if we include the ring maintenance overhead.

Latency: The time to complete each k^{th} iteration in these distributed subgradient methods is typically determined by computation (subgradient update) and communication (consensus negotiation or subiteration) latency. However, for low bitrate radios like CC2420 etc. in our target deployments, it has been shown that communication latency dominates computation by several orders of magnitude. Thus, the total convergence latency can be approximated by the communication latency.

As a result, the net response time of incremental subgradient method is $\mathcal{O}(\mu NM)$. It accounts for $\mathcal{O}(M)$ bits of the estimate $\hat{I}_k$ that circulate through a ring of N sensors in each iteration. The corresponding value for GC is given by $\mathcal{O}(\mu_c\varphi M d_{c,m})$ based on the observation that time to complete each consensus negotiation will be bounded by messaging between nodes in the largest communication neighborhood of $\mathcal{G}$ whose size is represented by $d_{c,m}$. It is important to note that although GC requires only local communication, the size of messages exchanged between nodes is still $\mathcal{O}(M)$, which significantly affects communication latency. Considering $\mu_c\varphi = K\mu$ where $K \geq 100$ from previous analysis, we can conclude that GC will exhibit a higher latency when $Kd_{c,m} > N$. This condition is satisfied in many realistic deployments with different communication topologies.

III. Fast Consensus

In this section, we describe our modification to the consensus algorithm so as to retain its flexibility, while simultaneously reducing the message and latency overhead. It is based on the observation that for each light source j, a *local* consensus on the value of its output intensity is only required between the subset of sensors γ_j that can measure its effect. As a result, in each k^{th} iteration, the proposed FC algorithm tries to drive local consensus between nodes in γ_j towards

$$\left[\hat{U}_k\right]_j = \frac{1}{|\gamma_j|}\sum_{l\in\gamma_j}\left[u_k^l\right]_j,\ \forall j = 1,..,M \tag{7}$$
$$\text{where}\quad \gamma_j = \{\, i \mid a_{ij} > 0,\ i \in [1,N]\}\,.$$

A. Algorithm

Local consensus in FC is achieved by selecting a separate W_j for each light source j and establishing communication between all nodes in γ_j. As a result, local consensus is achieved in only one iteration (instead of φ) for each subgradient update. With this proposed modification, the consensus iterations to obtain v_k^i in (5) can be broken down to calculate M components of v_k^i individually as

$$v_k^i = \left(\left[v_k^i\right]_1, \left[v_k^i\right]_2, .., \left[v_k^i\right]_M \right)^T$$
$$\text{where } \left[v_k^i\right]_j = \sum_{l=1}^{N} \left[W_j\right]_{il} \left[u_k^l\right]_j \ , \ j = 1, .., M. \tag{8}$$

Each $(p, q)^{th}$ element of W_j is determined by

$$[W_j]_{pq} = \begin{cases} \frac{1}{|\gamma_j|} & \text{if } p \in \gamma_j \text{ and } q \in \gamma_j, \\ 0 & \text{otherwise,} \end{cases} \tag{9}$$

which reduces (8) to

$$\left[v_k^i\right]_j = \begin{cases} \frac{1}{|\gamma_j|} \sum_{l\in\gamma_j} \left[u_k^l\right]_j & \text{if } i \in \gamma_j, \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

In contrast with prior consensus algorithms where W is derived from 1-hop communication graph $\mathcal{G}$, each W_j is instead closely tied to the physical properties of sensors and actuators that determine the membership of γ_j. Therefore, to reason about the communication properties of the FC algorithm, we define an *information exchange* graph $\mathcal{G}_{ie} = (\mathcal{V}, \mathcal{E}_{ie})$, where an edge $(i, l) \in \mathcal{E}_{ie}$ iff $\exists j$ such that $\{i, l\} \subseteq \gamma_j$. Our APF supports the information exchange as defined by $\mathcal{G}_{ie}$ through user-selectable two (multi-hop) network configurations that either involve only sensors (*Flat networks*), or include actuators as well (*Hierarchical networks*).

Section IV demonstrates that the proposed algorithm is advantageous when each actuator affects the measurements at only a small subset of sensors ($|\gamma_j| < N$) and each sensor i can sense the effect of only a small subset β_i of actuators ($|\beta_i| < M$). Both these properties are satisfied in many applications besides light control such as sprinkler irrigation, building HVAC systems, actuated surveillance, traffic control etc., where

- the effect of actuation, and/or range of sensing, is spatially limited due to the physical properties of sensors, actuators and their operating environment; for instance, a single light source can only illuminate a confined region, one sprinkler can only irrigate a bounded area of an agricultural field, and a pan-tilt-zoom camera has a limited field-of-view for a given resolution; and,
- actuators are deployed with planned but localized overlaps in their regions of influence to achieve complete geographic coverage.

In general, this procedure is applicable to coupled optimization problems of type (1) where each f^i is a function of a *strict* subset β_i of elements of decision variable x (assuming x is a vector or matrix), and each element $[x]_j$ only appears in a small subset γ_j of component functions.

B. Convergence Analysis

In this section, we start by finding closed forms for the update equations (5) in the proposed FC algorithm. We then show some inequalities that play a central role in the following convergence analysis of the algorithm. In this analysis, we use the definition of projection $\mathcal{P}_\mathcal{X}$ as outlined in [10].

From Equation (10), we can write $\forall i \in [1, N]$ where $j \in \beta_i$ (or equivalently $i \in \gamma_j$),

$$\left[v_k^i\right]_j = \frac{1}{|\gamma_j|} \sum_{l\in\gamma_j} [\hat{I}_{k-1}^l]_j - \frac{\alpha}{|\gamma_j|} \sum_{l\in\gamma_j} [g^l(\hat{I}_{k-1}^l)]_j \tag{11}$$

In the k^{th} iteration, all nodes $l \in \gamma_j$ have already achieved a consensus on $[\hat{I}_{k-1}^l]_j$ in the previous iteration, thus they all possess the same value $[\hat{I}_{k-1}]_j$, therefore

$$\left[v_k^i\right]_j = [\hat{I}_{k-1}]_j - \frac{\alpha}{|\gamma_j|} \sum_{l\in\gamma_j} [g^l(\hat{I}_{k-1})]_j \tag{12}$$

Thus we can rewrite the update equation (5) for FC as

$$\hat{I}_k = \mathcal{P}_\mathcal{X} \left[\hat{I}_{k-1} - T * g(\hat{I}_{k-1}) \right] \tag{13}$$

where entry $[\hat{I}_k]_j$ is obtained from consensus between nodes in γ_j, and T is an $M \times M$ diagonal matrix

$$T = diag\left(\frac{\alpha}{|\gamma_1|}, \frac{\alpha}{|\gamma_2|}, \cdots, \frac{\alpha}{|\gamma_M|} \right) \tag{14}$$

This shows that FC is a variation of the standard subgradient projection method (3)[1]. In the rest of this section, we focus on the light control problem (2) and analyze the convergence of the proposed FC algorithm. We rewrite (2) as follows,

$$\hat{I}^* = \arg\min_{\hat{I}\in\mathcal{X}} f(\hat{I}) \tag{15}$$

where $f(\hat{I}) = ||A * \hat{I} - L||^2$, $A = [a_1, a_2, \cdots, a_N]^T$, and $L = [\phi_i - L_i]$. Since $f(\hat{I})$ is differentiable, the subgradient is unique and equal to the gradient $\nabla f(\hat{I})$. Furthermore, $\nabla f(\hat{I})$ is Lipschitz continuous with constant $R = 2 * \lambda_{max}(A^T A)$, where $\lambda_{max}(A^T A)$ is the maximum eigenvalue of $A^T A$ [11]. This shows that our problem (15) satisfies the standard properties required in the analysis of gradient projection algorithms: $\mathcal{X}$ is closed and convex, f is convex, and $\nabla f(\hat{I})$ is Lipschitz continuous with constant $R > 0$. We introduce the following map, for $I \in \mathcal{X}$, as shown in [12]:

$$h(I) = T^{-1} * (I - \mathcal{P}_\mathcal{X}[I - T * \nabla f(I)]) \tag{16}$$

Now, we can rewrite (13) as follows

$$\hat{I}_k = \hat{I}_{k-1} - Th(\hat{I}_{k-1}) \tag{17}$$

From [11] we can infer that since $\nabla f(\hat{I})$ is Lipschitz continuous with constant R, f will satisfy the following: $\forall \hat{I}, Y \in \mathcal{X}$,

$$f(Y) \leq f(\hat{I}) + \nabla f^T(Y - \hat{I}) + \frac{R}{2}||Y - \hat{I}||^2 \tag{18}$$

Using $Y = \hat{I} - Th$, (16), (17), (18), and some algebra we can get the following upper bound on f: $\forall \hat{I}, Z \in \mathcal{X}$,[2]

$$f(\hat{I} - Th) \leq f(Z) + h^T(\hat{I} - Z) + h^T \left[\frac{R}{2}T^2 - T \right] h \tag{19}$$

Now we will prove a simple Lemma, that is useful in proving the convergence of the FC algorithm.

Lemma 1. For a matrix T of the form of (14), if $0 < T \leq \frac{1}{R}\Lambda$, then $-T \leq \left[\frac{R}{2}T^2 - T\right] \leq -\frac{\alpha}{2*|\gamma|_{max}}\Lambda < 0$. [3]

[1]Note that $\left[T * g(\hat{I}_{k-1})\right]$ is not necessarily a subgradient of f.

[2]We use h and ∇f instead of $h(\hat{I})$ and $\nabla f(\hat{I})$ for the ease of readability.

[3]Λ is an identity matrix of the same size as T.

Proof: $0 < T \leq \frac{1}{R}\Lambda \implies 0 < \frac{R}{2}T^2 \leq \frac{1}{2}T \implies -T < \frac{R}{2}T^2 - T \leq -\frac{1}{2}T \implies -T \leq \frac{R}{2}T^2 - T \leq -\frac{\alpha}{2*|\gamma|_{max}}\Lambda < 0.$

Proof of convergence of the FC algorithm:
For a constant stepsize matix T, using *lemma 1*, and (19) we can see that for $\hat{I} = \hat{I}_k$,

$$f(\hat{I}_k - Th) \leq f(Z) + h^T(\hat{I}_k - Z) + \frac{\alpha}{2 * |\gamma|_{max}}||h||^2 \quad (20)$$

This is of the same form as the projected gradient method in [12], where the stepsize is a constant scalar equal to $\frac{\alpha}{2*|\gamma|_{max}}$. We use the result in [12] to conclude that,

$$f(\hat{I}_k) - f^* \leq \frac{\alpha}{2k|\gamma|_{max}}||\hat{I}_0 - \hat{I}^*||^2 \quad (21)$$

where $\hat{I}_0$ is the initial estimate of the optimizer. Therefore, $\lim_{k\to\infty} f(\hat{I}_k) - f^* = 0.$ $\square$

C. Convergence Rate

From (21), we see that the convergence rate of the FC algorithm is proportional to $\frac{\alpha}{2|\gamma|_{max}}$. We can also conclude that for the global consensus algorithm, the convergence rate is proportional to $\frac{\alpha}{2N}$. Therefore our proposed FC algorithm is faster by a factor of $\frac{N}{|\gamma|_{max}}$.

IV. Cost Analysis

Similar to the analysis in Section II-D, message cost and latency in FC is $\mathcal{O}(\mu_f d_{f,a})$ and $\mathcal{O}(\mu_f|\beta|_m d_{f,m})$ respectively, where (i) $d_{f,a}$ and $d_{f,m}$ are respectively the average and maximum number of messages exchanged by a node to coordinate with its adjacent nodes in $\mathcal{G}_{ie}$, and (ii) $|\beta|_m$ defines the number of bits that need to be exchanged between nodes in terms of maximum number of actuators that affect any sensor in the deployment. This again assumes the existence of a broadcast or multi-cast mechanism wherever required, and ignores routing overhead. A straightforward improvement in FC over GC is in the size of messages that reduces latency by at least a factor of $M/|\beta|_m$.

Besides the communication modality and range that define the topology $\mathcal{G}$ of the network, d_f also depends on the coordination mechanisms provided by the run-time system. In this section, we discuss the effect of two distinct mechanisms provided by our APF on the cost and performance of the proposed FC algorithm. Our analysis and simulations confirm that FC reduces the number of messages and latency over GC by at least an order of magnitude.

A. Flat Networks

The proposed APF can establish communication between adjacent sensor nodes in $\mathcal{G}_{ie}$ by using multi-hop routes involving only intermediate sensors whenever $\mathcal{G}_{ie}$ is distinct from $\mathcal{G}$. In this configuration, $d_{f,m}$ is defined as

$$d_{f,m} = \max_{i\in[1,N]} |d_{ie,i}|, \text{ where } d_{ie,i} = \{l \mid (i,l) \in \mathcal{E}_{ie}\}. \quad (22)$$

In many realistic scenarios, $d_{f,m}$ is typically greater than $d_{c,m}$ because multiple sensors are deployed to measure the effect of each actuator for redundancy as well as complete coverage of all points/regions of interest where a desired *effect* is to be achieved. However, the previous section shows that FC has a faster rate of convergence than GC ($\mu_f < \mu_c$). Our evaluation confirms that this result, combined with the reduction due to smaller message sizes, offsets the effect of larger $d_{f,m}$ to reduce the overall latency. Similar analysis can be made for message cost using $d_{f,a}$.

B. Hierarchical Networks

Hierarchical network configuration in our APF, with each actuator j as a *master* of sensors in its γ_j, provides a more message efficient alternative than flat networks whenever it can be supported by the hardware. In this configuration, the sensor nodes communicate with only their masters to transmit their subgradient updates u_k^i and receive respective consensus values $[\hat{U}_k]_j$ (7) from the actuators in each iteration. As a result, the responsibility of performing the average operation (10) in consensus negotiations is shifted to the actuators, which is natural in FC where the matrices W_j are selected according to the respective masters. Moreover, sensors/controllers anyway need to communicate with the actuators to control them in our target applications, and with mutiple distributed controllers per actuator, it is natural to actuate them using the mean of control inputs desired by each controller.

It is important to note that each sensor i can have multiple masters (β_i) due to actuator overlap and it needs to communicate with all of them in every iteration. Thus, $d_{f,m}$ in this network configuration is reduced to

$$d_{f,m} = |\beta|_m = \max_{i\in[1,N]} |\beta_i|, \quad (23)$$

where $\beta_i = \{j \mid i \in \gamma_j, j \in [1, M]\}$ is the set of actuators that can affect the measurements at sensor i. Assuming that the 1-hop communication range is equal to, if not greater than, the range of actuator influence, we can conclude that $d_{f,m} \approx d_{c,m}$. Similar to the analysis in Section II-D, our evaluation demonstrates that $\mu_c\varphi = \kappa\mu_f$ where $\kappa \geq 10$. As a result, the latency is reduced by a factor of $\mathcal{O}(\kappa M/|\beta|_m)$ over GC. Similar analysis for message cost with $d_{f,a} \approx d_{c,a}$ shows reduction by a factor of $\mathcal{O}(\kappa)$.

V. Application Programming Framework

The network coordination in APF is exposed through a concise *Sync* interface (shown in Figure 1) in addition to a shared memory abstraction presented in [1]. It is supported by three different synchronization modules, one for each version of the distributed subgradient method, that respectively synchronize subgradient and consensus iterations within groups of nodes discovered at run-time. In this section, we describe the implementation of both consensus-based subgradient methods through the example of personalized light control application. The incremental subgradient method is not discussed for brevity, but is supported by a module that implements a reliable token-passing protocol on top of a source routing mechanism. The framework is implemented in TinyOS-2.x.

```
interface Sync {
  command void discover(ntable_t *ntbl, ..);
  event void discovered();
  command void sync(int est_I[]);
  event void synced();
}
```

Fig. 1. Sync interface in APF.

A. Global Consensus and Subgradient Algorithm

In the implementation of GC, all nodes first *discover()* their 1-hop neighbors and assign Metropolis weights to them in a distributed fashion as detailed in [6]. They begin their subgradient update in parallel when all the relevant neighbors have been *discovered()* and their information has been stored in a table (*ntbl*). A call to *sync()* at the end of each subgradient update synchronizes the modified estimates at the node with its coordinating neighbors.

Internally, this synchronization is handled by the module designed for inter-node coordination in *flat* networks. It basically caches estimates received from coordinating neighbors and signals *synced()* event when all the neighbors have reached the same iteration number. In addition, it uses redundant transmissions based on Trickle timers [13] to ensure reliability. This trade-off sacrifices communication energy for a simple, bug-free and manageable implementation with low memory utilization suitable for embedded platforms.

The nodes follow a subgradient update with φ consensus iterations. The sync() function is used to synchronize the consensus iterations as well, and the synced() event is used to trigger the next subgradient update or consensus negotiation. Access to local estimates of neighbors within the consensus operation is supported by a *NeighborTable* interface that provides an iterator over the list of neighbors [7][8].

B. Fast Consensus and Subgradient Method

The encoded FC algorithm is similar to the implementation of GC when the network is *flat* and actuators are *not* involved in the coordination. However, neighbor discovery is no longer limited to nodes 1-hop away; rather, adjacent nodes in $\mathcal{G}_{ie}$ are discovered through network-wide dissemination [14] of actuator identifiers in β_i for each sensor i. It can be easily shown that any pair of nodes, that have overlapping β_i, are adjacent in $\mathcal{G}_{ie}$ as their measurements are affected by at least one common actuator. This process is transparent to the programmer who is only responsible for specifying β_i for each sensor. Moreover, the high messaging cost, incurred by this one-time discovery, is amortized over the lifetime of static deployments where $\mathcal{G}_{ie}$ does not change during execution.

The inbuilt support for *hierarchical* networks enables clear division of subgradient updates and consensus iterations between sensors and actuators respectively. Neighbor discovery is also split between sensors and actuators such that the neighborhood at a sensor i consists of all actuators in β_i, while that at an actuator j consists of all sensors in γ_j. Calculation of W_j in the *flat* network configuration, although a bit more involved for the programmer, is accomplished by accessing disseminated β_i of neighboring sensors through the *NeighborTable* interface. However, this process is simplified

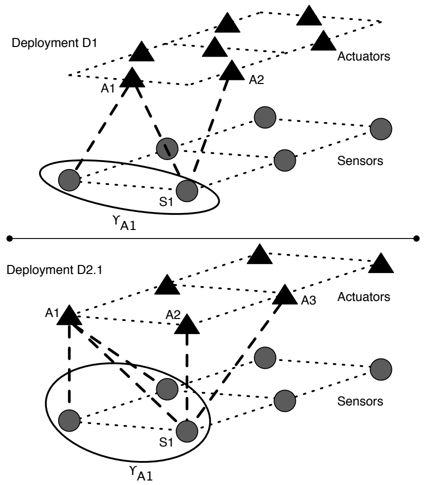

Fig. 2. Types of deployments. The influence sets γ of actuator A1 are marked. For sensor S1, $\beta_{S1} = \{A1, A2\}$ in D1 and $\beta_{S1} = \{A1, A2, A3\}$ in D2.1. Deployment D2.2 (not shown here) is similar to D2.1 but with larger influence of actuators.

in *hierarchical* networks by trivially accessing size of neighborhoods ($|\gamma_j|$) at each actuator.

Although, the synchronization module supporting *hierarchical* networks achieves synchronization of iterations over different coordination neighborhoods for sensors and actuators, it jointly results in the desired execution. Sensors call sync() after completing subgradient updates that results in transferring estimates to the respective actuators. The synced() event is invoked at the actuators when all the repective sensors have finished their subgradient update. Subsequently, actuators perform local consensus trivially by computing the mean of received estimates over their corresponding sensor neighborhoods. Next, a call to sync() at the actuators synchronizes updated estimates with the sensors that begin their next subgradient iteration at the synced() event.

VI. Evaluation: Algorithms

In this section, we compare the three versions of distributed subgradient method implemented in our proposed APF, which enables us to evaluate the impact of network configuration and algorithm parameters on more relevant and practical metrics like convergence latency rather than number of iterations. Our results demostrate that FC outperforms both the algorithms in many realistic scenarios. Moreover, contrary to intuition and prior results, the APF enables us to demonstrate that increasing φ or communication range of sensors in global consensus based approaches worsens performance of the algorithm.

A. Experimental Setup

We simulated two different types of deployments – D1 and D2 – of M light sources and N sensors in TOSSIM to evaluate our TinyOS-2.x implementations. They represent different possible organizations and densities of lights in a typical office space with cubicles as shown in Figure 2. The influence range of light sources is varied in D2 to analyze the effect of varying actuator zones and information exchange topologies ($\mathcal{G}_{ie}$) on the performance of the distributed algorithms. This variance

TABLE I
DEPLOYMENT PARAMETERS, $N = [25, 49, 100]$

Parameters	D1	D2.1	D2.2
$[\|\gamma_j\|_{min}, \|\gamma_j\|_{max}]$	[2,2]	[3,5]	[4,9]
$[\|\beta_i\|_{min}, \|\beta_i\|_{max}]$	[2,4]	[3,5]	[4,9]
$[\|d_{ie}\|_{min}, \|d_{ie}\|_{max}]$	[2,4]	[5,12]	[8,24]
M	40 ($N = 25$) 84 ($N = 49$) 180 ($N = 100$)	$M = N$	$M = N$

captures the effect of not only the fading of light with distance, but obstacles such as cubicle walls as well. The resulting deployements can be ordered in increasing range of actuators (measured by the number of sensors γ_j they affect) as D1, D2.1 and D2.2. The deployment parameters are summarized in Table I.

All sensors in these deployments are configured to be 1-hop away from the respective actuators in β_i that can influence their measurements, assuming that communication range is at least equal to, if not greater than, the actuation range. Communication range between sensors is varied according to the experiments to simulate various densities of communication graph $\mathcal{G}$. A maximum link MTU of 127 bytes for IEEE 802.15.4 is used which roughly allows $p_{link} = 96$ bytes of data payload including all routing and transport headers. In practice, although smaller link packets have significantly higher delivery ratios, we assume reliable delivery with the selected MTU to get optimistic results for both global consensus and incremental method that exchange larger messages than the proposed FC algorithm.

Each sensor is associated with a user-specific light intensity L_i^* as described in Section II. These desired values are *activated* when the respective user enters the office space (detected by some other mechanism), and are otherwise set to zero to conserve electricity. The simulated application tries to achieve incident light intensity at each sensor i that is within 1% of L_i^* on average. Following results assume the worst case situation when all the lights are initially off ($\hat{I}_0 = (0, .., 0)^T$) and all N users enter the office simultaneously. We simulated ten different scenarios, with randomly selected L_i^*, for each deployment type with different number of sensors.

B. Selection of Step-size (α)

Prior research [3][4][5], combined with convergence analysis of FC in Section IV, has shown that under certain realistic assumptions on the subgradients and a constant positive step-size α, the value of the objective function $F(\hat{I}_k)$ in all three versions of subgradient method will converge to within an ϵ ball of the optimal value $F(\hat{I}^*)$ where $\epsilon \propto \alpha$. Thus, selecting a smaller α can potentially improve accuracy of the final result. However, a smaller α also translates to a slower rate of convergence as shown in Section IV and prior work.

In this paper, we are concerned with faster convergence as that ultimately results in lower application response times. Thus, we empirically select the largest values of α for each deployment type and subgradient algorithm that does not result in oscillations and produces fast convergence rates. For a constant $\alpha(= 10)$ across all the algorithms, incremental method has the best convergence rate (measured in number of iterations) followed by FC and global consensus in that order. This confirms the results of convergence analyses in [3][5] and Section IV of this paper. For a local (FC) or global consensus (GC) based approach, a higher α results in faster convergence and is selected according to a scaling factor (greater than one) based on the analysis in Section IV. Our results in subsequent sections demonstrate that all the algorithms are able to achieve desired light intensities at sensors with the selected values of α.

TABLE II
PERFORMANCE: $\mathcal{G} = \mathcal{G}_{ie}$

Deploy.	Incremental		Fast Cons. (*Flat*)		Global Cons. ($\varphi = 1$)	
	μ	Time [s]	μ_f	Time [s]	μ_c	Time [s]
D1	8	27.02	9	2.58	483	1315.38
D2.1	7	13.92	16	8.69	160	924.13
D2.2	15	29.49	19	22.5	213	3067.43

TABLE III
PERFORMANCE: $\mathcal{G} = \mathcal{G}_0 \subset \mathcal{G}_{ie}$

Deploy.	Incremental		Fast Cons. (*Hier.*)		Global Cons. ($\varphi = 1$)	
	μ	Time [s]	μ_f	Time [s]	μ_c	Time [s]
D2.1	7	13.92	16	12.15	475	721.49
D2.2	15	29.49	19	34.58	656	998.91

C. FC: Subgradient Method with Fast Consensus

vs. Global Consensus: Table II summarizes the performance of the three algorithms in terms of the number of subgradient iterations and time required to achieve the desired light intensities. Each element in the table is obtained from an average over 10 scenarios with $N = 49$ where the communication range of sensors is set such that all edges in $\mathcal{G}_{ie}$ also represent 1-hop communication links, *i.e.* $\mathcal{G} = \mathcal{G}_{ie}$. It confirms our analysis in Section IV by demonstrating that FC, implemented on top of a *flat* network, reduces the number of iterations over GC by at least 90% when $\varphi = 1$. Similar results were obtained for higher values of N as well. Section VI-D evaluates the effect of denser $\mathcal{G}$ and higher φ on performance and message cost of GC.

Given $\mathcal{G} = \mathcal{G}_{ie}$ and $\varphi = 1$, it is expected that the reduction in latency due to FC will be proportional to the decrease in number of iterations over GC as the sensors exchange equal number of messages per iteration in both methods. On the contrary, Table II demonstrates that FC reduces latency by at least 99%. This result is an artefact of a smaller number of link packets, bounded by $\mathcal{O}(|\beta_i|^2/p_{link})$, that need to be exchanged per iteration in FC as compared to $\mathcal{O}(|d_{ie}| \cdot M/p_{link})$ packets in GC. Consequently, the average time to complete an iteration is reduced by more than 10x in FC. Nevertheless, the decrease in total number of messages exchanged per node in FC to achieve desired intensities is proportional to the decrease in iterations (Table IV).

However, if we decrease the communication range of sensors such that the resulting communication graph $\mathcal{G}_0$ consists of edges that are a subset of $\mathcal{E}_{ie}$, latency in FC, implemented with *flat* network support, will increase due to communication delays in multi-hop routing that is required to execute the information exchange for fast consensus. Support for *hierarchical* networks in our APF bounds this delay by executing

TABLE IV
TOTAL NUMBER OF MESSAGES EXCHANGED PER NODE

Deployment	Incremental	Fast Consensus		Global Consensus	
		(Flat)	*(Hier.)*	$(\mathcal{G} = \mathcal{G}_{ie})$	$(\mathcal{G} = \mathcal{G}_0)$
D1	16	31	31	1642 $(\mathcal{G}_0 = \mathcal{G}_{ie})$	
D2.1	14	148	71	1465	1424
D2.2	30	308	141	3441	1968

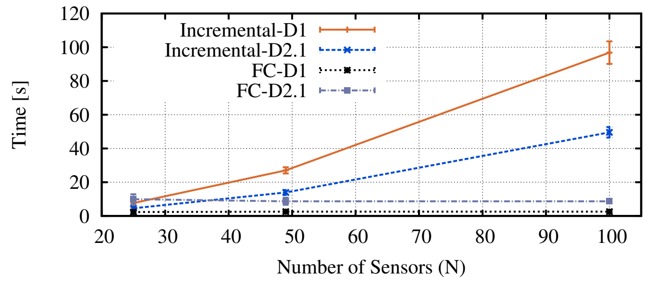

Fig. 3. Comparison of incremental method and FC in terms of convergence latency for $N = [25, 49, 100]$.

consensus negotiations in exactly two hops (given the assumption on direct sensor-actuator communication). Table III shows convergence latency of FC implemented with support for *hierarchical* networks. It is at least 96.5% faster than execution of GC on the same network topology $\mathcal{G}_0$. This is somewhat counter-intuitive as the number of iterations in GC actually increase with decreasing density of communication links in $\mathcal{G}$. Section VI-D discusses this in detail.

vs. Incremental Subgradient Method: Our analysis and evaluation shows that the incremental method requires typically fewer iterations than FC to achieve desired light intensities. However, time to complete each iteration in the incremental method is bounded by $\mathcal{O}(N)$ *subiterations* and $\mathcal{O}(M/p_{link})$ packets exchanged at each hop. Figure 3 confirms this analysis for deployments D1 and D2.1 and varying size of networks. Incremental method in D1 incurs a higher latency than in D2.1 due to a larger number of actuators (M) as shown in Table I. In contrast, latency in FC varies only as a function of $\mathcal{O}(|\beta_i|^2/p_{link})$, and thus, it scales well with increasing size of networks.

Table II shows that FC reduces convergence latency over the incremental method by at least 23.7% (D2.2) and at most 90.4% (D1) for $N = 49$ sensors. Nevertheless, incremental method is approximately 2-10x more efficient than FC in terms of messaging cost ignoring ring topology and routing overheads as shown in Table IV. It is important to note that the incremental method, however, requires a higher number of link packets to transmit each message, while the messages in FC fit in just one link packet. As a result, FC requires 2.6x *fewer* number of packets than the incremental method in deployment D1 despite 2x higher messaging cost, and only 3.4x more packets in D2.1 and D2.2 (down from the 10x higher messaging cost).

D. GC: Subgradient Method with Global Consensus

Researchers in [5] have shown that increasing the number of consensus negotiations (φ) for each subgradient iteration in GC improves the rate of convergence. As a result, without our APF, an application programmer will be compelled to use a higher φ for better performance. Although our evaluation shows that the average reduction in number of iterations is 69.6% when $\varphi = 10$ is used instead of $\varphi = 1$, latency actually increases by 49% to 353% for different deployments with $N = 49$ sensors and $\mathcal{G} = \mathcal{G}_{ie}$. This is because the reduction in iterations is not sufficient to offset the increase in time per iteration that is induced by a higher rate of communication between neighboring sensors for consensus. Figure 4 shows this result for a single simulation scenario with D2.1. Similar results were observed for other values of N and communication topologies $\mathcal{G}$ as well.

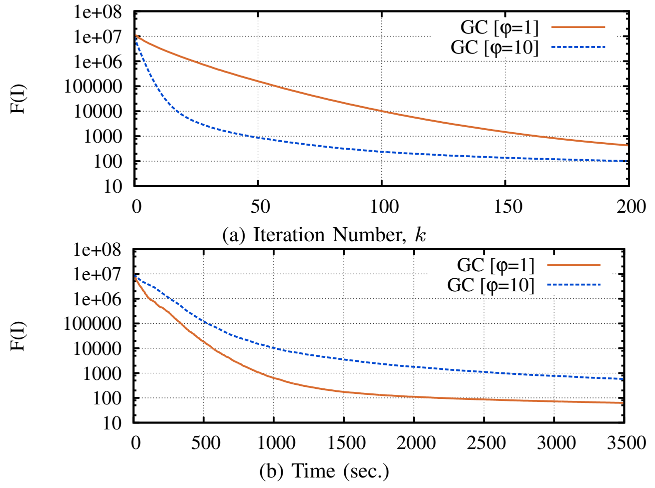

Fig. 4. Impact of φ on performance of GC. It shows evolution of $F(\hat{I})$ with GC for N=49 and D2.1 with $\mathcal{G} = \mathcal{G}_{ie}$ for a single simulation scenario. Figure (b) accounts for 600 iterations with $\varphi = 1$, but only 62 iterations with $\varphi = 10$.

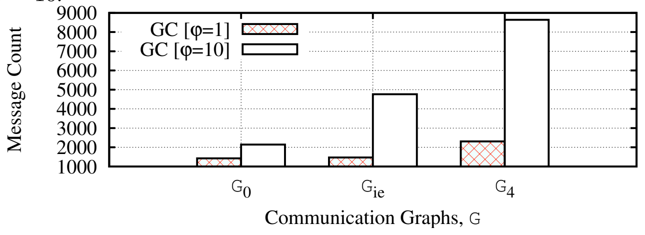

Fig. 5. Impact of φ and communication topologies on message cost of GC. It shows total messages exchanged per node in D2.1 with N=49 and communication graphs with densities increasing as $d(\mathcal{G}_0) < d(\mathcal{G}_{ie}) < d(\mathcal{G}_4)$.

Figure 5 shows that increasing φ has a similar effect on total messages exchanged per node to achieve desired light intensities with any given communication graph $\mathcal{G}$. Each point in the plot is obtained from an average of 10 simulation scenarios. Another similar but interesting trend is observed in the message count for a constant φ with increasing communication densities. It is well known that an increase in network density facilitates faster consensus between the whole network and decreases the number of iterations (μ_c) for a given φ as shown in Figure 6a. However, this reduction is again not sufficient to offset an increase in messaging due to a higher number of coordinating neighbors. The resulting trend is displayed in Figure 5 that shows successively worse message counts for increasing network density.

Figure 6 demonstrates that the increase in messaging due to increasing network density also offsets any reduction in iterations to increase latency as well. However, note that plots

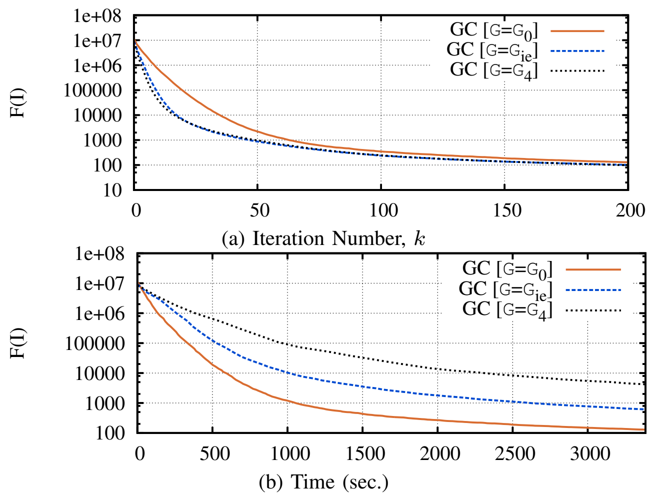

Fig. 6. Impact of communication topologies on performance of GC. It shows the evolution of $F(\hat{I})$ for N=49, $\varphi = 10$ and D2.1 with varying 1-hop communication graph $\mathcal{G} \in \{\mathcal{G}_0, \mathcal{G}_{ie}, \mathcal{G}_4\}$ for a single simulation scenario. Figure (b) accounts for 200, 60 and 23 iterations with $\mathcal{G}_0$, $\mathcal{G}_{ie}$ and $\mathcal{G}_4$ respectively.

for $\mathcal{G}_{ie}$ and $\mathcal{G}_4$ in Figure 6a almost coincide, where $\mathcal{G}_4$ includes additional communication links on top of $\mathcal{G}_{ie}$. This is an important observation that led us to develop the proposed FC algorithm as it shows that each sensor must coordinate with an optimal set of sensors, defined by $\mathcal{G}_{ie}$, to achieve fast convergence. Any communication beyond the information exchange graph, such as in $\mathcal{G}_4$, does not improve performance as the end points of edges *outside* $\mathcal{G}_{ie}$ do not estimate any common control input through the subgradient update step in each iteration.

VII. Related Work

In this paper, we discussed a variety of subgradient methods applied to a class of cooperative and distributed multi-agent optimization problems. They lend themselves to efficient embedded implementations due to relatively simpler comptuations per iteration, and are therefore favored over higher order interior-point methods that converge faster to high accuracy solutions [10]. As a result, they find use in our target class of applications where the goal is to quickly obtain near-optimal approximate solutions within loose tolerance bounds specified by the users.

Prior research on distributed subgradient methods, for solving problems of type (1), can be loosely positioned on two extreme ends of the spectrum based on their assumptions on the structure of the problem. At one end lie all the approaches that propose distributed methods for problems where component functions f^i and constraint set $\mathcal{X}$ can be *separated* into components of x [10]. At the other end are all the *general-purpose* algorithms that solve *non-separable* problems where each f^i depends on all or most of the components of x [5][10][2]. Our proposed algorithm lies somewhere in the middle, where the problem is in general non-separable, but each f^i depends on only a small subset of components of x. Our evaluation shows that although general-pupose methods can be applied to our problem, they are inefficient in many realistic scenarios.

Our proposed APF extends the *neighborhood* abstractions introduced in [7][8] to enable the programmers to define multi-hop neighborhoods in terms of sensor-actuator properties rather than communication range, and synchronize data and iterations across these groups. In contrast with [15] that proposes to explore membership of these groups based on accuracy vs. latency trade-offs in synchronization, we concretely define the optimal set of synchronization neighborhoods at each node based on physical properties of the nodes and the deployment.

Acknowledgment

This material is based upon work supported by the NSF under award #CCF-0820061 and the Center for Embedded Networked Sensing (CENS account #0120778). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF or CENS.

References

[1] R. Balani, K. Lin, L. F. Wanner, J. Friedman, R. K. Gupta, and M. B. Srivastava, “Programming Support for Distributed Optimization and Control in Cyber-Physical Systems,” *ACM/IEEE Second International Conference on Cyber-physical Systems*, April 2011.
[2] N. Z. Shor, K. C. Kiwiel, and A. Ruszcayǹski, *Minimization methods for non-differentiable functions*. Springer-Verlag New York, Inc., 1985.
[3] A. Nedic and D. Bertsekas, “Incremental subgradient methods for nondifferentiable optimization,” *SIAM Journal of Optimization*, vol. 12, no. 1, 2001.
[4] M. Rabbat and R. Nowak, “Distributed optimization in sensor networks,” *IPSN*, 2004.
[5] B. Johansson, T. Keviczky, M. Johansson, and K. Johansson, “Subgradient methods and consensus algorithms for solving convex optimization problems,” *IEEE Conference on Decision and Control*, 2008.
[6] L. Xiao and S. Boyd, “Fast linear iterations for distributed averaging,” *Systems & Control Letters*, Jan 2004.
[7] M. Welsh and G. Mainland, “Programming sensor networks using abstract regions,” *NSDI*, 2004.
[8] K. Whitehouse, C. Sharp, E. Brewer, and D. Culler, “Hood: a neighborhood abstraction for sensor networks,” in *MobiSys*, 2004, pp. 99–110.
[9] Y. Park, J. Shamma, and T. Harmon, “A Receding Horizon Control algorithm for adaptive management of soil moisture and chemical levels during irrigation,” *Environmental Modelling & Software*, vol. 24, no. 9, 2009.
[10] A. Nedic and A. Ozdaglar, *Cooperative Distributed Multi-agent Optimization*, Y. Eldar and D. Palomar, Eds. Cambridge University Press, UK, 2008.
[11] A. Beck and M. Teboulle, “A fast iterative shrinkage-thresholding algorithm for linear inverse problems,” *SIAM Journal of Imgaging Sciences*, vol. 2, March 2009.
[12] L. Vandenberghe, *Optimization Methods for Large-Scale Systems. Course notes for EE236C.* UCLA, Available at http://www.ee.ucla.edu/~vandenbe/ee236c.html, 2009.
[13] P. Levis, N. Patel, D. Culler, and S. Shenker, “Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks,” in *NSDI*, vol. 246, 2004.
[14] K. Lin and P. Levis, “Data Discovery and Dissemination with DIP,” in *IPSN*, 2008.
[15] T. Hnat and K. Whitehouse, “A Relaxed Synchronization Primitive for Macroprogramming Systems,” *INSS*, 2010.