

# On Distributed Optimization using Peer-to-Peer Communications in Wireless Sensor Networks

Björn Johansson, Cesare Maria Carretti, and Mikael Johansson

School of Electrical Engineering  
Royal Institute of Technology (KTH)  
100 44 Stockholm, Sweden

Email: bjorn.johansson@ee.kth.se, carretti@kth.se, mikael.johansson@ee.kth.se

**Abstract**—We describe and evaluate a suite of distributed and computationally efficient algorithms for solving a class of convex optimization problems in wireless sensor networks. The problem class has wide applications in estimation, detection, localization, coordination and resource-sharing. We focus on peer-to-peer algorithms where nodes only exchange data with their immediate neighbors, and consider three distinct alternatives: a dual-based broadcast algorithm, a novel stochastic unicast algorithm, and a linear broadcast algorithm tailored for least-squares problems. We implement the algorithms in the network simulator NS2 and present extensive simulation results for random topologies.

## I. INTRODUCTION

Several important tasks of wireless sensor networks (WSN:s) including estimation, detection, localization, and resource sharing, can be cast as optimization problems [1]. Recently, a large research effort has been devoted to understanding distributed quadratic programming problems, and the properties of the associated distributed averaging or consensus algorithms (see, e.g., [2], [3]). However, moving beyond the least-squares framework offers distinctive advantages in applications [4], [5] and the development of a generic optimization component for wireless sensor networks could potentially have a very broad applicability. Bringing state-of-the art optimization solvers to the resource-constrained sensor networking environment is a grand challenge. Modern high-performance solvers require significant memory and computations and rely on central access to all problem data. In contrast, for sensor network applications one would like to find distributed algorithms which remove the single point of failure of having a central computational node. The resource-constrained environment demands algorithms with small computational and memory requirements. It is desirable that the traffic for coordinating nodes is minimal and that the communication can be done in a way that makes economic use of the wireless medium to reduce energy consumption. Finally, several emerging applications require that nodes do not reveal “private” information when coordinating their decisions with the others.

Focusing on a specific class of convex optimization problems, this paper presents a suite of computationally efficient distributed optimization algorithms with small memory and code size requirements. Our algorithms are all of peer-to-peer nature, and do not rely on any networking functionality apart from nearest neighbor communications. However, the algorithms suggest rather different ways of coordinating

nodes: from passing around a “best estimate” using unicast transmissions to having nodes asynchronously broadcasting their suggestion for the global variables. We implement the most promising algorithms in NS2 and perform extensive and detailed packet-level simulations of algorithm performance in wireless sensor networks equipped with 802.15.4 radios, studying convergence rate and algorithm accuracy.

The paper is organized as follows. We present the general problem formulation in Section 2 and give an overview of possible solution approaches in Section 3. Then, in Section 4, we review relevant aspects of the MAC behavior of 802.15.4. In Section 5, we give detailed descriptions of the candidate optimization algorithms. We describe our detailed packet-level NS2 simulation model in Section 5 and show and discuss the results from extensive simulation studies in Section 6. Finally, we conclude the paper in Section 7.

## II. PROBLEM FORMULATION

We consider convex optimization problems on the form

$$\begin{aligned} & \underset{\theta, \mathbf{x}_1, \dots, \mathbf{x}_N}{\text{minimize}} && \sum_{i=1}^N f_i(\mathbf{x}_i, \theta) \\ & \text{subject to} && \mathbf{x}_i \in \mathcal{X}_i, \quad i = 1, \dots, N \\ & && \theta \in \Theta, \end{aligned} \quad (1)$$

where  $f_i(\mathbf{x}_i, \theta)$  is a cost function associated with node  $i$ ,  $\mathbf{x}_i$  is a vector of “private” variables local to node  $i$ , and  $\theta$  is a global decision variable that all nodes should agree upon. Associated to each vector of private variables  $\mathbf{x}_i$  is a feasible set  $\mathcal{X}_i$ , while the global variable is constrained to belong to the set  $\Theta$ . We assume that  $f_i$  are convex functions and that  $\mathcal{X}_i$  and  $\Theta$  are convex sets with non-empty interior. Furthermore, we let  $f^*$  denote the optimal value of (1), and let  $(\mathbf{x}^*, \theta^*)$  denote the optimizer of (1), with  $\mathbf{x} = (\mathbf{x}_1, \dots, \mathbf{x}_N)$ . Associated to the problem is a communication topology represented by a graph  $\mathcal{G}$  with edges  $\mathcal{E}$  and vertices  $\mathcal{V}$ . The presence of an edge  $(i, j)$  in  $\mathcal{G}$  means that node  $i$  can communicate (i.e., exchange information) directly with node  $j$ . We let  $\mathcal{N}(i) = \{j : (i, j) \in \mathcal{E}\}$  denote the set of neighbors to node  $i$ . The setup is illustrated in Fig. 1. The problem is to choose an optimization algorithm that can be easily implemented in a WSN, preserves energy in the nodes, and does not introduce a lot of communication overhead.

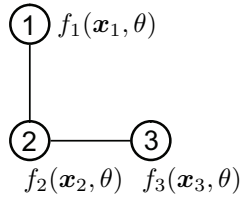


Fig. 1: Example setup with three nodes.

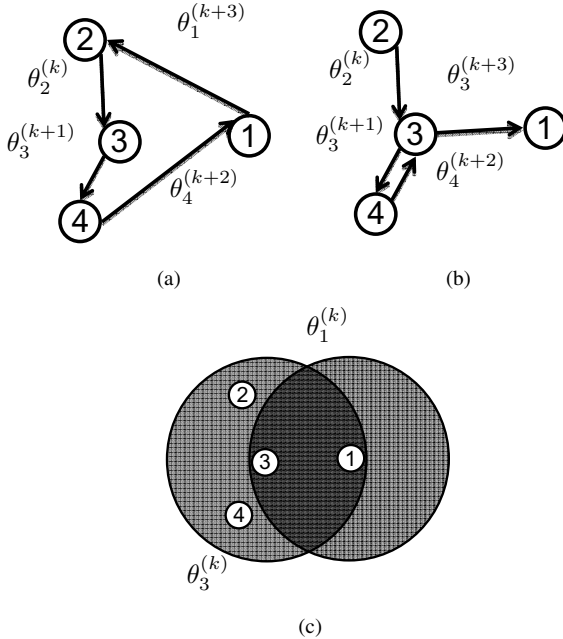


Fig. 2: (a) The ring setup, where information is passed around in a logical ring. (b) The random neighbor setup, where each node unicast information to a random neighbor. (c) The broadcast setup, where each node broadcast information to their neighbors. The broadcast setup and the random neighbor setup are both peer-to-peer.

### III. A TAXONOMY OF SOLUTION APPROACHES

There are many ways of structuring the computations for finding the optimal solution  $(\mathbf{x}^*, \theta^*)$ . The most obvious is a *centralized* approach where nodes pass information about  $(f_i, \mathcal{X}_i)$  to a central node that solves the convex optimization problem (1) using, e.g., the techniques in [1], and then distributes the solution to the nodes. This solution is sensitive to a single point of failure (the central node), requires individual nodes to reveal their cost functions and constraint sets, and demands potentially complex information transfer from nodes to the computational unit. It is often more attractive to use a *decentralized* approach, where nodes collaborate to find the optimal solution to the decision problem [6]. One way to develop such approaches is to rely on decomposition techniques from mathematical programming (see, e.g., [7], [8]). Many decomposition methods exist and they suggest very different ways of structuring the computations and the

communication required for solving the decision problem. One class of methods, based on primal decomposition and incremental subgradient methods [5], [9], result in *estimate passing* solutions: the nodes are organized in a ring, and the current estimate of the optimal solution is passed from one node to the other, see Fig. 2a. When a node receives the current estimate, it performs a local update accounting for its own cost functions and constraint, before passing the modified estimate to the next node. In this approach, nodes do not need to reveal their private objectives and constraints but the method requires reliable communications as the messages conveying the current estimate must not be lost. Other approaches suggest computations based on *peer-to-peer* communication and coordination where nodes exchange information only with neighbors, see Fig. 2b and Fig. 2c. No private information needs to be exchanged, and no network-wide signalling needs to be implemented. For the specific case when the nodes have no internal variables and the cost functions are quadratic,

$$f_i(\mathbf{x}_i, \theta) = \frac{1}{2}(\theta - c_i)^2, \quad i = 1, \dots, N,$$

many special algorithms exist, since the optimal solution is the network-wide average of the constants  $c_i$ . Methods for computing this average go under many names, including distributed averaging, consensus and agreement algorithms [2], [3]. More exotic algorithms for solving this problem have also been suggested, e.g., using nonlinearly coupled dynamical systems [10]. Although the theory for the quadratic case is rather well-developed, we would like to stress that the algorithms do not, in general, apply to our problem formulation (1).

#### A. The Impact of Wireless Communications

In our classification above, communication has been modelled in a very crude and abstract way. However, for wireless devices, the precise details of the communication strategy are critical and can have a large influence on message latency and node energy consumption. In this paper, we focus on tiny wireless sensor nodes and study algorithms that make minimal assumptions on the routing layer, and that operate using the most basic version of contention-based medium access without tight synchronization of node clocks and communication schedules. Specifically, in our peer-to-peer optimization algorithms nodes only communicate with their immediate neighbors and *no multihop routing is required or exploited*. Within this class of solutions, energy efficiency can be achieved by reducing the needed number of iterations for the algorithm to converge. To this end, one can either try to exploit the broadcast nature of the wireless medium (to avoid repeated unicast transmissions with the same information) or limit the number of retransmissions due to collisions. This is precisely the strategies of the algorithms that we will evaluate and compare in this paper.

#### IV. MAC BEHAVIOR IN 802.15.4

The target application is sensor networks, and the dominating standard for such networks is the IEEE standard 802.15.4 [11]. This protocol has support for several modes

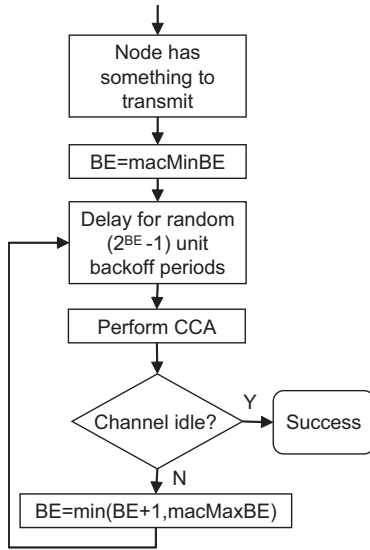


Fig. 3: Flowchart showing the MAC scheme, with the following abbreviations: Clear Channel Assessment (CCA) and Backoff Exponent (BE).

of operation. Since the algorithms we consider are based on peer-to-peer communication, we will focus on 802.15.4 in peer-to-peer operation. In this mode of operation, 802.15.4 uses unslotted CSMA-CA, and the behavior of the MAC for unslotted CSMA-CA is illustrated in a flowchart in Fig. 3.

Some of the optimization algorithms are synchronized and require that all communication activities have been completed in order to proceed to the next step in the algorithm. Thus, we need reliable communications, and we can address this by using acknowledgements (ACKs). There is an ACK function in 802.15.4, but we will complement this with some additional functionality at the application layer, see Section VI-A.

## V. ALGORITHMS

### A. Assumptions

We make the following technical assumptions.

*Assumption 1:* For all algorithms, (i) the nodes form a connected network and (ii) the functions  $f_i$  are convex and the sets  $\mathcal{X}_i$  and  $\Theta$  are convex with non-empty interior. Moreover, (iii) strong duality holds for (3); (iv) the subgradients  $\mathbf{h}(\boldsymbol{\lambda})$  of  $d(\boldsymbol{\lambda})$  defined in (5) are bounded,  $\mathbf{h}(\boldsymbol{\lambda}) \leq C$ , for all values of  $\boldsymbol{\lambda}$ ; and (v) the subgradients,  $g_i(\theta)$  of  $q_i(\theta)$  defined in (10) are bounded,  $g_i(\theta) \leq C$ , for all values of  $\theta$  in the set  $\Theta$ .

One way to check that (iii) is fulfilled is the so called Slater's constraint qualification (see, e.g., [1]). Moreover, the simplest way of fulfilling (iv) and (v) is to require that the sets  $\Theta$  and  $\mathcal{X}_i$  are bounded and convex.

### B. Dual Decomposition Based Algorithms

Dual based methods introduces one "local" copy of  $\theta$  in each node, and updates these estimates so that they converge towards the optimum. Formally, this is done by introducing

multiple copies of the global variables, require that these should be equal, and then relaxing these constraints, see [12].

We rewrite the problem as

$$\begin{aligned} & \underset{(\mathbf{x}_1, \dots, \mathbf{x}_N), (\theta_1, \dots, \theta_N)}{\text{minimize}} && \sum_{i=1}^N f_i(\mathbf{x}_i, \theta_i) \\ & \text{subject to} && \mathbf{x}_i \in \mathcal{X}_i, \quad i = 1, \dots, N \\ & && \theta_i \in \Theta, \quad i = 1, \dots, N \\ & && \theta_i = \theta_j, \quad j \in \mathcal{N}(i). \end{aligned} \quad (2)$$

If the communication graph is connected, then the formulation (2) is equivalent to (1). In general, all these constraints are not necessary: a necessary and sufficient condition for equivalence between the two formulations is that equalities are introduced for the smallest set of links that keep the nodes connected (compare with [13]). We can rewrite the problem into a more compact form

$$\begin{aligned} & \underset{\mathbf{x}, \boldsymbol{\theta}}{\text{minimize}} && \sum_{i=1}^N f_i(\mathbf{x}_i, \theta_i) \\ & \text{subject to} && \mathbf{x}_i \in \mathcal{X}_i, \quad i = 1, \dots, N \\ & && \theta_i \in \Theta, \quad i = 1, \dots, N \\ & && \mathbf{A}\boldsymbol{\theta} = \mathbf{0}, \end{aligned} \quad (3)$$

where  $\boldsymbol{\theta} = (\theta_1 \dots \theta_N)^\top$ ,  $\mathbf{x} = (\mathbf{x}_1 \dots \mathbf{x}_N)^\top$ , and  $\mathbf{A} \in \mathbb{R}^{E \times N}$ . Each row in the matrix  $\mathbf{A}$  corresponds to a link in  $\mathcal{G}$ , and the elements of  $\mathbf{A}$  are defined as

$$A_{li} = \begin{cases} 1, & \text{if link } l \text{ connects node } i \text{ with node } j, i > j \\ -1, & \text{if link } l \text{ connects node } i \text{ with node } j, i < j \\ 0, & \text{otherwise,} \end{cases} \quad (4)$$

(in the literature, this type of matrix is sometimes denoted the directed incidence matrix). We can of course also write the minimum needed number of equalities in this way.

In this formulation, the computations are complicated by the constraint  $\mathbf{A}\boldsymbol{\theta} = \mathbf{0}$ , which couples the local estimates of the network-wide decision variables across nodes. To distribute the computations to the nodes, we apply dual decomposition to the formulation (3). Formally, we introduce the Lagrangian

$$L(\boldsymbol{\lambda}, \mathbf{x}, \boldsymbol{\theta}) = \sum_{i=1}^N f_i(\theta_i, \mathbf{x}_i) + \boldsymbol{\lambda}^\top \mathbf{A}\boldsymbol{\theta}$$

and the dual function

$$\begin{aligned} d(\boldsymbol{\lambda}) = & \min_{\mathbf{x}, \boldsymbol{\theta}} L(\boldsymbol{\lambda}, \mathbf{x}, \boldsymbol{\theta}) \\ & \text{subject to} \quad \mathbf{x}_i \in \mathcal{X}_i, \quad i = 1, \dots, N \\ & \quad \quad \quad \theta_i \in \Theta, \quad i = 1, \dots, N, \end{aligned} \quad (5)$$

where the elements in  $\boldsymbol{\lambda}$  are called Lagrange multipliers. The Lagrange multipliers can be interpreted as "prices" for disagreeing with neighbors on the best global variable value.

The dual function can be written as  $d(\boldsymbol{\lambda}) = \sum_{i=1}^N d_i(\boldsymbol{\lambda})$  with

$$\begin{aligned} d_i(\boldsymbol{\lambda}) = & \min_{\mathbf{x}_i, \theta_i} f_i(\theta_i, \mathbf{x}_i) + \theta_i [\mathbf{A}^\top \boldsymbol{\lambda}]_i \\ & \text{subject to} \quad \mathbf{x}_i \in \mathcal{X}_i \\ & \quad \quad \quad \theta_i \in \Theta, \end{aligned} \quad (6)$$

where  $[\mathbf{y}]_j$  is the  $j$ :th element of the vector  $\mathbf{y}$ . Note that the dual function is separable, i.e., for fixed  $\boldsymbol{\lambda}$  the nodes can

individually decide the optimal values of its local variables  $(\mathbf{x}_i, \theta_i)$  by evaluating  $d_i(\boldsymbol{\lambda})$ . When strong duality holds [1], the optimal value of (1) is the same as the optimal value of the dual problem

$$\underset{\boldsymbol{\lambda}}{\text{maximize}} \quad \sum_{i=1}^N d_i(\boldsymbol{\lambda}). \quad (7)$$

However, when the cost functions are not strictly convex, there may be problems recovering the primal optimal  $(\mathbf{x}^*, \boldsymbol{\theta}^*)$ .

1) *Subgradient Algorithm*: The dual function is not differentiable in general, and the simplest way to perform the maximization in (7), is to use a subgradient algorithm. Thus, we need a subgradient, and a subgradient,  $\mathbf{h}$ , of  $d(\cdot)$  at  $\boldsymbol{\lambda}$  is given by

$$\mathbf{h} = \mathbf{A}\boldsymbol{\theta}^*(\boldsymbol{\lambda}), \quad (8)$$

where  $\boldsymbol{\theta}^*$  is the  $\boldsymbol{\theta}$  that minimizes the Lagrangian for the given  $\boldsymbol{\lambda}$  (see, e.g., [9, Chapter 8] for details). Now we have all the pieces needed for a decentralized algorithm that solves (3), and the result can be seen in Algorithm 1.

---

**Algorithm 1** Dual Based Subgradient Algorithm (DBSA) .

---

1: Let  $k = 0$  and  $\boldsymbol{\lambda}^{(0)} = 0$ .

2: **loop**

3: Each node  $i$  solves

$$\begin{aligned} & \underset{\mathbf{x}_i, \theta_i}{\text{minimize}} \quad f_i(\theta_i, \mathbf{x}_i) + \theta_i \left( \sum_{\{l | A_{li} \neq 0\}} A_{li} \lambda_l^{(k)} \right) \\ & \text{subject to} \quad \mathbf{x}_i \in \mathcal{X}_i, \theta_i \in \Theta \end{aligned}$$

to get  $\theta_i^{(k)}$ .

4: Each node  $i$  transmits its optimal  $\theta_i^{(k)}$  to its neighbors.

5: Each node  $i$  updates the elements of  $\boldsymbol{\lambda}^{(k)}$  that are part of their Lagrangian

$$\lambda_l^{(k+1)} = \lambda_l^{(k)} + \alpha \sum_{\{j | A_{lj} \neq 0\}} A_{lj} \theta_j^{(k)}, \quad l \in \{m | A_{mi} \neq 0\}.$$

6:  $k = k + 1$ .

7: **end loop**

---

Under Assumption 1, the best iteration of Algorithm 1 can be shown to converge to a ball around the optimal value  $f^*$  [9]. There are also convergence results for this algorithm when there are packet losses and the cost function is quadratic, see [12].

As we previously pointed out, the dual function is in general not differentiable due to the lack of strict convexity. This means that it can be troublesome to find the optimal  $(\mathbf{x}^*, \boldsymbol{\theta}^*)$  and it also hinders us from using other methods than subgradient based. One remedy is to add a quadratic term to the Lagrangian, and we get the so called *Augmented Lagrangian*

$$L_c(\boldsymbol{\lambda}, \mathbf{x}, \boldsymbol{\theta}) = \sum_{i=1}^N f_i(\theta_i, \mathbf{x}_i) + \boldsymbol{\lambda}^\top \mathbf{A}\boldsymbol{\theta} - \frac{c}{2} \|\mathbf{A}\boldsymbol{\theta}\|_2^2.$$

Using the Augmented Lagrangian, we can use the method of multipliers or the alternating direction multiplier method [6].

The most interesting theoretical property of these algorithms is that they converge for all constant step-sizes. However, in practice, tuning is still crucial to give acceptable performance, and we will not investigate these variants in this paper.

### C. Primal Decomposition Based Algorithms

Instead of relying on Lagrange multipliers to enforce all nodes to have a common  $\theta$ , we can use primal decomposition. This type of decomposition is based on the fact that if the resources are fixed, the optimization problem decouples. In this case, we fix the parameter  $\theta$  and we can rewrite (1) as

$$\begin{aligned} & \underset{\theta}{\text{minimize}} \quad \sum_{i=1}^N q_i(\theta) \\ & \text{subject to} \quad \theta \in \Theta, \end{aligned} \quad (9)$$

where

$$q_i(\theta) = \min_{\mathbf{x}_i \in \mathcal{X}_i} f_i(\mathbf{x}_i, \theta) \quad (10)$$

and  $q(\theta) = \sum_{i=1}^N q_i(\theta)$ . The functions  $q_i(\cdot)$  are not coupled, and, in general, not differentiable. Therefore, we have to resort to using subgradient based algorithms. Let each (convex) component  $q_i(\cdot)$  have a subgradient  $g_i(\theta)$  at  $\theta$  and we assume that they are bounded according to Assumption 1.

1) *Incremental Subgradient Algorithm*: The simplest algorithm that uses the primal formulation (9) and that is suitable for distributed implementation is the *incremental subgradient algorithm*. The basic version is based on estimate passing in a logical ring [5], but it does not fulfill our assumptions of a peer-to-peer algorithm. Nevertheless, based on its prototypical update,

$$\theta_{k+1} = \mathcal{P}_\Theta \{ \theta_k - \alpha g_{k \bmod N+1}(\theta_k) \},$$

where the estimate  $\theta_{k+1}$  is sent to node  $k \bmod N + 1$  in the ring, we can devise peer-to-peer algorithms.

We now consider a novel extension, introduced and further explained in [14], [15], of the randomized version of the incremental subgradient method. The update equation of the estimate is

$$\theta_{k+1} = \mathcal{P}_\Theta \{ \theta_k - \alpha g_{w_k}(\theta_k) \}, \quad (11)$$

where  $\mathcal{P}_\Theta\{\cdot\}$  denotes projection on the set  $\Theta$  and  $\alpha > 0$  is a fixed stepsize. In the original setup, the random variables  $w_k$  are independent and identically distributed, and takes on values from the set  $\{1, \dots, N\}$  with equal probability. This means that the estimate is passed around between randomly chosen nodes in the network, and successive estimates could be updated by nodes possibly several hops away. Thus, it is not a peer-to-peer algorithm. However, this can be remedied by sending the estimate to a neighboring node instead. In the extension,  $w_k$  is the state of a special Markov chain that takes on values in the finite set  $\{1, 2, \dots, N\}$  with transition matrix  $\mathbf{P}$ . The speciality of the Markov chain lies in the fact that its transition matrix respects the communication topology of the graph  $\mathcal{G}$ . It turns out that there is a simple way to find a Markov chain fulfilling these conditions, namely the so called

---

**Algorithm 2** Markov Randomized Incremental Subgradient Algorithm (MRISA) .
 

---

- 1: Initialize  $\theta_0$  and  $\alpha$ . Set  $k := 0$  and  $w_k := 1$ .
  - 2: **loop**
  - 3: At node  $w_k$ , compute a subgradient,  $g_{w_k}$ , for  $q_{w_k}(\theta_k)$ .
  - 4:  $\theta_{k+1} := \mathcal{P}_\Theta\{\theta_k - \alpha g_{w_k}\}$ .
  - 5: Send  $\theta_{k+1}$  to a random neighbor,  $w_{k+1}$ , with transition probability according to  $\mathbf{P}$ .
  - 6:  $k := k + 1$ .
  - 7: **end loop**
- 

Metropolis-Hastings scheme [16]. If the underlying communication topology is connected, then all necessary assumptions on the transition matrix are fulfilled if the elements are set to

$$P_{ij} = \begin{cases} \min\{\frac{1}{e_i}, \frac{1}{e_j}\} & \text{if } (i, j) \in \mathcal{E} \text{ and } i \neq j \\ \sum_{(i,k) \in \mathcal{E}} \max\{0, \frac{1}{e_i} - \frac{1}{e_k}\} & \text{if } i = j \\ 0 & \text{otherwise,} \end{cases} \quad (12)$$

where  $e_i$  is node  $i$ 's number of edges. The algorithm is summarized in Algorithm 2, and we have the following convergence result (see [15] for details): With probability 1,

- a) The sequence  $\{\theta_k\}_{k=0}^\infty$  fulfill

$$\begin{cases} \inf_{k \geq 0} q(\theta_k) = f^*, & \text{if } f^* = -\infty \\ \inf_{k \geq 0} q(\theta_k) \leq f^* + \frac{\alpha C^2 K}{2}, & \text{if } f^* > -\infty. \end{cases}$$

- b) If the set of optimal  $\theta$ ,  $\Theta^* = \{\theta \in \Theta | q(\theta) = f^*\}$ , is nonempty, then the sequence  $\{\theta_k\}_{k=0}^\infty$  fulfill

$$\min_{0 \leq k \leq \tau} q(\theta_k) \leq f^* + \frac{\alpha C^2 K}{2} + \delta,$$

where  $\tau$  is a (stochastic) stopping time with bounded expected value:

$$\mathbb{E}[\tau] \leq \frac{N}{2\alpha\delta} \left( \frac{\text{dist}(\theta_0)}{\Theta^*} \right)^2,$$

with  $\text{dist}_{\Theta^*}(\theta_0) = \inf\{\|\theta_0 - y\|_2 | y \in \Theta^*\}$ .

#### D. Linear Iterations for the Quadratic Case

For the special case with quadratic cost functions

$$f_i(\mathbf{x}_i, \theta) = \frac{1}{2}(\theta - c_i)^2, \quad i = 1, \dots, N, \quad (13)$$

the optimal solution is the average,  $\theta^* = \sum_{i=1}^N c_i / N$ , and it is possible to devise simple tailored algorithms. If we desire a linear iteration, then the simplest algorithm is on the form

$$\theta^{(k+1)} = \mathbf{W}\theta^{(k)},$$

where  $\mathbf{W}$  is a weighting matrix. Necessary and sufficient conditions on  $\mathbf{W}$  to guarantee convergence to the average are  $\mathbf{1}^\top \mathbf{W} = \mathbf{1}^\top$ ,  $\mathbf{W}\mathbf{1} = \mathbf{1}$ , and  $\rho(\mathbf{W} - \mathbf{1}\mathbf{1}^\top/N) < 1$ , where  $\mathbf{1}$  is the  $N$ -dimensional vector with all elements equal to 1 and  $\rho(\cdot)$  denotes the spectral radius [3]. A simple and practical way to fulfill these conditions are by taking  $\mathbf{W} = \mathbf{P}$  with  $\mathbf{P}$  from

---

**Algorithm 3** Linear Iterations Algorithm (LIA) .
 

---

- 1: Initialize  $\theta_i^{(0)} := c_i$  for all  $i = 1, \dots, N$ . Set  $k := 0$ .
  - 2: **loop**
  - 3: Each node  $i$  broadcasts  $\theta_i^{(k)}$  to all of its neighbors.
  - 4: Each node  $i$  updates its  $\theta$  according to  $\theta_i^{(k+1)} := W_{ii}\theta_i^{(k)} + \sum_{j \in \mathcal{N}(i)} W_{ij}\theta_j^{(k)}$ .
  - 5:  $k := k + 1$ .
  - 6: **end loop**
- 

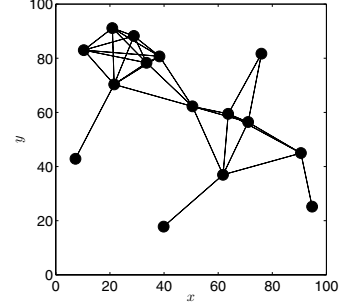


Fig. 4: The topology for one of the random networks with 15 nodes used in the Monte Carlo simulations. The filled circles denote nodes and a line between two nodes indicate that they can transmit to each other.

(12). It is also possible to find a  $\mathbf{W}$ , using convex optimization, that boosts the convergence rate, see [3] for details.

The resulting algorithm is summarized in Algorithm 3, and if the weighting matrix is chosen according to (12), Algorithm 3 will converge to the optimal value.

## VI. SIMULATIONS

We have concentrated our efforts on evaluating three algorithms: DBSA (Algorithm 1), MRISA (Algorithm 2), and LIA (Algorithm 3). There are, as mentioned before, an abundance of algorithms, and we have chosen these three due to their interesting properties and their simplicity. We consider the standard example of least-squares estimation: the nodes should minimize the deviation between the estimate and the measurements in the quadratic sense, see (13).

We now proceed with describing the NS2 simulation setup and implementation details of the algorithms.

### A. NS2 Implementation

NS2 is an event driven network simulator, and we decided to use it due to its extensibility and possibility to set many physical parameters. We have modified the simulation environment, such that the behavior of the nodes at application layer is consistent with the optimization algorithms described in this paper, giving us the possibility to easily change the key parameters of each algorithm through high level scripting (using the OTcl interface). We ran our application over the 802.15.4 module for NS2 [17], and the setup is summarized in Table I. The 802.15.4 module was set to peer-to-peer asynchronous mode, without any beacon. Furthermore, the two

TABLE I: NS2 simulation setup.

Physical and MAC layer	802.15.4
Transmitting Power	0.282 W
Antenna	Omnidirectional at 1.4 m
Frequency	2.4 GHz
Propagation Model	Two-ray ground
Queue	Droptail FIFO

ray ground propagation mode considers both the direct path and the ground reflect path, without any shadowing or fading. For further information about the simulations see [18]; the code is publicly and freely available upon request from the authors.

### B. DBSA

Due to the dual based subgradient algorithm's resilience to packet losses [12], we use unreliable broadcast. This means that each node broadcast its estimate to its neighbors but does *not* wait for ACKs. However, each node will wait for a specific *waiting time* to receive estimates from its neighbors. During this waiting time, each node broadcasts its estimate  $\zeta$  times to the neighboring nodes, to increase the probability of successful transmissions. The waiting time and the number of retransmissions during the waiting time are tunable parameters. Finally, the DBSA also has a tunable stepsize parameter,  $\alpha$ . With a fixed stepsize, complete convergence is not guaranteed, and we have to be content with convergence to a ball around the optimal solution.

### C. MRISA

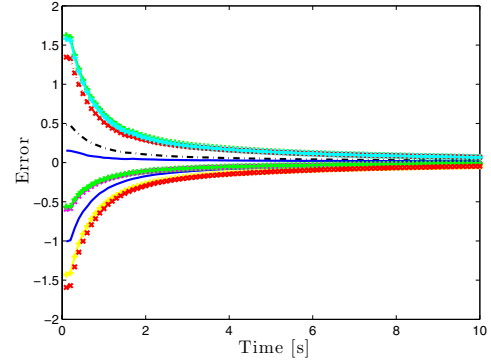
The MRISA has one tunable parameter, namely the stepsize  $\alpha$ . It is also possible to tune the Markov chain probabilities,  $P$ , to increase the exploration rate of the network, but we choose to stick to the simple scheme given by (12).

### D. LIA

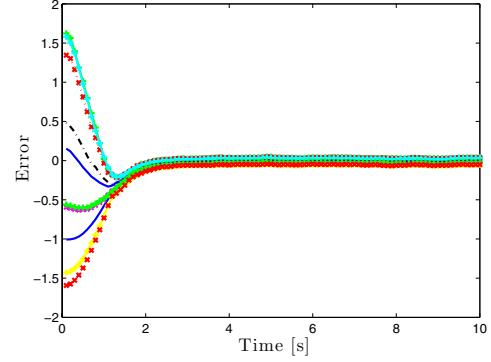
For the linear iterations algorithm, we use reliable broadcast. This means that each node broadcast its current estimate to its neighbors, and wait for ACKs. If the node does not get an ACK within a fixed time-out window, it broadcasts its estimate again with a list of nodes from which it did not get any ACKs. The time-out window length is a tunable parameter and it is reasonable to let it depend on the number of nodes in the network (whenever this is known).

### E. Performance Metrics

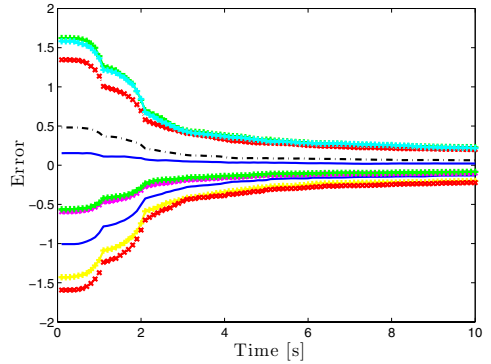
Depending on the precise configuration, current WSN nodes (e.g., Moteiv Tmote Sky) have higher power consumption when receiving than when transmitting. However, in general, transmitting can be much more expensive than receiving. Since the nodes are active all the time, due to the peer-to-peer mode, the relevant metric for nodes with higher power consumption while receiving than transmitting is the rate of convergence in terms of time, and this is the metric we use in this paper. However, in a setup where transmission is much more expensive than reception, then the relevant metric could be convergence time in terms of number of packets sent.



(a)



(b)



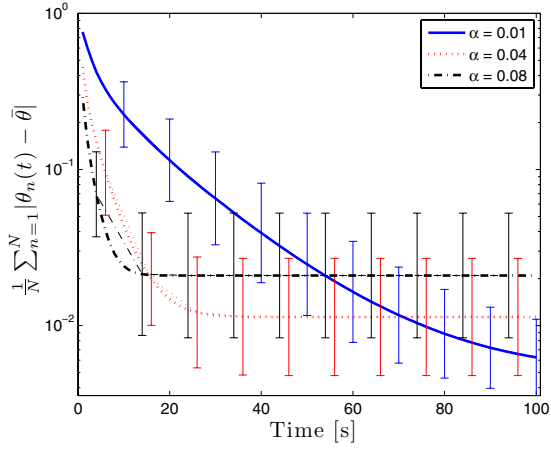
(c)

Fig. 5: These plots show the average of 1000 Monte Carlo simulations of the individual node error for 10 node random topologies. (a) DBSA, with  $\alpha = 0.08$ . (b) MRISA, with  $\alpha = 0.01$  (c) LIA.

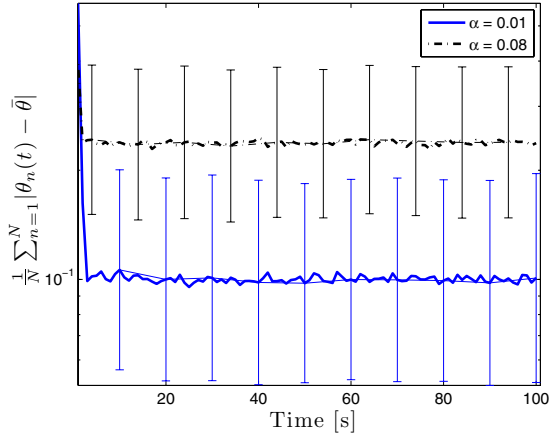
### F. Simulation Results

For DBSA, we set the waiting time to 0.1 seconds and the number of retransmissions,  $\zeta$ , to 3. These parameter were found by direct experimentation. Furthermore, we set the stepsize  $\alpha$  to 0.01, 0.04, and 0.08. For MRISA, we set the stepsize  $\alpha$  to 0.01 and 0.08. For LIA, we set the time-out window to  $1 + N/20$  seconds, which is a value that performed well in simulations.

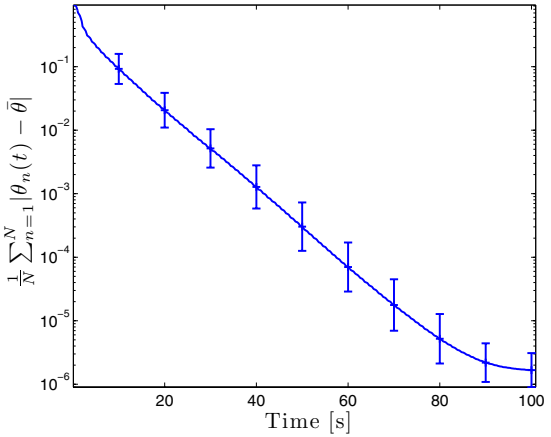
Random network topologies are used to evaluate typical performance of the algorithms, and we performed 1000 Monte



(a)

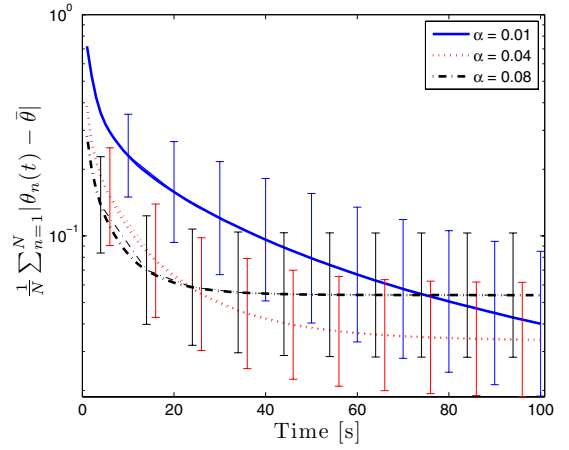


(b)

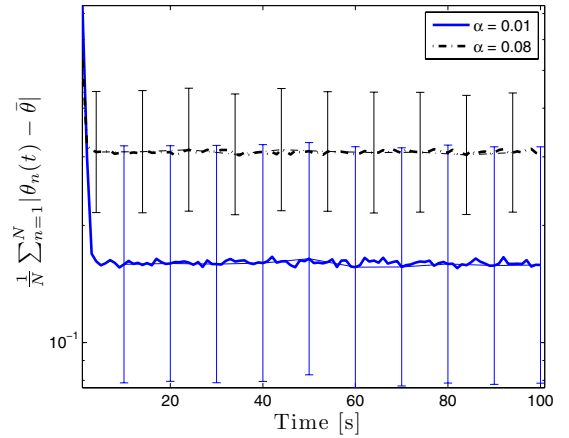


(c)

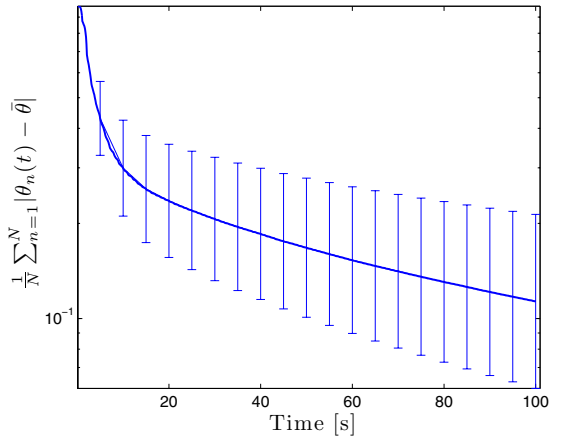
Fig. 6: These plots show the average of 1000 Monte Carlo simulations of the average of the absolute error for all nodes versus time for the 5 node random topology, see Fig. 4 for an example of a typical random network. The thick lines are the averages and the error bars show the standard deviation<sup>1</sup>. The optimal  $\theta$  is  $\theta^* = \bar{\theta} = 2.04$ . (a) DBSA. The algorithm was run with 3 different  $\alpha$  values according to the legend. (b) MRISA. The algorithm was run with 2 different  $\alpha$  values according to the legend. (c) LIA.



(a)



(b)



(c)

Fig. 7: These plots show the average of 1000 Monte Carlo simulations of the average of the absolute error for all nodes versus time for the 15 node random topology, see Fig. 4. The thick lines are the averages and the error bars show the standard deviation<sup>1</sup>. The optimal  $\theta$  is  $\theta^* = \bar{\theta} = 1.94$ . (a) DBSA. The algorithm was run with 3 different  $\alpha$  values according to the legend. (b) MRISA. The algorithm was run with 2 different  $\alpha$  values according to the legend. (c) LIA.

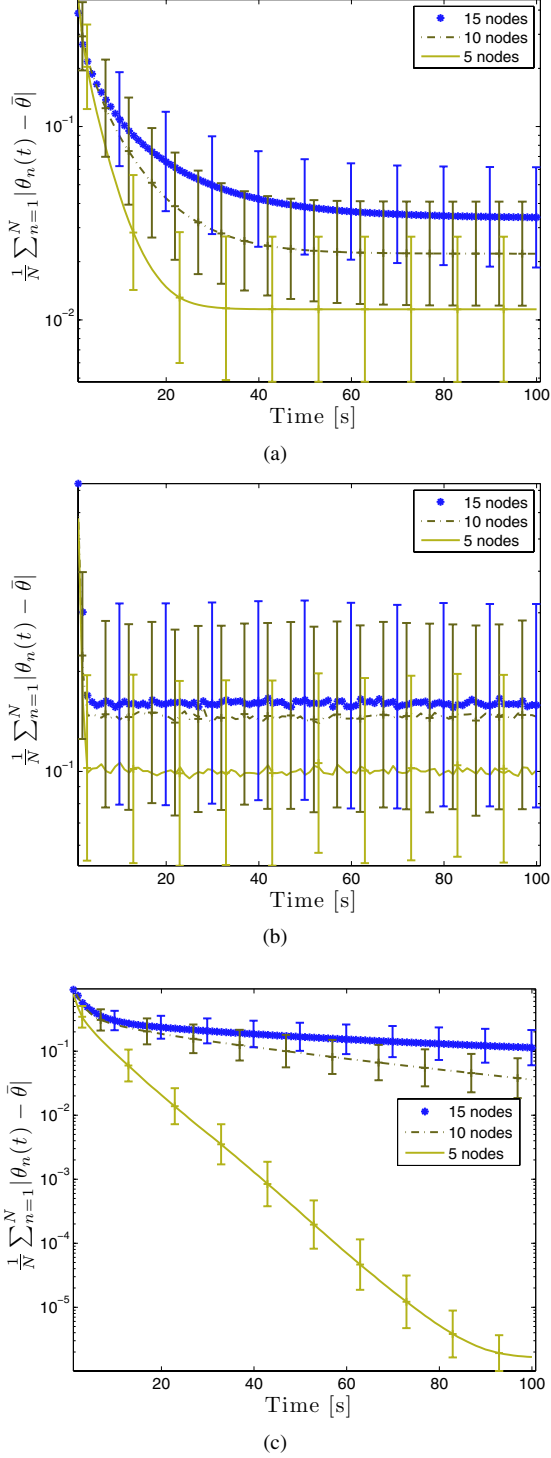


Fig. 8: These plots show the average of 1000 Monte Carlo simulations of the average of the absolute error for all nodes versus time for random topologies with 5, 10, and 15 nodes. The thick lines are the averages and the error bars show the standard deviation<sup>1</sup>. (a) DBSA. The algorithm was run with  $\alpha = 0.04$ . (b) MRISA. The algorithm was run with  $\alpha = 0.01$ . (c) LIA.

Carlo simulations for each setup. We start with 10 node networks with random positions. The number of nodes is low enough to admit a detailed inspection of the algorithm behavior, but still high enough in order for the algorithms to have interesting behavior. The individual node errors are shown in Fig. 5. We can see that MRISA has the fastest convergence to stationarity, but with higher bias than LIA (no bias) and DBSA.

Next, we simulate 15 node networks with random positions, see Fig. 4. The average absolute errors are shown in Fig. 7. Again, we can see that MRISA has the fastest convergence to stationarity but has larger bias than LIA (no bias) and DBSA.

Finally, we compare the three algorithms on random networks with 5, 10, and 15 nodes, using the best parameters found (DBSA:  $\alpha = 0.04$ . MRISA:  $\alpha = 0.01$ ). The plots indicate that LIA has much slower convergence when the number of nodes is increased, and the plots indicate that LIA does not scale well with the number of nodes. DBSA is less affected by the number of nodes, but the convergence time is slower and the bias is increased when the number of nodes increased. MRISA is least affected by the number of nodes, and the convergence behavior is almost the same when the number of nodes is increased. However, the bias will also increase.

## VII. CONCLUSIONS

In this paper, we have considered peer-to-peer algorithms for distributed optimization in wireless sensor networks. We have evaluated the performance of three specific algorithms on network nodes equipped with 802.15.4 compliant radios. While the linear iterations algorithm is tailored for averaging (least-squares) problems, where it excels in networks with few nodes, it is not applicable to general optimization problems. The other two algorithms are complementary and work surprisingly well in our simulations.

The simulations indicate that broadcast algorithms, where each node needs data from all of its neighbors to complete one iteration, do not scale with the number of nodes. One way of mitigating this is to divide the network into clusters and run the broadcast algorithm within each cluster, see, e.g., [19].

An interesting observation is that it is a design choice to put the stochastic behavior of the resulting system (optimization algorithm plus communication mechanism) either in the algorithm or in the communication mechanism. If we use a stochastic algorithm, in which the updates are explicitly stochastic, then we can use an unreliable communication mechanism with known time delay and number of required packets. On the other hand, if we use a deterministic algorithm, then we need a reliable communication mechanism, in which the time delay and number of packets needed are uncertain.

<sup>1</sup>If the error bars are plotted using  $\bar{x} \pm \sigma$ , i.e., average  $\pm$  standard deviation, then the lower value,  $\bar{x} - \sigma$ , may end up below zero. Negative values will ruin the plot due to the log scale. To get a more visually pleasing plot, the upper value of the error bar is  $u = \bar{x} + \sigma$ , while the lower value of the error bar is  $l = \frac{\bar{x}}{1 + \frac{\sigma}{\bar{x}}}$ , and the following holds  $\bar{x} = \sqrt{l \cdot u}$ .

Future work includes evaluating alternative algorithms, and investigating the impact of tuning parameters in greater detail. We are also planning to implement the algorithms on wireless sensor nodes and run large-scale experiments in our testbed.

#### REFERENCES

- [1] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.
- [2] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.
- [3] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems and Control Letters*, vol. 53, pp. 65–78, 2004.
- [4] B. Johansson, A. Speranzon, M. Johansson, and K. Johansson, "On decentralized negotiation of optimal consensus," *Automatica*, vol. 44, pp. 1175–1179, 2008.
- [5] M. Rabbat and R. Nowak, "Distributed optimization in sensor networks," in *IPSN*, 2004.
- [6] D. Bertsekas and J. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, 1997.
- [7] B. Johansson, P. Soldati, and M. Johansson, "Mathematical decomposition techniques for distributed cross-layer optimization of data networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, pp. 1535–1547, 2006.
- [8] D. Palomar and M. Chiang, "A tutorial on decomposition methods for network utility maximization," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, pp. 1439–1451, 2006.
- [9] D. Bertsekas, A. Nedić, and A. Ozdaglar, *Convex Analysis and Optimization*. Athena Scientific, 2003.
- [10] S. Barbarossa and G. Scutari, "Decentralized maximum likelihood estimation for sensor networks composed of nonlinearly coupled dynamical systems," *IEEE Transactions on Signal Processing*, vol. 55, pp. 3456–3470, 2007.
- [11] "IEEE Standard 802.15.4," September 2006.
- [12] M. G. Rabbat, R. D. Nowak, and J. A. Bucklew, "Generalized consensus computation in networked systems with erasure links," in *IEEE SPAWC*, 2005.
- [13] I. Schizas, A. Ribeiro, and B. Giannakis, "Consensus-based distributed parameter estimation in ad hoc wireless sensor networks with noisy links," in *IEEE ICASSP*, 2007.
- [14] B. Johansson, M. Rabi, and M. Johansson, "A simple peer-to-peer algorithm for distributed optimization in sensor networks," in *IEEE CDC*, 2007.
- [15] —, "A randomized incremental subgradient method for distributed optimization in networked systems," *SIAM Journal on Optimization*, 2008, submitted.
- [16] S. Boyd, P. Diaconis, and L. Xiao, "Fastest mixing markov chain on a graph," *SIAM Review*, vol. 46, pp. 667–689, 2004.
- [17] J. Zheng and M. J. Lee, "A comprehensive performance study of IEEE 802.15.4," *Sensor Network Operations*, *IEEE press*, vol. Wiley Interscience, chapter 4, pp. 218–237, 2006.
- [18] C. M. Carretti, "Comparison of distributed optimization algorithms in sensor networks," Master's thesis, Royal Institute of Technology (KTH), 2008.
- [19] S.-H. Son, M. Chiang, S. Kulkarni, and S. Schwartz, "The value of clustering in distributed estimation for sensor networks," in *IEEE Wirelsscom*, 2005.