Energy Optimized Topologies for Distributed Averaging in Wireless Sensor Networks*

Ioannis Ch. Paschalidis,† *Senior Member, IEEE,* and Binbin Li,‡ *Student Member, IEEE*

***Abstract*—We study the energy efficient implementation of averaging/consensus algorithms in wireless sensor networks. For static, time-invariant topologies we start from the recent result that a bidirectional spanning tree is preferable in terms of convergence time. We formulate the combinatorial optimization problem of selecting such a minimal energy tree as a mixed integer linear programming problem. Since the problem is NP-complete we devise a semi-definite relaxation and establish bounds on the optimal cost. We also develop a series of graph-based algorithms that yield energy efficient bidirectional spanning trees and establish associated bounds on the optimal cost. For dynamic, time-varying topologies we consider a recently proposed load-balancing algorithm which has preferable convergence time properties. We formulate the problem of selecting a minimal energy interconnected network over which we can run the algorithm as a sequential decision problem and cast it into a dynamic programming framework. We first consider the scenario of a large enough time horizon and show that the problem is equivalent to constructing a Minimum Spanning Tree. We then consider the scenario of a limited time horizon and employ a "rollout" heuristic that leverages the spanning tree solution and yields efficient solutions for the original problem. Some of our algorithms can be run in a distributed manner and numerical results establish that they produce near-optimal solutions.**

I. Introduction

WIRELESS Sensor NETworks (WSNETs) have gained in popularity due to the ease of their deployment in a variety of environments. Growing applications include building/industrial automation, surveillance, and environmental/wildlife monitoring. Due to limited battery power, nodes rely on short-range communications and form a multi-hop network to exchange information among them. Energy consumption is a key issue in WSNETs, since it directly impacts their lifespan. Energy in WSNETs nodes is consumed by the CPU, by the sensing subsystem, and by the radio, with the latter consuming the most [2]. Therefore, network topologies that are energy efficient are always preferred and have attracted significant attention recently.

A common function with many uses performed by a WSNET is the computation of an average value over the values "sensed" by individual nodes. A centralized process according to which a single node (a gateway) collects all measurements and computes the average requires too much data to be transported over a potentially large number of hops and is therefore energy wasteful and impractical. Moreover, such a process has to be repeated frequently often as the measurements, corresponding to an underlying physical system, change over time. Alternatively, a distributed computation of the average has the advantage of being fault-tolerant (the network operation is not dependent on a small set of sensor nodes) and self-organizing (network functionality does not require constant supervision) [3], [4].

Distributed averaging has been studied extensively as a problem of distributed *consensus* in multi-agent decision and control. Consensus in a general setting, including dynamic connectivity and communication delays, has been considered in [5], [6], [7], [8], [9]. When the connectivity of the network does not change over time (*static networks*) [10] shows that a so-called *bidirectional spanning tree* (a spanning tree in which all links are bidirectional) achieves consensus in time that has the same polynomial rate of growth as a lower bound established for all topologies. When the connectivity can change (*dynamic networks*) the network must become connected infinitely often, e.g., once every B time units for some large enough B, in order to achieve consensus. [10] shows that an asynchronous load-balancing algorithm from [11] can guarantee a convergence time which is polynomial in the number of nodes and B.

Motivated by these results, we consider the problem of building energy efficient WSNET topologies over which we can implement these distributed averaging algorithms. Due to the broadcasting nature of the wireless channel, when a node transmits data to another node, nearby nodes can listen to the transmission and receive the data as well. The energy cost of the transmission is proportional to the power level of the transmitting node, the larger this is the more nodes it can reach.

Our contribution concerns both static and dynamic networks. For static networks we formulate the combinatorial optimization problem of selecting the power levels of the nodes to support a bidirectional spanning tree. The problem has been shown to be NP-complete. We provide a *Mixed Integer Linear Programming (MILP)* formulation which is tractable only for relatively small instances. We also develop a *Semi-Definite Programming (SDP)* relaxation and a rounding scheme that can generate MILP feasible solutions. Further, we develop a series of graph-based algorithms that yield energy efficient bidirectional spanning trees and lower and upper bounds on the optimal energy cost. Two of the algorithms we propose can be run in a distributed manner, a feature that is particularly appealing for large WSNETs. An illustrative set of numerical results demonstrates that our graph-based

* Research partially supported by the NSF under grant EFRI-0735974, by the DOE under grant DE-FG52-06NA27490, by the ARO under grant W911NF-09-1-0492, and by the ODDR&E MURI10 program under grant N00014-10-1-0952. Preliminary results for the case of static networks were presented in [1].

† I. Ch. Paschalidis is with the Department of Electrical and Computer Engineering and the Division of Systems Engineering, Boston University, Boston, MA 02215, USA, e-mail: yannisp@bu.edu, url: http://ionia.bu.edu/.

‡ B. Li is with the Division of Systems Engineering, Boston University, Boston, MA 02215, USA, email: libinbin@bu.edu.

algorithms obtain solutions that are quite close to optimal.

To model dynamic networks we assume that communication between two nodes with sufficient power to talk to each other is intermittent (affected by the "state" of the channel). Hence, any packet generated by one of the nodes is received by the other node with a certain "success" probability. Our goal at any given point of time is to specify a pair of nodes that should attempt to communicate so as to guarantee that the network becomes connected once every B time units while it remains energy-efficient. We formulate this problem as a sequential decision problem and cast it into a *Dynamic Programming (DP)* framework. In particular, we provide a finite horizon formulation with a horizon of length B and a large terminal cost corresponding to the case that connectivity is not achieved at the end of the horizon. First we consider the regime where B is large enough and the terminal cost is never paid. We establish some structural properties of the optimal policy and show that it corresponds to the construction of a *Minimum Spanning Tree (MST)*. The MST problem can be solved in a distributed manner using an algorithm from [12].We then attempt to tackle the case where the horizon B is finite and a terminal cost can not be ignored. The corresponding DP problem is difficult to solve. We resort to heuristics, in particular, we propose a rollout algorithm ([13]) that leverages the MST solution.

The rest of this paper is organized as follows. In Sec. II we establish our notation and review results on averaging/consensus algorithms. Sec. III presents our results on static networks. Sec. IV collects our results on dynamic networks. Conclusions are in Sec. V.

Notational Conventions: Throughout the paper all vectors are assumed to be column vectors. We use lower case boldface letters to denote vectors and for economy of space we write $\mathbf{x} = (x_1, \ldots, x_N)$ for the column vector $\mathbf{x}$. $\mathbf{x}'$ denotes the transpose of $\mathbf{x}$, $||\mathbf{x}||$ its Euclidean norm, and $\mathbf{0}$, $\mathbf{1}$ the vector of all zeros and ones, respectively. We use upper case boldface letters to denote matrices and $\mathbf{A} = (a_{ij})_{i,j=1}^N$ indicates the matrix $\mathbf{A}$ with i, j element a_{ij}. We write $\mathbf{I}$ for the identity matrix and $\mathbf{0}$ for the matrix of all zeros. We use $diag(\mathbf{x})$ to denote a diagonal matrix with the elements of the vector $\mathbf{x}$ in the main diagonal and zeros elsewhere. Similarly, $diag(\mathbf{A}_1, \ldots, \mathbf{A}_n)$ denotes the block diagonal matrix with matrices $\mathbf{A}_1, \ldots, \mathbf{A}_n$ in the main diagonal. Finally, for any set $\mathcal{X}$, $|\mathcal{X}|$ denotes its cardinality.

II. Consensus and Averaging

A. The agreement algorithm

We start by describing the agreement algorithm run by a set $\mathcal{N} = \{1, 2, \ldots, N\}$ of nodes. The values stored at the nodes at time t are denoted by the vector $\mathbf{x}(t) = (x_1(t), \ldots, x_N(t))$. The nodes update their values as

$$x_i(t+1) = \sum_{j=1}^{N} a_{ij}(t) x_j(t), \qquad i = 1, \ldots, N, \tag{1}$$

or, in matrix form $\mathbf{x}(t+1) = \mathbf{A}(t)\mathbf{x}(t)$, where $\mathbf{A}(t) = (a_{ij}(t))_{i,j=1}^N$ is a nonnegative stochastic matrix. In words, every node forms a convex combination of its value with the values of the nodes it receives messages from, assuming that $a_{ij}(t) > 0$ if and only if node j sends messages to i at time t. As is common with consensus algorithms we make the following assumption.

Assumption A
There exists a positive constant α such that: (i) $a_{ii}(t) \geq \alpha$, for all i, t; (ii) $a_{ij}(t) \in \{0\} \cup [\alpha, 1]$, for all i, j; and (iii) $\sum_{j=1}^N a_{ij}(t) = 1$, for all i.

The communication pattern between the nodes at time t can also be represented by a directed graph $\mathcal{G}(t) = (\mathcal{N}, \mathcal{E}(t))$, where $\mathcal{E}(t)$ is the arc set and $(j, i) \in \mathcal{E}(t)$ if $a_{ij}(t) > 0$.

Our next two assumptions require that following an arbitrary time t, and for any i, j, there is a sequence of communications through which node i will influence (directly or indirectly) the value held by node j.

Assumption B
For every $t \geq 0$, the graph $\mathcal{G} = (\mathcal{N}, \cup_{s \geq t} \mathcal{E}(s))$ is strongly connected.

Assumption C (Bounded Interconnectivity Times)
There is some B such that for all k, the graph $(\mathcal{N}, \mathcal{E}(kB) \cup \mathcal{E}(kB+1) \cup \cdots \cup \mathcal{E}((k+1)B))$ is strongly connected.

For the agreement algorithm, the following result is from [6] and [14].

Theorem II.1 *Under Assumptions A and C, the agreement algorithm guarantees asymptotic consensus, that is, there exists some c, which depends on $\mathbf{x}(0)$ and the sequence of graphs $G(\cdot)$, such that $\lim_{t \to \infty} x_i(t) = c$, for all i.*

B. Static topologies and convergence results

Consider now the static connectivity case [5], i.e., $\mathbf{A}(t) = \mathbf{A}$ for all t. Notice $\mathbf{A}$ can be thought as the transition probability matrix of an (irreducible and aperiodic) Markov chain, thus, $\mathbf{A}^t$ converges to a matrix whose rows are equal to $\pi = (\pi_1, \ldots, \pi_n)$ – the steady-state probability vector. Interesting special cases of the consensus algorithm include: (i) the so called *bidirectional* or *symmetric* case where communication between nodes is always bidirectional, i.e., $(i, j) \in \mathcal{E}(t)$ if and only if $(j, i) \in \mathcal{E}(t)$, and (ii) the *equal-neighbor* case where the a_{ij}'s are equal for all $j \in \{i\} \cup \mathcal{N}_i(t)$ where $\mathcal{N}_i(t) = \{j | j \neq i, (j, i) \in \mathcal{E}(t)\}$.

We are interested in using the agreement algorithm to average the initial values $x_i(0)$, $i = 1, \ldots, n$, of the nodes. To that end, we can scale the initial values by replacing $x_i(0)$ with $x_i(0)/(N\pi_i)$ in which case the consensus algorithm converges to the average $\lim_{t \to \infty} x_i(t) = \sum_{i=1}^N \pi_i x_i(0)/(N\pi_i) = (1/N) \sum_{i=1}^N x_i(0)$. We will refer to this as the *scaled averaging algorithm* and it requires a centralized computation of the steady-state probabilities which needs global connectivity information (the matrix $\mathbf{A}$). An approach that uses only local information and no centralized computation is provided in [10] for the bidirectional, equal-neighbor case and employs two parallel runs of the consensus algorithm. The corresponding algorithm, given below, starts with initial values $\mathbf{x}(0)$ and maintains values $\mathbf{y}(t) = (y_1(t), \ldots, y_N(t))$, $\mathbf{z}(t) = (z_1(t), \ldots, z_N(t))$, and $\mathbf{x}(t)$ at each time t.

Algorithm 1

- *Initialize:* $y_i(0) = 1/|\mathcal{N}_i(t)|$ *and* $z_i(0) = x_i(0)/|\mathcal{N}_i(t)|$ *for all* i.
- *The nodes update* $\mathbf{y}(t), \mathbf{z}(t)$ *using the consensus algorithm as:* $\mathbf{y}(t+1) = \mathbf{A}\mathbf{y}(t)$ *and* $\mathbf{z}(t+1) = \mathbf{A}\mathbf{z}(t)$.
- *Each node* i *sets:* $x_i(t) = z_i(t)/y_i(t)$.

Noting that in the bidirectional equal-neighbor case the transition probabilities are given by $\pi_i = |\mathcal{N}_i(t)|/(\sum_{i=1}^N |\mathcal{N}_i(t)|)$, it can be easily verified that $\lim_{t\to\infty} x_i(t) = (1/N)\sum_{i=1}^N x_i(0)$ as desired.

To characterize the rate of convergence and the convergence time, let $\lambda_1(=1) \geq \lambda_2 \geq \ldots \geq \lambda_N$ be the eigenvalues of $\mathbf{A}$ and define $\mathbf{x}^* = \lim_{t\to\infty} \mathbf{A}^t\mathbf{x}(0)$, $\mathcal{X} = \{\mathbf{x}^* | \forall \mathbf{x}(0)\}$. The *convergence rate* of the consensus algorithm on a graph with connectivity matrix $\mathbf{A}$ is defined

$$\rho = \sup_{\mathbf{x}(0)\notin\mathcal{X}} \lim_{t\to\infty} \left(\frac{\|\mathbf{x}(t) - \mathbf{x}^*\|}{\|\mathbf{x}(0) - \mathbf{x}^*\|}\right)^{\frac{1}{t}} = \max\{|\lambda_2|, |\lambda_N|\}, \quad (2)$$

and the *convergence time* $T_N(\epsilon)$ is defined as

$$T_N(\epsilon) = \min\left\{\tau \,\middle|\, \frac{\|\mathbf{x}(t) - \mathbf{x}^*\|_\infty}{\|\mathbf{x}(0) - \mathbf{x}^*\|_\infty} \leq \epsilon, \forall t \geq \tau, \forall \mathbf{x}(0) \notin \mathcal{X}\right\}. \quad (3)$$

[10] shows that there exist graphs for which the convergence time $T_N(\epsilon)$ is $\Omega(N^2 \log(1/\epsilon))$. Moreover, in the *bidirectional equal-neighbor* case one can achieve the same polynomial growth of the convergence time if enough arcs are deleted so that the consensus algorithm runs on a so called *bidirectional spanning tree*. Specifically, a directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ is said to be a *bidirectional spanning tree* if (a) it is symmetric, i.e., if arc $(i,j) \in \mathcal{E}$ then $(j,i) \in \mathcal{E}$; (b) it contains all self-arcs (i,i); and (c) if we ignore the orientation of the arcs and delete self-arcs and any duplicate arcs between any two nodes then we obtain a spanning tree. For such bidirectional spanning trees, [10] shows that $\rho \leq 1 - 1/(3N^2)$ and $T_N(\epsilon) = O(N^2 \log(N/\epsilon))$. It follows that Algorithm 1 has $O(N^2 \log(N/\epsilon))$ convergence time when run on a bidirectional spanning tree. In fact, if the nodes know the total number of nodes N, Algorithm 1 is not needed and the scaled averaging algorithm can reach the average using local information only.

C. Dynamic topologies and convergence results

Next consider the case where communications are still bidirectional but the topology changes dynamically ([15], [16]). [10] shows that the convergence time of the general agreement algorithm is not polynomially bounded, even though it is an open question whether this is the case when restricting to symmetric graphs. When focusing on the equal-neighbor version of the agreement algorithm, [17] gives an example that has exponential convergence time. In particular, if B (cf. Assumption C) is proportional to N, the convergence time increases faster than an exponential in N.

For the symmetric dynamic network satisfying Assumption C, [10] proposes a variation of an old *load-balancing* algorithm from [11] to tackle the problem. We describe the steps that each node carries out at each time t in Algorithm 2. We denote the node executing the steps below as node i. Some steps refer to the neighbors of node i, which means nodes other than i that are its neighbors at the time the corresponding step is being executed.

Algorithm 2
For a node i *in the network, if* $\mathcal{N}_i(t)$ *is empty, node* i *does nothing at time* t*; else, node* i *carries out the following steps.*

- *Node* i *broadcasts its current value* x_i *to all of its neighboring nodes (every* k *with* $k \in \mathcal{N}_i(t)$*).*
- *Node* i *finds a neighboring node* j *with the smallest value:* $x_j = \min\{x_k \mid k \in \mathcal{N}_i(t)\}$. *If* $x_i \leq x_j$*, then node* i *does nothing further at this step. If* $x_j < x_i$*, then node* i *makes an offer of* $(x_i - x_j)/2$ *to* j.
- *If Node* i *does not receive any offers, it does nothing further at this step. Otherwise, it sends an acceptance to the sender of the largest offer and a rejection to all the other senders. It updates the value of* x_i *by adding the value of the accepted offer.*
- *If an acceptance arrives for the offer made by node* i*, node* i *updates* x_i *by subtracting the value of the offer.*

To characterize the rate of convergence and the convergence time, [10] introduces the following "Lyapunov" function to quantify the distance of the state $\mathbf{x}(t)$ from the desired limit:

$$V(t) = \|\mathbf{x}(t) - \frac{1}{N}\textstyle\sum_{i=1}^N x_i(0)\mathbf{1}\|^2. \quad (4)$$

It is shown that $V(t)$ is a monotonically non-increasing function of t when performing Algorithm 2. Given a sequence of graphs $G(t)$ on N nodes, and an initial vector $x(0)$, the convergence time $T_{G(\cdot)}(x(0), \epsilon)$ is defined as:

$$T_{G(\cdot)}(x(0), \epsilon) = \min\{t | V(\tau) \leq \epsilon V(0), \forall \tau \geq t\}. \quad (5)$$

The (worst case) convergence time, $T_N(B, \epsilon)$, is defined as the maximum value of $T_{G(\cdot)}(x(0), \epsilon)$, over all initial conditions $x(0)$, and all graph sequences $G(\cdot)$ on $\mathcal{N}$ that satisfy Assumption C for that particular B. [10] shows that there exists a constant $c > 0$ such that for every N and $\epsilon > 0$, $V((k+1)B) \leq (1 - 1/(2N^3))V(kB)$, i.e., $T_N(B, \epsilon) \leq cBN^3 \log \frac{1}{\epsilon}$.

III. Energy Efficient Topology for Averaging – Static Networks

A. Problem formulation

Consider a set of nodes $\mathcal{N} = \{1, 2, \ldots, N\}$. Each node i has some power limit P_i^{max} $(i = 1, \ldots, N)$ which determines its potential neighboring nodes. We let P_{ij} denote the minimum transmit power needed by node i to reach node j. Let $\mathcal{A}$ denote the set of *potential bidirectional* links between the nodes, that is, $\mathcal{A} = \{(i,j) \mid i < j,\ P_i^{max} \geq P_{ij},\ P_j^{max} \geq P_{ji}\}$. As defined, $\mathcal{A}$ contains undirected links; replacing each link (i,j) in $\mathcal{A}$ with the two corresponding directed links (i,j) and (j,i) we form an arc set $\mathcal{A}_c$. Consistent with Assumption B we assume that the graph $(\mathcal{N}, \mathcal{A}_c)$ is strongly connected.

Transmissions by a node i to node j can be heard without any additional energy expense by any other node k with $P_{ik} \leq P_{ij}$. The objective is to minimize the total transmit energy consumption over all nodes in the network while maintaining a bidirectional spanning tree. To that end, maintaining a connection between nodes i and j requires that i uses a transmit power no less than P_{ij} and j uses a transmit power no less than P_{ji}. We note that nodes consume energy when they receive as well, but this is no significantly different from the energy used while listening. In the model we consider, nodes are listening all the time so there is no need to include the corresponding energy consumption in our minimization. To reduce the latter energy, one could allow the nodes to "sleep"; see related work in [18]. We also remark that the model we introduced can accommodate any physical layer model that prescribes the transmit power needed to reach node j from i. As an example, we can use $P_{ij} = d_{ij}^{\alpha_c}$, where d_{ij} is the Euclidean distance between nodes i and j, and α_c is the channel loss exponent.

1) Mixed integer linear programming model: Our formulation is similar to [19] which proposed an MILP for minimum power multicasting in wireless networks with sectored antennas.

Let $P_i \in [0, P_i^{max}]$ be the transmit power of node i. Let F_{ij} (respectively, F_{ji}), $(i,j) \in \mathcal{A}$, represent the flow from node i to node j (respectively, from node j to i). We adopt indicator variables $X_{ij} \in \{0,1\}$, $(i,j) \in \mathcal{A}$, to show whether there is flow on link $(i,j) \in \mathcal{A}$. For those $(i,j) \notin \mathcal{A}$ we make the convention $X_{ij} = 0$; while for those $(i,j) \in \mathcal{A}$, if $F_{ij} > 0$ or $F_{ji} > 0$ then $X_{ij} = 1$.

The bidirectional topology optimization problem can be cast as a single-origin multiple-destination uncapacitated flow problem with integer constraints indicating the selection of bidirectional links. Any node can be selected as the origin of the flow; we select node 1. The corresponding flow problem involves routing $N-1$ units of supply from node 1 (which has no demand) to all other nodes each of which has one unit of demand and zero units of supply. The MILP formulation is in Fig. 1.

$$
\begin{aligned}
& C^{MILP} = \min \textstyle\sum_{i=1}^{N} Y_i && (6)\\
& s.t. \textstyle\sum_{\{j|(1,j)\in\mathcal{E}\}} F_{1j} = N-1, && (7)\\
& \textstyle\sum_{\{j|(1,j)\in\mathcal{E}\}} F_{j1} = 0, && (8)\\
& \sum_{\substack{\{j|(i,j)\in\mathcal{E}\\ \text{or } (j,i)\in\mathcal{E}\}}} F_{ji} - \sum_{\substack{\{j|(i,j)\in\mathcal{E}\\ \text{or } (j,i)\in\mathcal{E}\}}} F_{ij} = 1, \ \forall i \in \mathcal{N} \setminus \{1\}, && (9)\\
& (N-1)\cdot X_{ij} - F_{ij} \geq 0, \quad \forall (i,j) \in \mathcal{E}, && (10)\\
& (N-1)\cdot X_{ij} - F_{ji} \geq 0, \quad \forall (i,j) \in \mathcal{E}, && (11)\\
& Y_i - X_{ij}P_{ij} \geq 0, \quad \forall (i,j) \in \mathcal{E}, && (12)\\
& Y_j - X_{ij}P_{ji} \geq 0, \quad \forall (i,j) \in \mathcal{E}, && (13)\\
& \textstyle\sum_{(i,j)\in\mathcal{E}} X_{ij} = N-1, && (14)\\
& X_{ij} \in \{0,1\}, \quad \forall (i,j) \in \mathcal{E}, && (15)\\
& F_{ij}, F_{ji} \geq 0, \quad \forall (i,j) \in \mathcal{E}, && (16)\\
& Y_i \geq 0, \quad \forall i. && (17)
\end{aligned}
$$

Fig. 1. The MILP formulation of the minimal energy bidirectional spanning tree construction.

Some explanations are in order. The objective function (6) is equal to the total energy consumption by all nodes. Constraints (7)–(9) maintain flow conservation. Constraints (10)–(11) reflect our conventions for the binary variables X_{ij} setting them to one if there is flow on the link $(i,j) \in \mathcal{A}$ in either direction. Constraint (14) guarantees that no redundant links are selected and the resulting graph is a spanning tree. Finally, constraints (12)–(13) ensure that each selected link from $\mathcal{A}$ will be bidirectional and the appropriate level of transmit power will be accounted for at both nodes incident to the link. The number of integer variables X_{ij} is $|\mathcal{A}|$ and the number of continuous variables (F_{ij}'s and P_i's) is equal to $2|\mathcal{A}| + N$.

The problem considered has been shown to be NP-complete in [20], which has led to heuristics. For instance, [19] used an incremental cost mechanism to select communication links to form a spanning tree. [21] proposed a procedure to incrementally increase transmit powers until a spanning tree is formed. A similar approach is employed in [22] for minimum energy broadcasting with symmetric communication links.

B. An algorithm based on a semi-definite relaxation

In this section we develop an SDP relaxation of the MILP and use it to bound the objective as well as obtain a feasible solution. SDP minimizes a linear function subject to a linear matrix inequality. SDP problems are convex and can be solved in polynomial-time by interior point methods using available solvers (e.g., [23]).

We start with some background and notation. For symmetric matrices $\mathbf{M}$ and $\mathbf{Z}$ in $\mathbb{R}^{m\times m}$ their *Frobenius* inner product is given by

$$\mathbf{M} \bullet \mathbf{Z} \triangleq \sum_{i=1}^{m}\sum_{j=1}^{m} M_{ij}Z_{ij} = Tr(\mathbf{MZ}),$$

where $Tr(\cdot)$ denotes the trace of a matrix. Given symmetric matrices $\mathbf{M}_i \in \mathbb{R}^{m\times m}$, $i = 0,\dots,n$, an SDP (in its dual form) is a problem of the following form with decision variables the elements of the symmetric matrix $\mathbf{Z}$:

$$
\begin{aligned}
\max \quad & \mathbf{M}_0 \bullet \mathbf{Z}\\
\text{s.t.} \quad & \mathbf{M}_i \bullet \mathbf{Z} = c_i, \quad i = 1,\dots,n,\\
& \mathbf{Z} \succeq 0,
\end{aligned}
$$

where "$\succeq 0$" denotes positive semi-definiteness.

The integrality constraint (15) can be rewritten as

$$X_{ij}^2 - X_{ij} = 0, \quad \forall (i,j) \in \mathcal{A}. \qquad (18)$$

The inequalities (10), (11), (12), and (13) can be easily transformed into equalities by adding slack variables

$$
\begin{aligned}
(N-1)\cdot X_{ij} - F_{ij} - S_{ij} &= 0, \quad \forall (i,j) \in \mathcal{A}, && (19)\\
(N-1)\cdot X_{ij} - F_{ji} - S_{ji} &= 0, \quad \forall (i,j) \in \mathcal{A}, && (20)\\
P_i - X_{ij}P_{ij} - T_{ij} &= 0, \quad \forall (i,j) \in \mathcal{A}, && (21)\\
P_j - X_{ij}P_{ji} - T_{ji} &= 0, \quad \forall (i,j) \in \mathcal{A}, && (22)
\end{aligned}
$$

$$S_{ij}, S_{ji}, T_{ij}, T_{ji} \quad \geq \quad 0, \quad \forall (i,j) \in \mathcal{A}. \tag{23}$$

We will add the following (redundant in the MILP) constraint to make the relaxation tighter. Specifically, the constraint guarantees that every node is connected to some other node:

$$\sum_{\{j|(j,i)\in\mathcal{A}\}} X_{ji} + \sum_{\{j|(i,j)\in\mathcal{A}\}} X_{ij} - Q_i = 1, \quad \forall i \in \mathcal{N}, \tag{24}$$

$$Q_i \geq 0, \quad \forall i \in \mathcal{N}. \tag{25}$$

Let also adopt a special variable $V = 1$ or equivalently,

$$V^2 = 1, \quad V \geq 0. \tag{26}$$

Let now $\mathbf{x} = (X_{ij};\ (i,j) \in \mathcal{A})$, $\tilde{\mathbf{x}} = (\mathbf{x}, V)$, $\mathbf{p} = (P_1, \ldots, P_N)$, $\mathbf{f} = (F_{ij}, F_{ji};\ (i,j) \in \mathcal{A})$, $\mathbf{s} = (S_{ij}, S_{ji};\ (i,j) \in \mathcal{A})$, $\mathbf{t} = (T_{ij}, T_{ji};\ (i,j) \in \mathcal{A})$, and $\mathbf{q} = (Q_1, \ldots, Q_N)$. Let also $k(X_{ij})$ (respectively, $k(F_{ij})$, $k(T_{ij})$) denote the position of X_{ij} (respectively, F_{ij}, T_{ij}) in $\mathbf{x}$ (respectively, $\mathbf{f}$, $\mathbf{t}$). Define $\mathbf{X} = \tilde{\mathbf{x}}\tilde{\mathbf{x}}'$ and $\mathbf{Z} = diag(\mathbf{X}, diag(\mathbf{p}), diag(\mathbf{f}), diag(\mathbf{s}), diag(\mathbf{t}), diag(\mathbf{q}))$. Next, we consider the constraints of the MILP one by one and write them in an SDP form.

Eq. (7) can be written as

$$diag(\mathbf{0}, \mathbf{0}, \mathbf{B}_1, \mathbf{0}, \mathbf{0}, \mathbf{0}) \bullet \mathbf{Z} = N - 1, \tag{27}$$

where $\mathbf{B}_1$ is a matrix with $(\mathbf{B}_1)_{k(F_{1j}),k(F_{1j})} = 1$ for $(1,j) \in \mathcal{A}$ and all other entries zero.

Eq. (8) can be written as

$$diag(\mathbf{0}, \mathbf{0}, \mathbf{B}_2, \mathbf{0}, \mathbf{0}, \mathbf{0}) \bullet \mathbf{Z} = 0, \tag{28}$$

where $\mathbf{B}_2$ is a matrix with $(\mathbf{B}_2)_{k(F_{j1}),k(F_{j1})} = 1$ for $(1,j) \in \mathcal{A}$ and all other entries zero.

Eq. (9) can be written as

$$diag(\mathbf{0}, \mathbf{0}, \mathbf{B}_3^i, \mathbf{0}, \mathbf{0}, \mathbf{0}) \bullet \mathbf{Z} = 1, \quad \forall i \in \mathcal{N} \setminus \{1\}, \tag{29}$$

where $\mathbf{B}_3^i$ is a matrix with $(\mathbf{B}_3^i)_{k(F_{ji}),k(F_{ji})} = 1$ for all j such that (j,i) or $(i,j) \in \mathcal{A}$, $(\mathbf{B}_3^i)_{k(F_{ij}),k(F_{ij})} = -1$ for all j such that (j,i) or $(i,j) \in \mathcal{A}$ and all other entries zero.

Eq. (18) can be written as

$$diag(\mathbf{B}_4^{ij}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}) \bullet \mathbf{Z} = 0,\ \forall (i,j) \in \mathcal{A}, \tag{30}$$

where $\mathbf{B}_4^{ij}$ is a matrix with $(\mathbf{B}_4^{ij})_{k(X_{ij}),|\mathcal{A}|+1} = (\mathbf{B}_4^{ij})_{|\mathcal{A}|+1,k(X_{ij})} = -1/2$, $(\mathbf{B}_4^{ij})_{k(X_{ij}),k(X_{ij})} = 1$ and all other entries are equal to zero.

Similarly, $V^2 = 1$ (cf. (26)) can be written as

$$diag(\mathbf{B}_5, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}) \bullet \mathbf{Z} = 1, \tag{31}$$

where $(\mathbf{B}_5)_{|\mathcal{A}|+1,|\mathcal{A}|+1} = 1$ and all other entries zero.

Eq. (19), (20) can be written as

$$diag(\mathbf{B}_6^{ij}, \mathbf{0}, \mathbf{B}_7^{ij}, \mathbf{B}_7^{ij}, \mathbf{0}, \mathbf{0}) \bullet \mathbf{Z} = 0,\ \forall (i,j) \in \mathcal{A}, \tag{32}$$

$$diag(\mathbf{B}_6^{ij}, \mathbf{0}, \mathbf{B}_7^{ji}, \mathbf{B}_7^{ji}, \mathbf{0}, \mathbf{0}) \bullet \mathbf{Z} = 0,\ \forall (i,j) \in \mathcal{A}, \tag{33}$$

where $(\mathbf{B}_6^{ij})_{k(X_{ij}),|\mathcal{A}|+1} = (\mathbf{B}_6^{ij})_{|\mathcal{A}|+1,k(X_{ij})} = (N-1)/2$, $(\mathbf{B}_7^{ij})_{k(F_{ij}),k(F_{ij})} = -1$, and all other entries zero.

Eq. (21), (22) can be written as

$$diag(\mathbf{B}_8^{ij}, \mathbf{B}_9^i, \mathbf{0}, \mathbf{0}, \mathbf{B}_{10}^{ij}, \mathbf{0}) \bullet \mathbf{Z} = 0, \quad \forall (i,j) \in \mathcal{A}, \tag{34}$$

$$diag(\mathbf{B}_8^{ji}, \mathbf{B}_9^j, \mathbf{0}, \mathbf{0}, \mathbf{B}_{10}^{ji}, \mathbf{0}) \bullet \mathbf{Z} = 0, \quad \forall (i,j) \in \mathcal{A}, \tag{35}$$

where $(\mathbf{B}_8^{ij})_{k(X_{ij}),|\mathcal{A}|+1} = (\mathbf{B}_8^{ij})_{|\mathcal{A}|+1,k(X_{ij})} = -P_{ij}/2$, $(\mathbf{B}_9^i)_{i,i} = 1$, $(\mathbf{B}_{10}^{ij})_{k(T_{ij}),k(T_{ij})} = -1$, and all other entries are equal to zero.

Eq. (14) can be written as

$$diag(\mathbf{B}_{11}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}) \bullet \mathbf{Z} = N - 1, \tag{36}$$

where $(\mathbf{B}_{11})_{k(X_{ij}),|\mathcal{A}|+1} = (\mathbf{B}_{11})_{|\mathcal{A}|+1,k(X_{ij})} = 1/2$ for all $(i,j) \in \mathcal{A}$.

Eq. (24) can be written as

$$diag(\mathbf{B}_{12}^i, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{B}_{13}^i) \bullet \mathbf{Z} = 1, \quad \forall i \in \mathcal{N}, \tag{37}$$

where $(\mathbf{B}_{12}^i)_{k(X_{ji}),|\mathcal{A}|+1} = (\mathbf{B}_{12}^i)_{|\mathcal{A}|+1,k(X_{ji})} = 1/2$ for all $(j,i) \in \mathcal{A}$, $(\mathbf{B}_{12}^i)_{k(X_{ij}),|\mathcal{A}|+1} = (\mathbf{B}_{12}^i)_{|\mathcal{A}|+1,k(X_{ij})} = 1/2$, for all $(i,j) \in \mathcal{A}$, $(\mathbf{B}_{13}^i)_{i,i} = -1$, and all other entries zero.

Combining all the above equations we obtain the SDP relaxation of Figure 2. The following result is immediate since we are relaxing the MILP.

$$\begin{aligned} C^{SDP} = \min \quad & diag(\mathbf{0}, \mathbf{I}, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}) \bullet \mathbf{Z} \\ s.t. \quad & (27), (28), (29), (30), (31), \\ & (32), (33), (34), (35), (36), (37), \\ & \mathbf{Z} \succeq 0. \end{aligned} \tag{38}$$

Fig. 2. The SDP relaxation of the MILP of Fig. 1.

Proposition III.1 *It holds $C^{SDP} \leq C^{MILP}$.*

In general, the solution produced from the SDP does not provide us with integer X_{ij}'s. Thus, we will "project" the SDP solution to obtain effective MILP feasible solutions. To that end, notice that a matrix $\mathbf{X}$ that is optimal for the SDP is positive semi-definite and has the form

$$\mathbf{X} = \begin{bmatrix} \mathbf{R_x} & \mathbf{m_x} \\ \mathbf{m_x'} & 1 \end{bmatrix}. \tag{39}$$

It follows that $\mathbf{\Sigma_x} = \mathbf{R_x} - \mathbf{m_x}\mathbf{m_x'} \succeq 0$ can be interpreted as a covariance matrix. This motivates us to think of the SDP as providing a probability distribution for $\mathbf{x}$ which we can sample (and project) to obtain MILP feasible solutions. On a notational remark, we will use $\mathbf{x}(X_{ij})$ and $\mathbf{m_x}(X_{ij})$ to denote the value of $\mathbf{x}$ and $\mathbf{m_x}$, respectively, corresponding to X_{ij}.

The algorithm of Fig. 3 yields a feasible solution $\mathbf{x}^r$ of the MILP. It uses the following parameters: a rounding threshold α ($0 < \alpha \leq 1$), a shrinking factor β ($0 < \beta < 1$), and a maximum number of iterations. These parameters can be tuned to improve performance.

Note that the algorithm produces an integer solution $\mathbf{x}^r$ that specifies which links in $\mathcal{A}$ are selected. Due to Step 7, $\mathbf{x}^r$ induces a bidirectional spanning tree with $N-1$ bidirectional links. To evaluate the cost of this tree, for each node $i \in \mathcal{N}$ set $P_i^r = \max\{P_{ij} \mid \mathbf{x}^r(X_{ij}) = 1 \text{ or } \mathbf{x}^r(X_{ji}) = 1\}$. The total energy cost of $\mathbf{x}^r$ is:

$$C^{RSDP} = \sum_{i=1}^{N} P_i^r, \tag{40}$$

Step 1: Solve the SDP of Fig. 2 and obtain an optimal solution for $\mathbf{X}$ in the form (39). Let $\mathbf{R_x}$, $\mathbf{m_x}$, and $\mathbf{\Sigma_x}$ as defined earlier. Set $\mathbf{x}^r = \mathbf{0}$.
Step 2 (Rounding): $\forall (i,j) \in \mathcal{A}$ such that $\mathbf{x}^r(X_{ij}) = 0$ set $\mathbf{x}^r(X_{ij}) = 1$ if $\mathbf{m_x}(X_{ij}) > \alpha$. Check whether the undirected graph $(\mathcal{N}, \{(i,j) \mid \mathbf{x}^r(X_{ij}) = 1\})$ is connected. If it is, go to Step 7.
Step 3 (Sampling): Draw $\mathbf{x}^s = \mathbf{m_x} + (\mathbf{\Sigma_x})^{1/2}\mathbf{g}$ where $\mathbf{g}$ is a standard Gaussian random vector $N(\mathbf{0}, \mathbf{I})$. For all $(i,j) \in \mathcal{A}$ such that $\mathbf{x}^r(X_{ij}) = 0$ set $\mathbf{x}^r(X_{ij}) = 1$ if $\mathbf{x}^s(X_{ij}) > \alpha$.
Step 4: Check again whether the graph induced by $\mathbf{x}^r$ is connected. If it is, go to Step 7. If the maximal iteration count has been reached go to Step 6.
Step 5: In the SDP of Fig. 2 fix all X_{ij} with $\mathbf{x}^r(X_{ij}) = 1$ and resolve to obtain an optimal solution for $\mathbf{X}$ in the form (39) with components $\mathbf{R_x}$, $\mathbf{m_x}$, and associated $\mathbf{\Sigma_x}$. Go to Step 2.
Step 6 (Connectivity) : Reduce $\alpha := \beta\alpha$. Redraw $\mathbf{x}^s = \mathbf{m_x} + (\mathbf{\Sigma_x})^{1/2}\mathbf{g}$ where $\mathbf{g} \sim N(\mathbf{0}, \mathbf{I})$. For all $(i,j) \in \mathcal{A}$ such that $\mathbf{x}^r(X_{ij}) = 0$ set $\mathbf{x}^r(X_{ij}) = 1$ if $\mathbf{x}^s(X_{ij}) > \alpha$. If the updated $\mathbf{x}^r$ corresponds to a connected graph, go to Step 7; otherwise, repeat Step 6.
Step 7 (Link Trimming): Sort all $(i,j) \in \mathcal{A}$ such that $\mathbf{x}^r(X_{ij}) = 1$ in descending order of the power needed to support the link, i.e., $P_{ij} + P_{ji}$. Examine each link (i,j) in this list in descending order. If (i,j) can be removed without disconnecting the network then set $\mathbf{x}^r(X_{ij}) = 0$.
Step 8: Output $\mathbf{x}^r$.

Fig. 3. A recursive SDP algorithm that produces a solution $\mathbf{x}^r$ corresponding to a bidirectional spanning tree.

and $C^{MILP} \leq C^{RSDP}$.

We note that C^{RSDP} is the energy cost of building the bidirectional spanning tree as well as applying one iteration of the agreement Algorithm 1.

C. Graph-based algorithms

Next we devise a series of graph-based algorithms. By construction, all these algorithms provide upper bounds to the optimal MILP cost and in some cases a lower bound as well. The graph we will consider is an augmented graph that "transfers" the energy costs from the nodes to links so that suitable network flow algorithms can be employed.

1) Augmented graph construction: Consider the set of "real" nodes $\mathcal{N}$. For each $i \in \mathcal{N}$, define $\mathfrak{N}(i) = \{j | j \in \mathcal{N}, (i,j) \text{ or } (j,i) \in \mathcal{A}\}$ as the potential neighboring nodes of i. For each real node i we add a set of $|\mathfrak{N}(i)|$ artificial nodes, each one corresponding to a different potential neighbor of i. Each artificial node corresponds to a different power level that node i may work on. The detailed augmented graph construction is in Fig. 4. We will denote this graph by $\mathcal{G}_a = (\mathcal{N} \cup \mathcal{V}, \mathcal{E}_a)$ where $\mathcal{V}$ denotes the set of artificial nodes.

The tag of an artificial node v indicates that v can act as a relay for data sent from i to j, which requires power of at least P_{ij} at node i. The augmented graph contains links between a node and its artificial nodes and also links between artificial

Step 1: For each ordered pair (i,j) of nodes $i,j \in \mathcal{N}$ such that $j \neq i$ and $P_{ij} \leq P_i^{max}$ create an artificial node v associated with node i and assign to v the tag $\mathbf{T}_v = (i, j, P_{ij})$. Add an undirected link from i to v with cost $\mathbf{T}_v(3) = P_{ij}$.
Step 2: Consider each pair of artificial nodes v_1 and v_2 with tags $\mathbf{T}_{v_1} = (i_1, j_1, P_{i_1 j_1})$ and $\mathbf{T}_{v_2} = (i_2, j_2, P_{i_2 j_2})$ such that $v_1 \neq v_2$ and $i_1 \neq i_2$. If $P_{i_1 j_1} = \mathbf{T}_{v_1}(3) \geq P_{i_1 i_2}$ and $P_{i_2 j_2} = \mathbf{T}_{v_2}(3) \geq P_{i_2 i_1}$ then add an undirected link (v_1, v_2) with cost $w\,(\mathbf{T}_{v_1}(3) + \mathbf{T}_{v_2}(3))$, where w is a very small positive constant.

Fig. 4. Construction of the augmented graph.

nodes corresponding to potential bidirectional links between real nodes. We assign a small cost to these latter links to avoid degenerate solutions in the algorithms we employ. From the graph $\mathcal{G}_a$ we construct a directed graph $\tilde{\mathcal{G}}_a = (\mathcal{N} \cup \mathcal{V}, \mathcal{A}_a)$ such that for every undirected link $(i,j) \in \mathcal{E}_a$, $\mathcal{A}_a$ contains both directed links (i,j) and (j,i), each with cost the same as the cost of the undirected link (i,j).

2) Minimum cost flow problem: In this subsection, we will apply a minimum cost flow algorithm to the (directed) augmented graph $\tilde{\mathcal{G}}_a$. Then, based on the optimal solution, we will construct a bidirectional spanning tree for the original graph $\mathcal{G}$ and derive bounds on the optimal value of the MILP.

We take one real node in $\mathcal{N}$, say node s, as the source node having a supply of $N-1$ units. Every other real node in $\mathcal{N} \setminus \{s\}$ has a demand of 1 unit. All the artificial nodes in $\mathcal{V}$ have zero supply and demand. With these demands and supplies we solve the uncapacitated min cost flow problem on graph $\tilde{\mathcal{G}}_a$ (see [24]). This is the linear programming (LP) problem of minimizing the total cost of shipping flows from the nodes with supplies to the nodes with demands, where arc costs indicate cost per unit of flow. Such an LP can be solved by standard LP methods or special purpose methods (e.g., network simplex) that exploit its special structure. Let C_s^{MCF} denote its optimal value and $\mathbf{f}_s^{MCF} = (F_{ij}; \forall (i,j) \in \mathcal{A}_a)$ denote an optimal solution where F_{ij} is the optimal flow on arc $(i,j) \in \mathcal{A}_a$. Clearly, we can solve N different min cost flow problems depending on the selection of the source node $s = 1, \ldots, N$.

Given the solution of any of the above flow problems, we can construct a bidirectional spanning tree for the original graph $\mathcal{G}$. The procedure is described in Fig. 5. It builds an undirected graph $\mathcal{G}_{BST,s}^{MCF} = (\mathcal{N}, \mathcal{E}_{BST,s}^{MCF})$ which is a bidirectional spanning tree of $\mathcal{G}$.

In Step 1 we identify all links between artificial nodes with a positive flow. Note that a real node i may have an artificial node v with a link (u,v) (or (v,u)) selected in Step 1 but with no flow in the (i,v) or the (v,i) link. This may happen if v serves as a relay to node u. Step 2 is similar to Step 7 of the Algorithm in Fig. 3 and eliminates redundant links to end up with a spanning tree ($N-1$ links). Note that the solution to the min cost flow problem guarantees that we will end up with a connected network since it satisfies flow conservation and ships flow from the source s to every real node in $\mathcal{N} \setminus \{s\}$.

Step 1: Solve the min cost flow problem with source s. Initialize $\mathcal{E}_{BST,s}^{MCF} = \emptyset$. For any pair (v,u), $v,u \in \mathcal{V}$, such that $\mathbf{f}_s^{MCF}(v,u) > 0$ or $\mathbf{f}_s^{MCF}(u,v) > 0$ add the link $(\min\{i,j\}, \max\{i,j\})$ to $\mathcal{E}_{BST,s}^{MCF}$ where i and j are the real nodes corresponding to v and u, respectively.
Step 2: Sort all $(i,j) \in \mathcal{E}_{BST,s}^{MCF}$ in descending order of the power needed to support the link, i.e., $P_{ij} + P_{ji}$. Examine each link (i,j) in this list in descending order. If (i,j) can be removed without disconnecting the network then remove (i,j) from $\mathcal{E}_{BST,s}^{MCF}$.
Step 3: Set the power at i as $P_i = \max\{P_{ij} \mid (i,j) \text{ or } (j,i) \in \mathcal{E}_{BST,s}^{MCF}\}$. Let $C_{BST,s}^{MCF} = \sum_{i\in\mathcal{N}} P_i$.

Fig. 5. Constructing a bidirectional spanning tree from the min cost flow solution.

The minimum cost flow problem also provides us with bounds on the optimal objective value C^{MILP} of the MILP.

Theorem III.2 *As $w \to 0$ it holds*

$$\frac{N}{2(N-1)} \min_s C_s^{MCF} \le \frac{1}{2(N-1)} \sum_{s=1}^{N} C_s^{MCF} \le C^{MILP} \le \min_s C_{BST,s}^{MCF}.$$

Proof: The far right inequality is immediate since the Algorithm of Fig. 5 produces a bidirectional spanning tree, thus forming a feasible solution to the MILP. The far left inequality is also obvious.

To establish the middle inequality we will start from an optimal solution of the MILP and construct a feasible solution for the minimum cost flow problem. Let $\{X_{ij}\}$, $\{F_{ij}\}$ and $\{P_i\}$ form an optimal solution of the MILP. For each real node $i \in \mathcal{N}$ add a virtual node v_i with tag $\mathbf{T}_{v_i} = (i, j, Y_i)$, where $j = \arg\max_j\{P_{ij} | X_{ij} = 1 \text{ or } X_{ji} = 1\}$. Then, introduce the directed arcs (i, v_i) and (v_i, i) with arc weights P_i. For any two real nodes $i_1, i_2 \in \mathcal{N}$, if $X_{i_1 i_2} = 1$, then introduce the directed arcs (v_{i_1}, v_{i_2}) and (v_{i_2}, v_{i_1}) with arc weights $w(\mathbf{T}_{v_{i_1}}(3)+\mathbf{T}_{v_{i_2}}(3))$. Note that the graph we have constructed is a bidirectional spanning tree because it has been constructed based on the solution of the MILP.

Fix now some node $s \in \mathcal{N}$ and designate it as a source node. Set the flows on the graph constructed above in order to ship $N-1$ units of supply from s to each other node which should receive 1 unit of flow. The resulting flow vector is a feasible solution of the min cost flow problem. Notice that the flow on any arc (v_{i_1}, v_{i_2}) between artificial nodes can not be greater than $N-1$. Denote the largest weight of such an arc as $w \max_{i_1,i_2}(\mathbf{T}_{v_{i_1}}(3) + \mathbf{T}_{v_{i_2}}(3)) = w\mathbf{T}_{\max}$. Then, the flow cost on (v_{i_1}, v_{i_2}) must be no more than $w(N-1)\mathbf{T}_{\max}$. Since there are $(N-1)$ arcs between artificial nodes with positive flows, the flow cost on these arcs can not be larger than $w(N-1)^2\mathbf{T}_{\max}$. The feasibility of the constructed feasible solution for the min cost flow problem yields

$$C_s^{MCF} \le \sum_{i=1,i\ne s}^{N} P_i + P_s(N-1) + w(N-1)^2\mathbf{T}_{\max}$$
$$= C^{MILP} + (N-2)\cdot P_s + w(N-1)^2\mathbf{T}_{\max}.$$

Adding the above inequalities for $s = 1,\ldots,N$ we have

$$\sum_{s=1}^{N} C_s^{MCF} \le 2(N-1)C^{MILP} + wN(N-1)^2\mathbf{T}_{\max}.$$

Taking $w \to 0$ we arrive at the desired result. ■

3) Distributed approaches: The approach we detailed in the previous sections yields a bidirectional spanning tree and bounds on the optimal performance. It is though a centralized approach as both the standard min cost flow solution methods and Step 2 of the Algorithm in Fig. 5 require centralized computations. In this subsection we discuss how to construct a bidirectional spanning tree in a distributed manner.

The first observation is that the min cost flow problem we are solving is a single-source-multiple-destination problem. It follows that optimal flows are shipped along shortest paths from the source to each destination. As a result, an alternative way for solving the problem is to compute these shortest paths. To that end, we can use the distributed asynchronous Bellman-Ford algorithm [25] on the graph $\tilde{\mathcal{G}}_a$. Note that based on our connectivity assumption and the way $\tilde{\mathcal{G}}_a$ is constructed, every link is bidirectional, the graph is strongly connected, and every link has a positive cost which implies that all cycles have a positive cost. Under these conditions, the asynchronous Bellman-Ford algorithm converges. Notice that in worst case, the distributed asynchronous Bellman-Ford algorithm may require an excessive number of iterations to terminate [25]. Alternatively, some other distributed algorithms [26] based on Dijkstra's algorithm can be employed to find the single source shortest paths.

The second observation deals with the distributed implementation of Step 2 in the Algorithm of Fig. 5. Examine that algorithm and note that after Step 1, the undirected graph $\mathcal{G}_{BST,s}^{MCF} = (\mathcal{N}, \mathcal{E}_{BST,s}^{MCF})$ is connected because it is constructed from the min cost flow solution which ships flow to every node from the source node s. (One can easily construct examples, where $\mathcal{G}_{BST,s}^{MCF}$ contains more than $N-1$ links.) We can now assign to each link $(i,j) \in \mathcal{E}_{BST,s}^{MCF}$ a weight equal to $P_{ij} + P_{ji}$ which is the power needed to support the link. Then we can run a distributed algorithm proposed by in [12] to construct a minimum weight spanning tree. The resulting spanning tree is a bidirectional spanning tree whose overall energy cost can be evaluated as in Step 3 of the Algorithm in Fig. 5.

Notice that for the distributed graph-based algorithms, one incurs some communication overhead for finding the bidirectional spanning tree. We next assess the associated energy cost.

In the distributed approach we presented above, the total number of messages exchanged by the shortest path algorithm is $O\big((|\mathcal{N}| + |\mathcal{V}|)^{\frac{5}{3}}\big)$ [26]. Computing the minimum spanning tree requires $O\big(|\mathcal{E}_{BST,s}^{MCF}| + |\mathcal{N}| \log |\mathcal{N}|\big)$ messages ([12]). In the case of a planar and non-dense network, we have $|\mathcal{V}| = O(|\mathcal{N}|)$ and $|\mathcal{E}_{BST,s}^{MCF}| = O(|\mathcal{N}|)$ and the total number of messages exchanged is $O(|\mathcal{N}|^{\frac{5}{3}})$. The corresponding energy cost is $O(|\mathcal{N}|^{\frac{5}{3}}) \cdot P_{\max} = O(|\mathcal{N}|^{\frac{5}{3}})$, where $P_{\max}$ denotes the maximum energy cost for one message to be transmitted.

Once the sub-optimal bidirectional spanning tree is found, the energy cost of building the topology and applying one iteration of the agreement Algorithm 1 is $C^{MCF}_{BST,s}$.

A similar distributed approach using the algorithm in [12] can also be used on the undirected augmented graph $\mathcal{G}_a = (\mathcal{N} \cup \mathcal{V}, \mathcal{E}_a)$, see Fig. 6.

Step 1: Apply the algorithm of [12] to construct a minimum spanning tree of $\mathcal{G}_a$.
Step 2: Remove all artificial nodes which are leaf nodes in the tree and their incident arcs until no artificial node which is a leaf node remains.
Step 3: The resulting graph contains a path between any two real nodes, potentially through artificial nodes. Remove all artificial nodes but maintain the exact same path between any two real nodes (going through the same set of real nodes as before).
Step 4: The resulting undirected graph is a spanning tree $\mathcal{G}^{MST} = (\mathcal{N}, \mathcal{E}^{MST})$. Set the power level of node i as $P_i = \max\{P_{ij} \mid (i,j) \in \mathcal{E}^{MST}\}$. Let $C^{MST} = \sum_{i\in\mathcal{N}} P_i$.

Fig. 6. Constructing a bidirectional spanning tree by solving a min weight spanning tree problem on the augmented graph $\mathcal{G}_a$.

Since only the distributed minimum spanning tree algorithm is employed in the algorithm of Fig. 6, the total energy cost of finding and constructing the bidirectional spanning tree is $O\big(|\mathcal{E}_a| + (|\mathcal{N}| + |\mathcal{V}|)\log(|\mathcal{N}| + |\mathcal{V}|)\big) = O\big(|\mathcal{N}|\log|\mathcal{N}|\big)$, in the case of $|\mathcal{E}_a| = O\big(|\mathcal{N}|\big)$ and $|\mathcal{V}| = O\big(|\mathcal{N}|\big)$. Similarly, after computing the spanning tree, the energy cost of building the topology and applying one iteration of the agreement Algorithm 1 is C^{MCT}.

D. *Numerical results*

We generate networks by uniformly scattering N nodes on a 10×10 square. We assume that the minimum power needed by a node to reach another node is d^2, where d is their distance. We set the maximum power of each node to $10^2\lambda$, where λ is a parameter we can tune. Small values of λ induce a "sparse" network where a large λ implies many potential bidirectional links and induces a "dense" network. In the results we present we take $\lambda = 0.2$ for sparse networks and $\lambda = 0.4$ for dense networks.

Tables I and II report our results for different instances (N ranging from 10 to 50) corresponding to sparse and dense networks, respectively. In the tables, MILP denotes the optimal value of the MILP in Fig. 1 which was solved using the CPLEX solver. The larger instances were not possible to solve in a reasonable amount of time. RT denotes the running time for each algorithm we compare. (All algorithms were run on a computer with a Ubuntu-8.04-OS, 2GB of memory, and an Intel-XEON-2.00GHz CPU.) SDP denotes the optimal value of the SDP relaxation given in Fig. 2 which was solved using the SDPA solver. The "Ratio" rows compute the ratio of the entries in the immediately preceding row over the MILP optimal cost. RSDP UB denotes the cost of the bidirectional spanning tree obtained from the Algorithm in Fig. 3. In that algorithm we set the maximum number of iterations to 4, and use $\alpha = 0.5$, and $\beta = 0.9$. MCF LB and UB denote the values obtained from Thm. III.2 and the Algorithm of Fig. 5, respectively. Finally, MST UB denotes the cost of the spanning tree obtained from the Algorithm of Fig. 6.

TABLE I
NUMERICAL EXPERIMENTS ON SPARSE NETWORKS.

Number of Nodes	10	20	30	40	50
MILP	49.32	62.77	81.19	NA	NA
MILP RT (sec)	< 1	66	59792		
SDP	23.85	18.66	24.75	25.69	28.42
Ratio	0.48	0.30	0.30		
RSDP UB	63.98	82.59	116.08	126.27	95.91
Ratio	1.30	1.32	1.43		
RSDP RT (sec)	7	178	765	5519	24454
MCF LB	25.99	24.99	38.57	49.52	48.39
Ratio	0.53	0.40	0.48		
MCF UB	49.32	64.05	87.44	86.79	92.08
Ratio	1.00	1.02	1.08		
MCF RT (sec)	< 1	2	6	32	120
MST UB	51.24	67.26	84.96	87.15	89.77
Ratio	1.04	1.07	1.05		
MST RT (sec)	< 1	< 1	1	11	154

TABLE II
NUMERICAL EXPERIMENTS ON DENSE NETWORKS.

Number of Nodes	10	20	30	40	50
MILP	77.11	82.71	68.38	NA	NA
MILP RT (sec)	1	41	66416		
SDP	42.06	31.40	16.68	19.50	18.07
Ratio	0.55	0.38	0.24		
RSDP UB	81.05	135.24	112.62	162.41	120.79
Ratio	1.05	1.64	1.65		
RSDP RT (sec)	10	405	5138	19048	72822
MCF LB	45.86	46.12	39.81	39.73	38.52
Ratio	0.59	0.56	0.58		
MCF UB	83.16	82.89	72.63	85.95	71.92
Ratio	1.08	1.00	1.06		
MCF RT (sec)	1	3	36	110	314
MST UB	83.16	84.56	71.42	87.76	72.24
Ratio	1.08	1.02	1.04		
MST RT (sec)	1	1	35	208	998

The results indicate that both the min-cost-flow-based algorithm (Secs. III-C2 and III-C3) and the minimum-weight-spanning-tree-based algorithm (Fig. 6) can produce solutions that very close to the optimal for all those instances where an optimal MILP solution is obtainable. For the larger instances they dominate the SDP-based solution. Moreover, the lower bounds of Thm. III.2 based on the min-cost-flow-based algorithm are tighter than the lower bounds from the SDP relaxation, though the results vary, depending on the network configurations. It is important to note that the bidirectional spanning trees produced by the approaches in Sec. III-C3 can be obtained in a distributed manner which is critical for large-scale wireless sensor networks.

IV. ENERGY EFFICIENT TOPOLOGY FOR AVERAGING – DYNAMIC NETWORKS

A. *Problem formulation*

Next we turn our attention to networks with dynamically changing topologies.

1) Network model: We retain the network model in Sec. III-A with the following modification. Notice that any two nodes i, j with $(i, j) \in \mathcal{A}$ can communicate with each other. However, any attempt to communicate may not be successful due to the dynamics in the physical environment and the interference from other nodes. To model this uncertainty we introduce a *truncated* geometric distribution with parameters (p_k, M_k) for each (bidirectional) link $k = (i, j) \in \mathcal{A}$. We denote by Y_k the number of trials until a success, which follows

$$P(Y_k = y) = \begin{cases} \frac{p_k(1-p_k)^{y-1}}{1-(1-p_k)^{M_k}}, & y = 1, 2, \ldots, M_k; \\ 0, & \text{otherwise.} \end{cases} \tag{41}$$

Notice that the coefficient $1/\big(1-(1-p_k)^{M_k}\big)$ is to ensure that the probabilities add up to one. Suppose one would like build a link between two nodes. One can use proper scheduling schemes to combat interference from neighboring links (see for instance work on the CSMA protocol [27] and other schemes that completely avoid interference [28]). We assume that the use of these techniques ensures that it takes no more than M_k trials for a successful transmission. The expected number of trials needed is:

$$\begin{aligned} E[Y_k] &= \sum_{y=1}^{M_k} P(Y_k = y) \cdot y \\ &= \frac{1 - (M_k + 1)(1 - p_k)^{M_k} + M_k(1 - p_k)^{M_k+1}}{p_k\big(1 - (1 - p_k)^{M_k}\big)}. \end{aligned} \tag{42}$$

For every trial on link k, let $c_k \geq 0$ be the energy cost for each node, which depends on the power level needed to support the link. Comparing this with the model we considered in Sec. III, c_k can be seen as the sum of the energy costs incurred by the two nodes incident to link k. The expected cost to successfully construct this link is:

$$\begin{aligned} E[C_k] &= c_k E[Y_k] \\ &= c_k \frac{1 - (M_k + 1)(1 - p_k)^{M_k} + M_k(1 - p_k)^{M_k+1}}{p_k\big(1 - (1 - p_k)^{M_k}\big)}. \end{aligned} \tag{43}$$

Once messages have been successfully exchanged on link k then there are no subsequent transmissions (hence no energy cost) until the next time messages need to be exchanged on this link (as may be needed by Algorithm 2).

We note that given m_k failed trials on link k, the total number of trials until a success still follows a truncated geometric distribution (memoryless property):

$$\begin{aligned} P(Y_k = y + m_k | Y_k > m_k) &= \frac{P(Y_k = y + m_k)}{P(Y_k > m_k)} \\ = \begin{cases} \frac{p_k(1-p_k)^{y-1}}{1-(1-p_k)^{M_k-m_k}}, & y = 1, 2, \ldots, M_k - m_k; \\ 0, & \text{otherwise.} \end{cases} \end{aligned} \tag{44}$$

This is equivalent to a geometric distribution with parameter $(p_k, M_k - m_k)$. Analogously, $E[Y_k|Y_k > m_k]$ and $E[C_k|Y_k > m_k]$ can be calculated analytically. The following Lemma is immediate.

Lemma IV.1 *$P[Y_k = m_k + 1|Y_k > m_k]$ is a monotonically increasing function of m_k.*

Remarks: Two issues should be clarified here. First, we assume that any potential link in the network can be constructed after some potentially large but fixed number of trials, i.e., we assume M_k to be finite. This is to ensure that Assumption C holds for the consensus algorithm to converge. Otherwise, any policy could end up with a non-connected graph with some positive probability. Second, the only other property needed for results in this section to hold is monotonicity (cf. Lemma IV.1). Essentially, every failed trial provides some more information about the link between two nodes, such as channel condition, antenna angle etc., thus, facilitating the link construction in the next trial. For simplicity of the exposition, we adopt the truncated geometric distribution to characterize the link construction process. Generally, any model resulting in Lemma IV.1 is applicable.

2) Dynamic programming formulation: As we have seen in Sec. II-C, to guarantee convergence of the load balancing algorithm we need to enforce Assumption C. The convergence time was shown to be polynomial in B and N. In this section we formulate the problem of efficiently enforcing Assumption C, either by minimizing the time until the graph becomes strongly connected or by minimizing the energy cost of doing so.

As before, let $\mathcal{G}(t) = (\mathcal{N}, \mathcal{E}(t)), \mathcal{E}(t) \subseteq \mathcal{A}$, denote the communication pattern between the nodes at time t and let $\mathcal{E}_c(t) = \mathcal{E}(1) \cup \mathcal{E}(2) \cup \cdots \cup \mathcal{E}(t)$ and $\mathcal{G}_c(t) = (\mathcal{N}, \mathcal{E}_c(t))$. We treat these graphs as undirected and we say that $\mathcal{G}_c(t)$ is strongly connected if the directed graph formed by replacing each link (i, j) with the two directed links (i, j) and (j, i) is strongly connected. We split time into blocks of length B. In each block, say $\{1, \ldots, B\}$, we seek to successfully connect enough links so that $\mathcal{G}_c(B)$ is strongly connected. We can repeat this process in every such block, so we only focus on the link selection decisions concerning a single block.

For simplicity of the analysis, we assume that at every discrete time instant we select a single link and attempt to establish communications between its incident nodes. Such an attempt may succeed or fail according to the model we presented earlier. We note that more than one links can attempt communications at the same time as long as there is no interference between them. To account for this we would need to assume an interference model and resolve the underlying scheduling problem (see e.g., [28], [29]). In this paper however, we are only focusing on the dynamic link selection and we forgo these physical layer issues.

Before we present the DP formulation let us define the "state" of the network at time t which consists of two parts. One part is the graph $\mathcal{G}_c(t)$, which includes all successfully constructed links up to time t. The other part contains information on links that have been attempted but not constructed (those that have not been tried can be viewed as have been tried 0 times). We denote the state of these links by $\mathcal{L}(t) = \{k(m_k)\}_{k=1, k \notin \mathcal{E}_c(t)}^{|\mathcal{A}|}$, where $k(m_k)$ indicates that we have made m_k attempts to construct link $k \in \mathcal{A}$. We define the state at time t as $\mathcal{S}(t) = (\mathcal{G}_c(t), \mathcal{L}(t))$.

Recall that for each link k the number of trials until a success follows a truncated geometric distribution with parameters (p_k, M_k). As we have seen, given m_k failures the p.m.f. of the residual number of trials until a success also follows a truncated geometric distribution with parameters $(p_k, M_k - m_k)$. The success probability of the next trial on link k is $\frac{p_k}{1-(1-p_k)^{M_k-m_k}}$. Given the information at time t and under the assumption that at t we only select a single link, say k, to attempt to construct, $\mathcal{S}(t)$ is Markovian and evolves as follows:

$$\mathcal{S}(t+1) = (\mathcal{G}_c(t+1), \mathcal{L}(t+1)) = \begin{cases} ((\mathcal{N}, \mathcal{E}_c(t) \cup \{k\}), \mathcal{L}(t) \setminus k(m_k)), & w.p. \\ & \frac{p_k}{1-(1-p_k)^{M_k-m_k}}, \\ (\mathcal{G}_c(t), \mathcal{L}(t) \setminus k(m_k) \cup k(m_k+1)), & w.p. \\ & 1 - \frac{p_k}{1-(1-p_k)^{M_k-m_k}}. \end{cases} \tag{45}$$

If we wish to minimize the total cost of constructing a strongly connected graph $\mathcal{G}_c(B)$ we end up with the following finite-horizon DP iteration:

$$J_t(\mathcal{S}(t)) = \min_{k \in \{\mathcal{A}, \emptyset\},\ k \notin \mathcal{E}_c(t)} E\big[c_k + J_{t+1}(\mathcal{S}(t+1))\big], \tag{46}$$

where $J_t(\mathcal{S}(t))$ is the optimal cost-to-go (or value) function at state $\mathcal{S}(t)$. We make the convention $c_\emptyset = 0$. The boundary conditions are $J_t(\mathcal{S}(t)) = 0$ if $\mathcal{G}_c(t)$ is strongly connected. We note that as written, (46) allows no link to be selected at any given time t. This amounts to "idling," incurs a zero immediate cost, and can be selected when connectivity of $\mathcal{G}_c(t)$ has already been achieved. The horizon (block length) in (46) is equal to B and we impose a terminal cost at the end of the horizon equal to

$$J_B(\mathcal{S}(B)) = \begin{cases} W, & \text{if } \mathcal{G}_c(B) \text{ is not strongly connected,} \\ 0, & \text{otherwise,} \end{cases} \tag{47}$$

where $W \gg 1$ is a large enough penalty.

We next establish a useful monotonicity property of the value function $J_t(\mathcal{S}(t))$.

Lemma IV.2 *It holds that $J_t(\mathcal{S}^\alpha(t)) \geq J_t(\mathcal{S}^\beta(t))$ for all t and $\mathcal{S}^\alpha(t) = (\mathcal{G}_c^\alpha(t), \mathcal{L}^\alpha(t))$, $\mathcal{S}^\beta(t) = (\mathcal{G}_c^\beta(t), \mathcal{L}^\beta(t))$ such that $\mathcal{G}_c^\alpha(t) \subseteq \mathcal{G}_c^\beta(t)$ and $\mathcal{L}^\alpha(t)$ coincides with $\mathcal{L}^\beta(t)$ for all links $k \notin \mathcal{E}_c^\beta(t)$.*

Proof: We will establish the result using a coupling argument. Fix some time t and consider two systems α and β corresponding to states $\mathcal{S}^\alpha(t)$ and $\mathcal{S}^\beta(t)$, respectively. Notice that the randomness is only due to the link construction process. By defining the two systems on a common probability space, we can assume that for each common link they encounter the same sequences of random numbers uniformly distributed on $[0, 1]$, which in turn decide the success or failure of constructing the link. For some particular sample path let $\pi^\alpha(t)$ be the optimal sequence of links selected by System α starting at t. A policy for β is to mimic α, that is, attempt to construct the exact same links at exactly the same time if the link selected by α does not belong to $\mathcal{G}_c^\beta(t)$. Otherwise, that is if the link selected by α is already in $\mathcal{G}_c^\beta(t)$, System β can idle (at no cost). Due to the coupling, both Systems face the exact same success and failure events on the common links they attempt to construct. Since System β idles on the links it has already constructed, its cost will be no more than the System α cost. Taking expectations and noting that System β used a suboptimal policy we establish that $J_t(\mathcal{S}^\alpha(t)) \geq J_t(\mathcal{S}^\beta(t))$. ■

An immediate corollary is that $\mathcal{G}_c(B)$ should be a tree; otherwise we can simply construct only links in a spanning tree of $\mathcal{G}_c(B)$ which will reduce the overall cost. The DP formulation is insightful but it does not lead to practical (efficient) algorithms. Next we develop such algorithms.

B. Energy Efficient Averaging in Dynamic WSNETs with Large Enough Horizon Length

We start with the simpler case where B is large enough, or more specifically $B \geq \sum_{k \in \mathcal{A}} M_k$, so that for any feasible policy network interconnectivity is guaranteed to be reached before B, hence, the terminal cost is always 0. The next Proposition establishes another monotonicity property of $J_t(\cdot)$. Because the horizon is long enough and we pay no terminal cost with probability one (w.p.1.), we do not need to keep track of time; we will simplify the notation for the states and the cost-to-go function by writing $\mathcal{S} = (\mathcal{G}_c, \mathcal{L})$ and $J(\mathcal{S})$, respectively.

Proposition IV.1 *Suppose we are at some state $\mathcal{S}^\alpha = (\mathcal{G}_c^\alpha, \mathcal{L}^\alpha)$ and consider link $k \notin \mathcal{E}_c^\alpha$. Assume that there is a positive probability that link k participates in the connected graph at the end of the horizon, i.e., $k \in \mathcal{E}_c^\alpha$ at time B. Consider some other state $\mathcal{S}^\beta = (\mathcal{G}_c^\beta, \mathcal{L}^\beta)$ such that $\mathcal{G}_c^\alpha = \mathcal{G}_c^\beta$ and $\mathcal{L}^\beta = \mathcal{L}^\alpha \setminus k(m_k) \cup k(m_k+1)$. Then, $J(\mathcal{S}^\alpha) > J(\mathcal{S}^\beta)$.*

Proof: We will use a coupling argument as in the proof of Lemma IV.2. Consider two systems α and β corresponding to states $\mathcal{S}^\alpha$ and $\mathcal{S}^\beta$, respectively. By defining the two systems on a common probability space, we can assume that for each common link they encounter the same sequences of random numbers uniformly distributed on $[0, 1]$, which in turn decide the success or failure of constructing the link. Fix some some particular sample path Ω and let $\pi^\alpha(\mathcal{S}^\alpha)$ the optimal sequence of links selected by System α starting at state $\mathcal{S}^\alpha$. System β mimics the System α decisions.

Given Ω we distinguish two cases. In Case 1, System α does not select link k until the end of the horizon. Then, System β which mimics System α selects the exact same links and both systems find selected links at the exact same state. Given the coupling we introduced, both systems accumulate the same cost until the end of the horizon.

In Case 2, System α does select link k until the end of the horizon. System β selects link k at exactly the same times. Every time that both Systems select link k System β finds k in a state with one more failure. Note that for the corresponding success probabilities it holds $P^\alpha[\text{Success}|m_k] = \frac{p_k}{1-(1-p_k)^{M_k-m_k}} < \frac{p_k}{1-(1-p_k)^{M_k-m_k-1}} = P^\beta[\text{Success}|m_k+1]$. Success is decided by drawing a random number s uniformly from $[0, 1]$ and declaring success in connecting link k in

System α if $s \leq P^{\alpha}[\text{Success}|m_k]$. Given the coupling we introduced, success in System β is determined by the exact same random number. It follows that for all realizations of s that link k gets connected in System α it also gets connected in System β. For these realizations where link k gets connected in both Systems at the same time it follows that both systems accumulate the same cost until the end of the horizon.

There are however realizations Ω in which at some attempt, say at time t', to connect link k it only gets connected in System β. For these realizations, using the same argument as in Lemma IV.2, the total cost until the end of the horizon in System β is no more than the cost in System α.

Given the assumption we made in the statement of the proposition, note that there are realizations with positive probability in which System α will need to select link k after time t'. For these sample paths System β can idle (at no cost) while System α attempts to connect link k. Since no penalty will be payed by either System idling does not affect the total cost. Thus, the overall cost of System β is strictly less than that of System α.

We have shown that for all sample paths Ω the cost of System β is no more than that of System α and that there are sample paths Ω with positive probability for which the cost of System β is strictly less than that of System α. Taking expectations over all sample paths we establish that $J(\mathcal{S}^{\alpha}) > J(\mathcal{S}^{\beta})$. ■

We next evaluate a policy which is much easier to compute than the optimal DP policy. Specifically, we consider the policy which when it selects a certain link k it continues to try that link until it becomes connected.

Starting from an empty graph $\mathcal{G}_c(0)$, suppose we select link k. We will concentrate on the trials required for connecting k. Let us denote by $\tilde{J}(k(m_k))$ the cost-to-go function of the particular policy we described after m_k failed trials on k. Let $\tilde{J}(k(M_k))$ denote the cost-to-go after link k has been successfully connected. The probability of constructing link k at the m_k-th trial is $p_k^{(m_k)} = \frac{p_k}{1-(1-p_k)^{M_k - m_k + 1}}$, where $0 < m_k \leq M_k$. We have

$$\begin{aligned}
&\tilde{J}(\mathcal{S}(0)) \\
&= c_k + p_k^{(1)}\tilde{J}(k(M_k)) + (1-p_k^{(1)})\tilde{J}(k(1)) \\
&= c_k + p_k^{(1)}\tilde{J}(k(M_k)) + \\
&\quad (1-p_k^{(1)})[c_k + p_k^{(2)}\tilde{J}(k(M_k)) + (1-p_k^{(2)})\tilde{J}(k(2))] \\
&= c_k + c_k(1-p_k^{(1)}) + (p_k^{(1)} + (1-p_k^{(1)})p_k^{(2)})\tilde{J}(k(M_k)) + \\
&\quad (1-p_k^{(1)})(1-p_k^{(2)})\tilde{J}(k(2)) \\
&= \cdots \\
&= c_k(P[Y_k \geq 1] + P[Y_k \geq 2] + \cdots + P[Y_k \geq M_k]) \\
&\quad + (P[Y_k = 1] + \cdots + P[Y_k = M_k])\tilde{J}(k(M_k)) \\
&= c_k E[Y_k] + \tilde{J}(k(M_k)).
\end{aligned}$$

We can now proceed in the same manner and attempt to connect a second link. We repeat this process until we form a spanning tree of $\mathcal{A}$ (so that $\mathcal{G}_c(t)$ becomes strongly connected for some t). It follows that if $\mathcal{T}$ denotes a spanning tree of $\mathcal{A}$ then

$$\tilde{J}(\mathcal{S}(0)) = min_{\mathcal{T}} \sum_{k \in \mathcal{T}} c_k E[Y_k]. \tag{48}$$

The policy we analyzed gives rise to the following algorithm. We note that the MST problem can be solved in a distributed manner by using an algorithm in [12].

Algorithm 3
For every link k present in $\mathcal{A}$, assign $c_k E[Y_k]$ as its weight and compute the Minimum weight Spanning Tree (MST).

We will now establish that this MST-based algorithm is optimal.

Proposition IV.3 *Algorithm 3 is optimal.*

Proof: Consider the optimal policy, say π, obtained by solving the DP problem in (46). The optimal policy constructs a connected graph $\mathcal{G}_c(B)$ at the end of the horizon and, given the assumption in effect in this section, pays no terminal cost. Pick a spanning tree $\mathcal{T}_c$ of $\mathcal{G}_c(B)$ (e.g., uniformly over all spanning trees of $\mathcal{G}_c(B)$) and let $l_1, \ldots, l_{N-1}$ denote the links it contains. For any sample path Ω the cost accumulated by π may contain unsuccessful trials at various links, hence, it is no less than the cost paid to construct $\mathcal{T}_c$. The latter cost is $\sum_{i=1}^{N-1} c_{l_i} y_{l_i}$ where y_{l_i} is the number of trials contained in Ω in order to construct link l_i. Taking expectations over all Ω that result in the same $\mathcal{T}_c$ we obtain a cost of $\sum_{i=1}^{N-1} c_{l_i} E[Y_{l_i}]$. Clearly, this cost is no less than $\tilde{J}(\mathcal{S}(0))$.

Finally, consider an arbitrary sample path of π and let $p_{\mathcal{T}}$ the probability that it leads to a spanning tree $\mathcal{T}$ of $\mathcal{A}$. This probability corresponds to both the sample path and the manner in which a spanning tree is selected from $\mathcal{G}_c(B)$. Taking expectation over all sample paths we obtain an optimal expected cost equal to $\sum_{\mathcal{T}} p_{\mathcal{T}} \sum_{l_i \in \mathcal{T}} c_{l_i} E[Y_{l_i}]$ which is also no less than $\tilde{J}(\mathcal{S}(0))$. This establishes the optimality of the MST-based algorithm. ■

As in the static case, we will investigate the energy cost associated with the MST computation. By employing the distributed algorithm of [12] and following the same analysis as in the static case, a total of $O\big(|\mathcal{A}|+|\mathcal{N}|\log|\mathcal{N}|\big)$ *successful* messages have to be exchanged. Considering the worst case where each message needs the maximum number of trails, the total number of such trials is $O\big((|\mathcal{A}|+|\mathcal{N}|\log|\mathcal{N}|)\cdot M_{\max}\big)$, where $M_{\max} = \max_{k\in\mathcal{A}} M_k$. Therefore, the total energy cost is $O\big((|\mathcal{A}|+|\mathcal{N}|\log|\mathcal{N}|)\cdot M_{\max}\cdot c_{\max}\big) = O(|\mathcal{A}|+|\mathcal{N}|\log|\mathcal{N}|)$, where $c_{\max} = \max_{k\in\mathcal{A}} c_k$ is the maximum energy cost for one trial at any link in the network. Note that the energy cost considered here takes into account the cost of building the topology and applying the load balancing algorithm for every block of B time units. Once the MST is found, the *expected* energy cost of building the topology and applying one iteration of the load balancing Algorithm 2 is $\tilde{J}(\mathcal{S}(0))$.

If we interpret the cost c_k as the energy cost for a trial on link k then the MST-based algorithm minimizes the *expected energy cost to reach connectivity*. We next consider a number of alternative options on the selection of these costs.

1) *Minimum expected interconnectivity time.* In this case we set $c_k = 1$ for all links $k \in \mathcal{A}$. Essentially we

seek to minimize the expected time until we construct a connected graph (a spanning tree of $\mathcal{A}$).

2) *Mixed cases.* As a way to take into account both the energy cost and time we introduce a fixed "set-up" cost c_0 for each trial on any link. Specifically, we set $c_k := c_0 + c_k$ thus penalizing many trials (hence a long time to reach connectivity) even if these trials do not cost a lot in terms of energy.

1) Numerical results: We generate networks by uniformly scattering N nodes on a 10×10 square. We assume that the minimum power needed by a node to reach another node is d^2, where d is their distance. We employ the sigmoid function to relate p_k, the success probability for trial on link k, with d_k, the distance between the two nodes incident to k, i.e., $p_k = 2/(e^{d_k^2/50} + 1)$. We assume that the maximum power of each node is large enough to cover the whole region. The maximum number of trials M_k for each link k is an integer uniformly drawn from $[1, 5]$.

We compare our MST-based algorithm for each cost selection we discussed with the corresponding DP algorithm. All algorithms are run on a computer with Ubuntu-8.04-OS, 2GB of memory, and Intel-XEON-2.00GHz CPU. As expected, our algorithms output the same results as the DP algorithm. However, in terms of running time, our algorithms are much more efficient than DP, which becomes incredibly slow when $n \geq 5$ and runs out of memory almost every time. In contrast, our algorithms usually take less than 1 second for most instances. Table III shows typical numerical experiments, where NA means that the DP algorithm runs out of memory.

TABLE III
MST-BASED ALGORITHM VS. DP.

	MST-based Algorithm		DP Algorithm	
N	Running Time	Result	Running Time	Result
3	<1 sec	47.99	<1 sec	47.99
4	<1 sec	55.93	10.01 sec	55.93
5	<1 sec	127.84	2.45×10^3 sec	127.84
6	<1 sec	149.42	6.21×10^4 sec	NA

C. Energy Efficient Averaging in Dynamic WSNETs with Limited Horizon Length

We now turn our attention to the more challenging case where a terminal penalty is incurred when interconnectivity can not be reached within a *small* block of length B, i.e., $|\mathcal{N}|-1 \leq B < \sum_{k\in\mathcal{A}} M_k$ such that some policy may fail within B time units. Since solving the DP problem is not practical even for reasonably-sized instances we seek suboptimal solutions. To that end, we will leverage the so-called rollout algorithms introduced in [13].

Rollout algorithms offer approximate solutions to discrete optimization problems using procedures that are capable of magnifying the effectiveness of any given heuristic algorithm through sequential application. In particular, in [13], the problem is embedded within a DP framework, and several types of rollout algorithms are introduced, which are related to notions of policy iteration. It has been proved that under certain conditions the rollout algorithm is guaranteed to improve the performance of the original heuristic algorithm.

The key idea of rollout algorithms is to employ one or more suboptimal policies and use their value function in a policy improvement step. The policy obtained through this step is then applied. To make matters concrete consider the decisions induced by the DP iteration in (46), namely, at time t we select link

$$l = \arg \min_{k\in\{\mathcal{A},\emptyset\},\ k\notin\mathcal{E}_c(t)} E\big[c_k + H_{t+1}(\mathcal{S}(t+1))\big], \qquad (49)$$

with the only difference being that instead of using the optimal policy to evaluate the cost-to-go at the next state we use a heuristic/suboptimal policy $\mathcal{H}$ whose cost-to-go is denoted by $H_{t+1}(\mathcal{S}(t+1))$.

As policy $\mathcal{H}$ we can employ one of the MST-based heuristics we developed in Sec. IV-C: Algorithm 3 which minimizes expected cost and its special case with $c_k = 1$ for all k which minimizes the time to reach interconnectivity. More specifically, starting from state $\mathcal{S}(t+1) = (\mathcal{G}_c(t+1), \mathcal{L}(t+1))$ we apply each one of the MST-based heuristics to compute the expected cost of adding links to $\mathcal{G}_c(t+1)$ in order to form a connected subgraph of $\mathcal{A}$. To that end, we can simply (i) concatenate each connected component of $\mathcal{G}_c(t+1)$ into one aggregate node g, (ii) form the graph $\mathcal{G}_{t+1}$ whose node set includes g and all remaining nodes in $\mathcal{N}$ that were not included in g and whose edge set includes all links in $\mathcal{A}$ that are not included in $\mathcal{E}_c(t+1)$, and then (iii) form an MST of $\mathcal{G}_{t+1}$. Note that if $\mathcal{G}_c(t+1)$ contains no cycles then the resulting graph will be a spanning tree of $\mathcal{A}$. Let $\tilde{J}_{\text{cost}}(\mathcal{S}(t+1))$ be the total cost for constructing the MST we just described starting from state $\mathcal{S}(t+1)$ when we use Algorithm 3 (cf. (48)). Let also $\tilde{J}_{\text{time}}(\mathcal{S}(t+1))$ be the corresponding cost when we use Algorithm 3 with c_k's set to one. These costs account for the cost to connect the necessary links but do not account for any potential penalty cost that has to be paid if no connectivity is achieved before the end of horizon. We next describe how such expected penalty costs can be computed.

Suppose that at state $\mathcal{S}(t+1)$ the computed MST selects links $1, \dots, K$, where each one of these links has already been tried $m_1, \dots, m_K$ times, respectively. The time Y_k needed to connect link $k = 1, \dots, K$ has a truncated geometric distribution with parameters $(p_k, M_k - m_k)$. Its z-transform is given by

$$E[z^{Y_k}] = \frac{p_k}{[1-p_k-(1-p_k)^{M_k-m_k+1}]} \sum_{y=1}^{M_k-m_k} (1-p_k)^y z^y. \qquad (50)$$

The z-transform of the total time needed to construct the selected MST is

$$\prod_{k=1}^{K} E[z^{Y_k}] = \prod_{k=1}^{K} \frac{p_k}{[1-p_k-(1-p_k)^{M_k-m_k+1}]} \cdot \prod_{k=1}^{K} \sum_{y=1}^{M_k-m_k} (1-p_k)^y z^y. \qquad (51)$$

The coefficient of z^y, for $y = K, \ldots, \sum_{k=1}^{K}(M_k - m_k)$ in (51) is equal to the probability that it will take y steps to construct the MST. Given that we are at time $t+1$, we can compute the probability, say p^F_{t+1}, that a penalty cost will be paid by summing up all coefficients of z^y for all $y > B - (t+1)$. Thus, the expected penalty cost is equal to Wp^F_{t+1}.

Now, at any time $t+1$ and state $\mathcal{S}(t+1)$ with a connected $\mathcal{G}_c(t+1)$, policy $\mathcal{H}$ does not need to select any links, hence $H_{t+1}(\mathcal{S}(t+1)) = 0$. Finally, at time B, the policy $\mathcal{H}$ has no time to act and for any state $\mathcal{S}(B)$ the cost-to-go is W if $\mathcal{G}_c(B)$ is not connected and 0 otherwise.

Collecting all of the above and letting $H^{\text{cost}}_{t+1}(\cdot)$ denote the cost-to-go of the MST-based policy which uses Algorithm 3 we have

$$H^{\text{cost}}_{t+1}(\mathcal{S}(t+1)) = \begin{cases} 0, & \text{if } \mathcal{G}_c(t+1) \text{ is connected,} \\ W, & \text{if } t+1 = B \text{ and } \mathcal{G}_c(B) \\ & \text{is not connected,} \\ \tilde{J}_{\text{cost}}(\mathcal{S}(t+1)) + Wp^F_{t+1}, & \text{otherwise.} \end{cases} \tag{52}$$

Similarly, we can obtain the cost-to-go $H^{\text{time}}_{t+1}(\cdot)$ of the MST-base policy that uses Algorithm 3 with c_k's set to one. The following algorithm uses both MST-based policies we discussed to obtain an improved policy.

Algorithm 4

1) *Given the current state $\mathcal{S}(t)$ and for each link $k \in \mathcal{A}$ such that $k \notin \mathcal{E}_c(t)$ consider performing one more trial on k. Compute the two possible next states $\mathcal{S}(t+1)$ as in (45) corresponding to a success or failure on link k. For each possible next state $\mathcal{S}(t+1)$:*
   a) *Apply Algorithm 3 and compute $H^{cost}_{t+1}(\mathcal{S}(t+1))$.*
   b) *Apply Algorithm 3 with the c_k's set to one and compute $H^{time}_{t+1}(\mathcal{S}(t+1))$.*
2) *Select link l such that*

$$l = \arg \min_{\substack{k \in \{\mathcal{A}, \emptyset\} \\ k \notin \mathcal{E}_c(t)}} c_k + E\big[\min\{H^{cost}_{t+1}(\mathcal{S}(t+1)), H^{time}_{t+1}(\mathcal{S}(t+1))\}\big], \tag{53}$$

where the expectation is taken with respect to the two possible outcomes for the next state.

As it has been pointed out in [13] rollout algorithms may not necessarily terminate and can cycle. While the principle of optimality precludes cycling when one applies an optimal policy sequentially, with a suboptimal policy it is possible that a sequence of actions can lead to the exact same state and form a cycle. In our setting this is not possible because we are dealing with a finite number of states and no state gets repeated. To see this note that every action (selection of a link) increases the number of times that link has been tried (which is reflected in $\mathcal{L}(t)$). We summarize this argument in the following proposition.

Proposition IV.4 *The rollout algorithm introduced in Algorithm 4 is terminating.*

We note that Algorithm 4 is not a distributed algorithm since the approximated cost-to-go function and the link selection process are based on global network information. For each block of B time units, the total energy cost is associated with connecting the sequence of selected links, which in turn is determined by the network status, the value of B and the terminal penalty cost W.

1) Numerical results: We generate networks by uniformly scattering N nodes on a $a \times a$ square, where $a = 20$. We assume that the minimum power needed by a node to reach another node is d^2, where d is their distance. The maximum power available at each node is λa^2. We employ the sigmoid function to relate p_k, the success probability of trial on link k, with d_k, the distance of nodes incident to k, i.e., $p_k = 2/(e^{d_k^2/10}+1)$. To simplify the calculations we perform, we set $p_i = 0.1$ if $p_i < 0.1$. The maximum number of trials M_k for link k is an integer drawn uniformly from the interval $[1, 10]$. We set the penalty cost as $W = 1000 \times \max_{i,j \in \{1,2,\ldots,n\}} d^2_{(ij)}$.

First, we test the effectiveness of rollout algorithms and how the residual horizon length affects the preference between fast interconnectivity (choosing $H^{\text{time}}_{t+1}(\cdot)$ in (53)) and energy efficiency (choosing $H^{\text{cost}}_{t+1}(\cdot)$ in (53)). We set $\lambda = 2$, so that each node has enough power to cover the whole area (a *dense* network). For various B, we run 20 instances of the algorithm and compute the average number of steps needed and average energy cost required to achieve interconnectivity within the time block of length B. The results are shown in Tab. IV, where SC stands for "success counts," denoting the number of instances reaching interconnectivity. AS stands for "average steps," denoting the average number of steps needed to reach interconnectivity. AE stands for "average energy cost," denoting the average total energy consumed.

TABLE IV
THE EFFECTIVENESS OF THE ROLLOUT ALGORITHM.

B	25			30			35		
N	SC	AS	AE	SC	AS	AE	SC	AS	AE
10	20	12.3	743.1	20	13.7	531.3	20	14.1	528.7
12	20	13.4	947.1	20	14.9	755.6	20	15.4	636.9
14	20	14.2	707.6	20	15.4	444.6	20	17.1	409.7
16	20	15.0	944.5	20	15.3	805.2	20	16.0	383.5

Next we compare the performance of the rollout algorithm with DP in small enough instances so the latter is tractable. The results are in Table V. For DP we report the optimal cost-to-go starting at $t = 0$ and the corresponding running time. For the rollout algorithm we report its performance (average over 50 runs), running time, and "success counts," denoting the number of instances reaching interconnectivity. Clearly, the rollout is much faster and for most instances its performance is close enough to DP. There are however two cases with small B ($B = 5$) when rollout instances will pay a penalty while the optimal policy manages to avoid it; in these cases the rollout performance is much worse than DP.

Next, we assess the scalability of the algorithm. In this scenario, we set $\lambda = 0.3$, which is perhaps closer to practical *sparse* sensor network deployments. We gradually increase the number of nodes and record the time needed to make

TABLE V
THE SUB-OPTIMALITY OF THE ROLLOUT ALGORITHM.

	DP Algorithm		Rollout Algorithm		
B	$\mathcal{J}(0)$	Time	Rollout	Time	SC
3 Node Case					
5	166.69	0.23 sec	1628.1	0.18 sec	49
10	46.68	0.44 sec	48.40	0.19 sec	50
15	46.68	0.65 sec	47.27	0.20 sec	50
4 Node Case					
5	2792.5	341.49 sec	6390.7	0.79 sec	46
10	87.54	668.95 sec	89.23	0.76 sec	50
15	84.10	994.15 sec	86.03	0.77 sec	50

a decision using our rollout algorithms (cf. Eq. (53)). The results are in Table VI. The first column lists the total number of nodes. In the second column, the first number is the number of total potential links while the second number is the number of links in a complete graph with the same number of nodes. The third column corresponds to the running time for each iteration of the rollout algorithm. Notice that although ρ is small in our setting, the network contains many links, even compared to the complete network.

TABLE VI
THE EFFICIENCY OF THE ROLLOUT ALGORITHM IN A SPARSE NETWORK.

n	Link Density	Decision Time (sec)
10	68 / 90	0.32
20	242 / 380	2.97
30	474 / 870	11.66
40	928 / 1560	50.01
50	1296 / 2450	104.21
60	2276 / 3540	447.75

We conclude that our rollout algorithm works quite well in all instances. In almost all randomly generated instances, it produces a suboptimal policy which can reach network interconnectivity within the limited horizon length. We observed that as the limited horizon length B increases the algorithm tends to prefer the minimum cost MST-based policy (i.e., $H_{t+1}^{\text{cost}}(\cdot)$ wins in the minimization appearing in (53)), which is not surprising given that the risk of paying a penalty is reduced. Moreover, as Table VI indicates, our rollout algorithm is scalable and can efficiently output suboptimal solutions within a reasonable running time even for large-scale instances.

V. CONCLUSIONS AND FUTURE WORK

In this paper, we considered the problems of constructing energy efficient topologies in WSNETs which are preferable for running consensus/averaging algorithms. First, for the case of static networks, we considered the problem of constructing a minimum power bidirectional spanning tree. Constructing the tree is an NP-complete problem. We provided a mixed integer programming formulation and several methods for constructing good feasible solutions. Two of our methods can be implemented in a distributed manner and are based on running well known graph algorithms (shortest path, minimum weight spanning tree) on an appropriately designed augmented graph. Computationally, both methods are very efficient and can yield solutions on the order of minutes for moderate instances (of about 50 nodes). Further, an illustrative set of numerical results shows that the solutions we obtain are close to optimal (0%–8% in both dense and sparse network instances).

Second, for the case of dynamic networks, we considered the problem of constructing an interconnected network using minimal energy. We posed the problem as a dynamic programming problem and established a number of structural properties. We studied approximate/suboptimal algorithms that are more accommodating of large-scale instances. To develop these approximate algorithms we first considered a scenario where there is a large enough horizon over which a connected network needs to be built. We established that in such a regime a policy that solves a minimum spanning tree problem is optimal. In the more general case where the horizon is not large enough we developed a rollout algorithm which leverages the MST-based solutions. Through numerical results we demonstrated that the proposed rollout algorithm is effective and efficient-enough to handle large problems. Future work will consider distributed algorithms for the limited horizon case.

REFERENCES

[1] I. C. Paschalidis and B. Li, "On energy optimized averaging in wireless sensor networks," in *Proceedings of the 48th IEEE Conference on Decision and Control*, Shanghai, China, December 2009, pp. 3763–3768.

[2] V. Shnayder, M. Hempstead, B. Chen, G. Allen, and M. Welsh, "Simulating the power consumption of large-scale sensor network applications," in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys 04)*, 2004, pp. 188–200.

[3] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *Joint issue of the IEEE Transactions on Information Theory and IEEE/ACM Transactions on Networking*, vol. 52, no. 6, pp. 2508–2530, 2006.

[4] A. Dimakis, A. Sarwate, and M. Wainwright, "Geographic gossip: Efficient aggregation for sensor networks," in *Proceedings of the 5th International Conference on Information Processing in Sensor Networks (ISPN 2006)*, Nashville, TN, 2006.

[5] M. DeGroot, "Reaching a consensus," *Journal of the American Statistical Association*, vol. 69, no. 345, pp. 118–121, 1974.

[6] J. Tsitsiklis, "Problems in decentralized decision making and computation," Ph.D. dissertation, Department of EECS, MIT, Cambridge, MA, 1984.

[7] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 2953–2958, 2003.

[8] R. Olfati-Saber and R. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1520–1533, 2004.

[9] V. Blondel, J. Hendrickx, A. Olshevsky, and J. Tsitsiklis, "Convergence in multiagent coordination, consensus, and flocking," in *Proceedings of the Joint 44th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC05)*, Seville, Spain, 2005, pp. 2996–3000.

[10] A. Olshevsky and J. Tsitsiklis, "Convergence speed in distributed consensus and averaging," *SIAM Journal on Control and Optimization*, no. 1, pp. 33–55, 2009.

[11] D. Bertsekas and J. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods.* Prentice Hall, 1989.

[12] R. Gallager, P. Humblet, and P. Spira, "A distributed algorithm for minimum-weight spanning trees," *ACM Transactions on Programming Languages and Systems*, vol. 5, no. 1, pp. 66–77, 1983.

[13] D. Bertsekas, J. Tsitsiklis, and C. Wu, "Rollout algorithms for combinatorial optimization," *Journal of Heuristics*, vol. 3, pp. 245–262, 1997.

[14] J. Tsitsiklis, D. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE Trans. Automat. Contr.*, vol. 31, no. 9, pp. 803–812, 1986.

[15] M. Mehyar, D. Spanos, J. Pongsajapan, S. Low, and R. Murray, “Distributed averaging on asynchronous communication networks,” in *Proceedings of the Joint 44th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC05)*, Seville, Spain, 2005, pp. 2996–3000.

[16] C. Moallemi and B. V. Roy, “Consensus propagation,” in *Proceedings of the 19th Neural Information Processing Systems (NIPS 2005)*, Vancouver, Canada, December 2005.

[17] M. Cao, D. Spielman, and A. Morse, “A lower bound on convergence of a distributed network consensus algorithm,” in *Proceedings of the Joint 44th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC05)*, Seville, Spain, 2005.

[18] W. Lai and I. C. Paschalidis, “Optimally balancing energy consumption versus latency in sensor network routing,” *ACM Trans. Sensor Networks*, vol. 4, no. 4, pp. 21:1–21:28, 2008.

[19] A. Das, R. Marks, M. El-Sharkawi, P. Arabshahi, and A. Gray, “Optimization methods for minimum power bidirectional topology construction in wireless networks with sectored antennas,” in *Proc. IEEE GLOBECOM*, Dallas, Texas, November 29 - December 3 2004.

[20] A. E. F. Clementi, P. Penna, and R. Silvestri, “Hardness results for the power range assingment problems in packet radio networks,” in *Proc. 3rd International Workshop on Randomization and Approximation in Computer Science (APPROX 1999)*, *Lecture Notes in Computer Science*, vol. 1671, 1999, pp. 195–208.

[21] J. Wieselthier, G. Nguyen, and A. Ephremides, “On the construction of energy-efficient broadcast and multicast trees in wireless networks,” in *Proceedings of IEEE INFOCOM*, March 2000, pp. 585–594.

[22] D. Yuan, J. Bauer, and D. Haugland, “Minimum-energy broadcast and multicast in wireless networks: An integer programming approach and improved heuristc algorithms,” *Ad Hoc Networks*, vol. 6, no. 5, pp. 696–717, 2008.

[23] K. Fujisawa, M. Kojima, K. Nakata, and M. Yamashita, *SDPA-M (SemiDefinite Programming Algorithm in Matlab) User’s Manual - Version 6.2.0*, http://homepage.mac.com/klabtitech/sdpa-homepage/files/sdpamManual.pdf, 2005.

[24] D. Bertsimas and J. Tsitsiklis, *Introduction to Linear Optimization*. Belmont, MA: Athena Scientific, 1997.

[25] D. Bertsekas and R. Gallager, *Data Networks*, 2nd ed. Prentice-Hall, 1992.

[26] G. Frederickson, “A single source shortest path algorithm for a planar distributed network,” *Lecture Notes in Computer Science*, vol. 182, pp. 143–150, 1985.

[27] L. Jiang and J. Walrand, “Approaching throughput-optimality in a distributed CSMA algorithm: collisions and stability,” in *Proceedings of the 2009 MobiHoc*, 2009, pp. 5–8.

[28] I. C. Paschalidis, W. Lai, and X. Song, “Optimized scheduled multiple access control for wireless sensor networks,” *IEEE Trans. Automat. Contr.*, vol. 54, no. 11, pp. 2573–2585, 2009.

[29] I. C. Paschalidis, W. Lai, and D. Starobinski, “Asymptotically optimal transmission policies for large-scale low-power wireless sensor networks,” *IEEE/ACM Trans. Networking*, vol. 15, no. 1, pp. 105–118, 2007.

Ioannis Ch. Paschalidis (M’96, SM’06) is a Professor at Boston University with appointments in the Department of Electrical and Computer Engineering and the Division of Systems Engineering. He is a Co-Director of the Center for Information and Systems Engineering (CISE) and the Academic Director of the Sensor Network Consortium. He completed his graduate education at the Massachusetts Institute of Technology (MIT) receiving an M.S. (1993) and a Ph.D. (1996), both in Electrical Engineering and Computer Science. In September 1996 he joined Boston University where he has been ever since. He has held visiting appointments with MIT and Columbia University. His current research interests lie in the fields of systems and control, networking, applied probability, optimization, operations research and computational biology.

Binbin Li received the B.Sc. degree in automation and the M.Sc. degree in control science and engineering from Tsinghua University, Beijing, China, in 2004 and 2006, respectively. He is currently a Ph.D. candidate in the Division of Systems Engineering at Boston University. His research interests include optimization and decision theory with main applications in communication and sensor networks.