On Energy Optimized Network Construction for Distributed Averaging in a Dynamic Environment

Ioannis Ch. Paschalidis* Binbin Li**

*Dept. of Electrical & Computer Eng., and Division of Systems Eng., Boston University, Boston, MA 02215, USA, yannis@bu.edu, http://ionia.bu.edu/

**Center for Information & Systems Eng., Boston University, Brookline, MA 02446, USA, libinbin@bu.edu

Abstract: We study the energy costs of running a distributed averaging/consensus algorithm in a dynamic, time-varying wireless sensor network. It has been recently shown in Olshevsky and Tsitsiklis [2009] that running a load-balancing algorithm over a symmetric interconnected network is preferable in terms of convergence time. We formulate the problem of selecting a minimal energy interconnected network as a sequential decision problem and cast it into a Dynamic Programming (DP) framework. This problem is hard to solve, especially when incurring a penalty cost for not reaching interconnectivity within a pre-determined block of time. We first consider the scenario of a large enough time horizon and show that solving the DP is equivalent to constructing a Minimum Spanning Tree (MST), which can be done in a distributed manner. We then consider the scenario of a limited time horizon and employ a “rollout” heuristic that leverages the MST solution and yields efficient solutions for the original DP. Numerical experiments verify the effectiveness and efficiency of our algorithms.

1. INTRODUCTION

Wireless Sensor NETworks (WSNETs) have gained in popularity due to the ease of their deployment in a variety of environments. Growing applications include industrial automation, surveillance, and environmental monitoring. Due to limited battery power, nodes rely on short-range communications and form a multi-hop network to exchange information. Power 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 (Shnayder et al. [2004]). Therefore, network topologies that are energy efficient are always preferred and have attracted significant attention.

A common function with many uses in a WSNET is the computation of an average over the values “sensed” by individual nodes. A distributed computation of the average is preferable for being fault-tolerant (not dependent on a small set of sensor nodes) and self-organizing (network functionality does not require constant supervision).

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 Tsitsiklis [1984], Jadbabaie et al. [2003], Olfati-Saber and Murray [2004], Olshevsky and Tsitsiklis [2009]. In our earlier work (Paschalidis and Li [2009]), we considered averaging in static, time-invariant networks. In this paper, we focus on dynamic, time-varying networks.

The common assumption required to establish convergence of consensus/averaging algorithms is that the network becomes connected infinitely often, e.g., once every $B$ time units for some large enough $B$. For dynamic networks with asymmetric connectivity, Olshevsky and Tsitsiklis [2009] show that the traditional synchronous algorithms yield an exponentially large convergence time in the worst case. Yet, for dynamic networks satisfying a symmetric communication condition, the asynchronous load-balancing algorithm of Bertsekas and Tsitsiklis [1989] can guarantee a convergence time which is polynomial in the number of nodes and $B$. Motivated by this result, we will study how to construct energy efficient network topologies over which we can implement the load-balancing algorithm.

We will 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. 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.

* Research partially supported by the NSF under grant EFRI-0735974, by the DOE under grant DE-FG52-06NA27490, and by the ONR under MURI FY2010 grant N000141010952.
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 Gallager et al. [1983]. 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 use a rollout algorithm that leverages the MST solution.

The rest of this paper is organized as follows. In Sec. 2 we establish some of our notation and review results on averaging/consensus algorithms. In Sec. 3, we introduce the network model, formulate the problem as a DP, and establish a monotonicity property. In Sec. 4.1, we show that for a large horizon \( B \) the DP problem can be solved via an MST algorithm. In Sec. 4.3, we consider the case with a pre-determined \( B \) and introduce the rollout heuristic. Simulation results demonstrate the effectiveness and efficiency of the rollout algorithm. We draw some conclusions in Sec 5.

2. CONSENSUS AND AVERAGING

2.1 The agreement algorithm

The agreement algorithm is an iterative procedure for the solution of the distributed consensus problem. It was introduced in DeGroot [1974] for the time-invariant case, and in Tsitsiklis [1984] for “asynchronous” and time-varying environments.

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

\[
x_i(t+1) = \sum_{j=1}^{n} a_{ij}(t)x_j(t), \quad i = 1, \ldots, n
\]

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 \( \alpha \) such that: (i) \( a_{ii}(t) \geq \alpha \), for all \( i, t \); (ii) \( a_{ij}(t) \in \{0, 1\} \), 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 \( (i, j) \in \mathcal{E}(t) \) if \( a_{ij}(t) > 0 \). Our next assumption requires that for any \( i, j \), node \( i \) will influence (directly or indirectly) the value held by node \( j \) within a finite time length.

Assumption B (Bounded Interconnectivity Times)

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

For the agreement algorithm, the following result is proved in Tsitsiklis et al. [1986] in a more general setting that allows for communication delays under a slightly stronger version of Assumption B.

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

2.2 Dynamic topologies and convergence results

In this section, we consider the case where communications are bidirectional (i.e., if \( (i, j) \in \mathcal{E}(t) \) then \( (j, i) \in \mathcal{E}(t) \)) but the network topology changes dynamically. Averaging algorithms for such a context have been considered previously in Melyari et al. [2005], Moallemi and Roy [2005]. We start by investigating the worst-case performance of the agreement algorithm in a dynamic environment. Unfortunately, Olshevsky and Tsitsiklis [2009] show that its convergence time is not polynomially bounded. For the symmetric dynamic network satisfying Assumption B, Olshevsky and Tsitsiklis [2009] propose a variation of an old load-balancing algorithm from Bertsekas and Tsitsiklis [1989] to tackle the problem. We describe the steps that each node carries out at each time \( t \) in Algorithm 1. We define its neighbors by \( \mathcal{N}_A(t) = \{i \neq A : (A, i) \in \mathcal{E}(t)\} \).

Algorithm 1

For a node \( A \) in the network, if \( \mathcal{N}_A(t) \) is empty, node \( A \) does nothing at time \( t \); else, node \( A \) carries out the following steps.

- Node \( A \) broadcasts its current value \( x_A^{(0)} \) to all of its neighboring nodes (every \( j \) with \( j \in \mathcal{N}_A(t) \)).
- Node \( A \) finds a neighboring node \( B \) with the smallest value: \( x_B = \min\{x_j : j \in \mathcal{N}_A(t)\} \). If \( x_A \leq x_B \), then node \( A \) does nothing further at this step. If \( x_B < x_A \), then node \( A \) makes an offer of \( (x_A - x_B) / 2 \) to \( B \).
- If Node \( A \) 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 \( x_A \) by adding the value of the accepted offer.
- If an acceptance arrives for the offer made by node \( A \), node \( A \) updates \( x_A \) by subtracting the value of the offer.

To characterize the rate of convergence and the convergence time, Olshevsky and Tsitsiklis [2009] introduce the following “Lyapunov” function to quantify the distance of the state \( \mathbf{x}(t) \) from the desired limit: \( V(t) = \|\mathbf{x}(t) - \frac{1}{2} \sum_{i=1}^{n} x_i(0)1\|_2^2 \), where \( 1 \) is the vector of all ones. It is shown that \( V(t) \) is a monotonically non-increasing function of \( t \) when performing Algorithm 1. Given a sequence of graphs \( \mathcal{G}(t) \) on \( n \) nodes, and an initial vector \( \mathbf{x}(0) \), the convergence time is defined as: \( T_{\mathcal{G}(t)}(\mathbf{x}(0), \epsilon) = \min\{t | V(\tau) \leq \epsilon V(0), \forall \tau \geq t \} \).

The (worst case) convergence time, \( T_n(B, \epsilon) \), is defined as the maximum value of \( T_{\mathcal{G}(t)}(\mathbf{x}(0), \epsilon) \), over all initial conditions \( x(0) \), and all graph sequences \( \mathcal{G}(\cdot) \) on \( \mathcal{N} \) that satisfy Assumption B for that particular \( B \). Olshevsky and Tsitsiklis [2009] show that there exists a constant \( c > 0 \) such that for every \( n \) and \( \epsilon > 0 \), \( V((k+1)B)) \leq (1 - 1/\epsilon)^k V(kB) \), i.e., \( T_n(B, \epsilon) \leq \epsilon Bn^3 \log\frac{1}{\epsilon} \).
3. PROBLEM FORMULATION

In this section we first present a model for the WSNET and then formulate link selection as a DP problem.

3.1 Network model

Consider a set of nodes \( \mathcal{N} = \{1, 2, \ldots, n\} \) scattered in some area. Every node has some power level, say \( P_1, P_2, \ldots, P_n \), which determines its potential neighboring nodes. We let \( P_{ij} \) denote the minimum transmit power needed by node \( i \) to reach node \( j \). As an example, we can assume \( P_{ij} = d_{ij}^\alpha \), where \( d_{ij} \) is the Euclidean distance between nodes \( i \) and \( j \), and \( \alpha \) is the channel loss exponent. Let \( \mathcal{A} \) denote the set of potential bidirectional links between the nodes, that is, \( \mathcal{A} = \{(i, j) \mid P_i \geq P_j, P_j \geq P_i\} \).

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 \( \mathcal{A}_c \). We assume that the graph \( (\mathcal{N}, \mathcal{A}_c) \) is strongly connected.

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

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

(2)

The expected number of trials needed to successfully construct this link is:

\[
E[Y_k] = \frac{1 - (M_k + 1)(1-p_k)^{M_k} + M_k(1-p_k)^{M_k+1}}{p_k(1-(1-p_k)^{M_k})}. \tag{3}
\]

For every trial on link \( k \), we let \( c_k \geq 0 \) be the energy cost for each node, which depends on the power level needed to support the link. The expected cost to successfully construct this link is:

\[
E[C_k] = c_k E[Y_k].
\]

Once messages have been successfully exchanged on link \( k \), 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 1).

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):

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

(4)

This is equivalent to a geometric distribution with parameter \( (p_k, M_k - m_k) \). The following Lemma is immediate.

Lemma 2. \( P(Y_k = m_k + 1|Y_k > m_k) \) is a monotonically increasing function of \( m_k \).

3.2 Dynamic programming formulation

As we have seen in Sec. 2.2, to guarantee convergence of the load balancing algorithm we need to enforce Assumption B. The convergence time was shown to be polynomial in \( B \) and \( n \). In this section we formulate the problem of efficiently enforcing Assumption B, either by minimizing the time until the graph becomes strongly connected or by minimizing the energy cost of doing so.

As before, let \( \mathcal{G}(t) = (\mathcal{N}, \mathcal{E}(t)) \) 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 will treat these graphs as undirected and we will say that \( \mathcal{G}_c(t) \) is strongly connected if the corresponding 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.

Before we present the DP formulation let us define the “state” of the network at time \( t \), consisting of two parts. One part is the graph \( \mathcal{G}_c(t) \), which includes all successfully constructed links up to \( t \). The other part contains information on links that have been attempted but not constructed (those that have not been tried can be viewed as having been tried 0 times). We define the state of these links by \( \mathcal{Z}(t) = \{k(m_k)\}_{k=1}^{\mathcal{G}_c(t)} \), 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 \( \mathcal{S}(t) = (\mathcal{G}_c(t), \mathcal{Z}(t)) \).

Recall that for each link \( k \) the number of trials until a success follows a truncated geometric distribution \( (p_k, M_k) \). Given \( m_k \) failures the probability of the residual number of trials until a success follows a truncated geometric distribution \( (p_k, M_k - m_k) \). The success probability of the next trial on link \( k \) is \( p_k \). Given \( m_k \) failures the p.m.f. of the residual number of trials until a success follows a truncated geometric distribution \( (p_k, M_k - m_k) \). The success probability of the next trial on link \( k \) is \( p_k \). 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:

\[
\mathcal{S}(t+1) = (\mathcal{G}_c(t+1), \mathcal{Z}(t+1)) = \begin{cases} (\mathcal{G}_c(t) \cup \{k\}, \mathcal{Z}(t) \setminus k(m_k)), & \text{w.p. } \mathbb{P}_k, \\ (\mathcal{G}_c(t), \mathcal{Z}(t) \cup (k(m_k) \cup k(m_k+1))), & \text{w.p. } 1 - \mathbb{P}_k, \end{cases}
\]  

where \( \mathbb{P}_k = p_k/(1 - (1-p_k)^{M_k-m_k}) \).

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:

\[
J_t(\mathcal{S}(t)) = \min_{k \in \mathcal{A} \setminus \mathcal{A}(t)} \mathbb{E}[c_k + J_{t+1}(\mathcal{S}(t+1))],
\]  

(6)

where \( J_t(\mathcal{S}(t)) \) is the optimal cost-to-go (or value) function at state \( \mathcal{S}(t) \). We make the convention \( c_0 = 0 \).
The boundary conditions are $J_t(S(t)) = 0$ if $G_c(t)$ is strongly connected. We note that as written, (6) 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 $G_c(t)$ has already been achieved. The horizon (block length) in (6) is equal to $B$ and we impose a terminal cost at the end of the horizon equal to
\[ J_B(B) = \begin{cases} 0, & \text{if } G_c(B) \text{ is strongly connected}, \\ W, & \text{otherwise}, \end{cases} \]
where $W \gg 1$ is a large enough penalty.

The following lemma holds (the proof is omitted).

**Lemma 3.** It holds that $J_t(S(t)) \leq J_t(S^\beta(t))$ for all $t$ and $S^\alpha(t) = (G_c^\alpha(t), S^\alpha(t)), S^\beta(t) = (G_c^\beta(t), S^\beta(t))$ such that $G_c^\alpha(t) \subseteq G_c^\beta(t)$ and $S^\alpha(t)$ coincides with $S^\beta(t)$ for all links $k \notin E_c^\beta(t)$.

An immediate corollary of this lemma is that $G_c(B)$ should be a tree; otherwise we can simply construct only links in a spanning tree of $G_c(B)$ which will reduce the overall cost.

### 4. ENERGY EFFICIENT AVERAGING IN DYNAMIC WSNETS

The DP formulation is insightful but it does not lead to practical (efficient) algorithms. In this section we develop such algorithms.

#### 4.1 Large enough horizon length

We start with the simpler case where $B$ is large enough so that for any feasible policy interconnectivity is guaranteed before $B$, hence, we pay no terminal cost w.p.1. Since we do not need to keep track of time, we simplify the notation for the states and the cost-to-go function by writing $\mathcal{S} = (G_c, L)$ and $J(\mathcal{S})$, respectively. The following proposition holds (the proof is omitted).

**Proposition 4.** Suppose we are at some state $\mathcal{S}^\alpha = (G_c^\alpha, S^\alpha)$ and link $k \notin E_c^\alpha$. Assume that there is a positive probability that link $k$ participates in the connected graph at the end of the horizon, i.e., $k \in E_c^\alpha$ at time $B$. Consider some other state $\mathcal{S}^\beta = (G_c^\beta, S^\beta)$ such that $G_c^\alpha = G_c^\beta$ and $S^\beta = S^\alpha \setminus k(m_k) \cup k(m_k + 1)$. Then, $J(\mathcal{S}^\beta) > J(\mathcal{S}^\alpha)$. We next evaluate a policy which which is much easier to compute and implement 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 $G_c(0)$, 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 $k$. Note that this notation does not not keep track of time as we are now facing a long enough horizon so that a connected graph is achieved before we reach $B$. Let $\tilde{J}(k(M_k))$ denote the cost-to-go after link $k$ has been successfully connected. We have $\tilde{J}(\mathcal{S}(0)) = c_k E[Y_k] + \tilde{J}(k(M_k))$.

We can now proceed in the same manner from the state where link $k$ is connected and attempt to connect a second link. We can repeat this process until we have formed a spanning tree of $\mathcal{S}$ (so that $G_c(t)$ becomes strongly connected for some $t$). It follows that if $\mathcal{S}$ denotes a spanning tree of $\mathcal{S}$ then
\[ \tilde{J}(\mathcal{S}(0)) = \min_{\mathcal{S} \in \mathcal{S}} \sum_{k \in \mathcal{S}} c_k E[Y_k]. \]

The policy we analyzed gives rise to the following algorithm. We note that the MST problem can be solved in polynomial time. A distributed computation of the MST is possible by using an algorithm in Gallager et al. [1983].

**Algorithm 2**

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

We establish that this MST-based algorithm is optimal.

**Proposition 5.** Algorithm 2 is optimal.

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

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).

#### 4.2 Numerical results for large enough horizon length

We generate networks by uniformly scattering $n$ nodes on a $10 \times 10$ 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 programmed with Matlab 7.6.0 (R2008a) and 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 in every instance. 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 1 shows some typical numerical experiments, where NA means that the DP algorithm runs out of memory.

#### 4.3 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...
Table 1. MST-based Algorithm vs. DP.

<table>
<thead>
<tr>
<th>n</th>
<th>Running Time</th>
<th>Result</th>
<th>Running Time</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>&lt; 1 sec</td>
<td>47.99 secs</td>
<td>&lt; 1 sec</td>
<td>47.99 secs</td>
</tr>
<tr>
<td>4</td>
<td>&lt; 1 sec</td>
<td>59.93 secs</td>
<td>10.01 secs</td>
<td>NA</td>
</tr>
<tr>
<td>5</td>
<td>&lt; 1 sec</td>
<td>127.84 secs</td>
<td>2.45×10^2 secs</td>
<td>127.84 secs</td>
</tr>
<tr>
<td>6</td>
<td>&lt; 1 sec</td>
<td>249.32 secs</td>
<td>6.21×10^2 secs</td>
<td>NA</td>
</tr>
</tbody>
</table>

Table 1. MST-based Algorithm vs. DP.

MST-based Algorithm vs. DP Algorithm

<table>
<thead>
<tr>
<th>Running Time</th>
<th>Result</th>
<th>Running Time</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>&lt; 1 sec</td>
<td>47.99 secs</td>
<td>&lt; 1 sec</td>
</tr>
<tr>
<td>4</td>
<td>&lt; 1 sec</td>
<td>59.93 secs</td>
<td>10.01 secs</td>
</tr>
<tr>
<td>5</td>
<td>&lt; 1 sec</td>
<td>127.84 secs</td>
<td>2.45×10^2 secs</td>
</tr>
<tr>
<td>6</td>
<td>&lt; 1 sec</td>
<td>249.32 secs</td>
<td>6.21×10^2 secs</td>
</tr>
</tbody>
</table>

B. As policy we can employ one of the MST-based heuristics we developed in Sec. 4.1: Algorithm 2 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 $\mathcal{S}(t+1) = (\mathcal{G}(t+1), \mathcal{I}(t+1))$ we apply each one of the MST-based heuristics to compute the expected cost of adding links to $\mathcal{G}(t+1)$ in order to form a connected subgraph of $\mathcal{G}$. To that end, we can simply concatenate all nodes that are already connected in $\mathcal{G}(t+1)$ 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 $\mathcal{G}$ that are not included in $\mathcal{G}(t+1)$, and (iii) form an MST of $\mathcal{G}_{t+1}$. To prove our results we use the expected total cost for constructing the MST $\mathcal{G}(t+1)$ when we use Algorithm 2 (cf. (8)). Let also $J_{\text{time}}(\mathcal{G}(t+1))$ be the corresponding cost when we use Algorithm 2 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 $\mathcal{G}(t+1)$ the computed MST selects links $1, \ldots, K$, where each one of these links has already been tried $m_1, \ldots, m_K$ times, respectively. The time $Y_k$ needed to connect link $k = 1, \ldots, K$ has a truncated geometric distribution with parameters $(p_k, M_k - m_k)$. Its $z$-transform is given by $E[z^{Y_k}] = \frac{p_k}{1 - p_k (1 - p_k)^{M_k - m_k}}$. The $z$-transform of the total time needed to construct the selected MST is $\prod_{k=1}^K E[z^{Y_k}]$. The coefficient of $z^y$, for $y = K, \ldots, \sum_{k=1}^K (M_k - m_k)$, 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 $p_{t+1}$, 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 $W p_{t+1}$.

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

Collecting all of the above and letting $H_{\text{cost}}(\cdot)$ be the cost-to-go of the MST-based policy using Alg. 2 we have

$$H_{\text{cost}}(\mathcal{G}(t+1)) = \begin{cases} 0, & \text{if } \mathcal{G}(t+1) \text{ is connected}, \\ W, & \text{if } t + 1 = B \text{ and } \mathcal{G}(t) \text{ is not connected}, \\ J_{\text{cost}}(\mathcal{G}(t+1)) + W p_{t+1}, & \text{otherwise}. \end{cases}$$

We can also obtain the cost-to-go $H_{\text{time}}(\cdot)$ of the MST-base policy that uses Algorithm 2 with $c_k$’s set to one. The following algorithm uses both MST-based policies we discussed to obtain an improved policy.

**Algorithm 3**

(i) Given the current state $\mathcal{G}(t)$ and for each link $k \in \mathcal{G}$ such that $k \notin \mathcal{G}(t)$ consider performing one more trial on $k$. Compute the two possible next states $\mathcal{G}(t+1)$ as in (5) corresponding to a success or failure on link $k$. For each possible next state $\mathcal{G}(t+1)$: (a) apply Algorithm 2 and compute $H_{\text{cost}}(\mathcal{G}(t+1))$; (b) apply Algorithm 2 with the $c_k$’s set to one and compute $H_{\text{time}}(\mathcal{G}(t+1))$.

(ii) Select link $l$ such that

$$l = \arg \min_{k \in \{1, \ldots, K\} \setminus \mathcal{G}(t)} \left[ c_k + \min_{k \notin \mathcal{G}(t)} \left( H_{\text{cost}}(\mathcal{G}(t+1)), \right. \right.$$}

$$
\left. H_{\text{time}}(\mathcal{G}(t+1)) \right] , \tag{10}
\]$$

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

We end this section with the following proposition:

**Proposition 6.** The rollout algorithm introduced in Algorithm 3 is terminating.

4.4 Numerical results for limited horizon length

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 $p a^2$. 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., $p_k = 2/(e^{d_k/10} + 1)$. To simplify the calculations we perform we set $p_k = 0.1$ if $p_k < 0.1$. The maximum number of trials $M_k$ for link $k$ is an integer drawn uniformly from $[1, 10]$. We set the penalty cost as $W = 1000 \times \max_{j \in \{1, 2, \ldots, n\}} d_{ij}^2$ for all programs are tested under the same environment as stated in Sec. 4.2.

First, we test the effectiveness of rollout algorithms and how the residual horizon length affects the preference between fast interconnectivity (choosing $H_{\text{time}}(\cdot)$ in (10))
and energy efficiency (choosing $H^{\text{cost}}_t$ in (10)). We set $\rho = 2$, so that each node has enough power to cover the whole area, which yields 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 numerical results are shown in Tab. 2, where SC stands for “success counts,” denoting the number of instances (out of 20) reaching interconnectivity within $B$ steps. 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 by all nodes in the network selected by the algorithm.

Table 2. The effectiveness of the rollout algorithm.

<table>
<thead>
<tr>
<th>$B$</th>
<th>25</th>
<th>30</th>
<th>35</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>SC</td>
<td>AS</td>
<td>AE</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>12.3</td>
<td>743.1</td>
</tr>
<tr>
<td>12</td>
<td>20</td>
<td>13.4</td>
<td>947.1</td>
</tr>
<tr>
<td>14</td>
<td>20</td>
<td>14.2</td>
<td>707.6</td>
</tr>
<tr>
<td>16</td>
<td>20</td>
<td>15.0</td>
<td>944.5</td>
</tr>
</tbody>
</table>

Next we compare the performance of the rollout algorithm with DP in small enough instances so the latter is tractable. The results are shown in Table 3. 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 (out of 50) reaching interconnectivity within $B$ steps. Clearly, the rollout is much faster and for most instances its performance is close enough to DP. There are however two cases with small enough $B$ ($B = 5$) 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.

Table 3. The sub-optimality of the rollout algorithm.

<table>
<thead>
<tr>
<th>$B$</th>
<th>$f(0)$</th>
<th>Time</th>
<th>Rollout</th>
<th>Time</th>
<th>SC</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 Node Case</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>166.69</td>
<td>0.23 secs</td>
<td>1628.1</td>
<td>0.18 sec</td>
<td>49</td>
</tr>
<tr>
<td>10</td>
<td>46.68</td>
<td>0.44 secs</td>
<td>48.40</td>
<td>0.19 sec</td>
<td>50</td>
</tr>
<tr>
<td>15</td>
<td>46.68</td>
<td>0.65 secs</td>
<td>47.27</td>
<td>0.20 sec</td>
<td>50</td>
</tr>
<tr>
<td>4 Node Case</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>126.57</td>
<td>13.19 secs</td>
<td>156.66</td>
<td>0.61 sec</td>
<td>50</td>
</tr>
<tr>
<td>10</td>
<td>85.97</td>
<td>25.83 secs</td>
<td>92.20</td>
<td>0.73 sec</td>
<td>50</td>
</tr>
<tr>
<td>15</td>
<td>85.97</td>
<td>38.42 secs</td>
<td>86.54</td>
<td>0.60 sec</td>
<td>50</td>
</tr>
</tbody>
</table>

5. CONCLUSIONS AND FUTURE WORK

We considered the problem of constructing an interconnected network using minimal energy in a wireless sensor network setting. Such a network structure is preferable for running a consensus/averaging algorithm in terms of convergence rate and time. 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


