UNIVERSITY
OF TRENTO

DEPARTMENT OF INFORMATION AND COMMUNICATION TECHNOLOGY

38050 Povo – Trento (Italy), Via Sommarive 14
http://www.dit.unitn.it

MAXIMIZING SENSING COVERAGE IN WIRELESS SENSOR NETWORKS THROUGH OPTIMAL SCATTERING OF WAKE-UP TIMES

Luigi Palopoli, Roberto Passerone, Amy L. Murphyz, Gian Pietro Picco, and Alessandro Giusti

July 2007

Technical Report DIT-07-048

Maximizing Sensing Coverage in Wireless Sensor Networks Through Optimal Scattering of Wake-up Times

Luigi Palopoli*, Roberto Passerone*, Amy L. Murphy‡, Gian Pietro Picco*, and Alessandro Giusti†
* Dept. of Information and Communication Technology, University of Trento, Italy
{luigi.palopoli,roberto.passerone,gianpietro.picco}@unitn.it
‡ Fondazione Bruno Kessler—IRST, Trento, Italy, murphy@itc.it
† Dept. of Electronics and Information, Politecnico di Milano, Italy, giusti@elet.polimi.it

Abstract

Wireless sensor networks (WSNs) often rely on duty-cycling, alternating periods of low-power stand-by with others where sensing, computation, and communication are performed. Duty-cycling brings substantial energy savings, but may complicate the WSN design. The effectiveness of a node in performing its task (e.g., sensing events occurring in an area) is affected by its wake-up schedule. Random schedules lead to deployments that are either ineffective (e.g., insufficient sensing coverage) or inefficient (e.g., areas covered by multiple nodes simultaneously awake).

In this paper, we focus on the problem of scattering *the nodes' wake-up times* optimally*, to achieve maximal coverage of a given area. In previous work [5], we presented a decentralized protocol that improves significantly over random wake-up schedules. Instead, here we provide a centralized* optimal *solution that complements the work in [5] by identifying the theoretical upper bound to distributed protocols. Moreover, the modeling framework we present, based on integer programming techniques, is general enough to encompass alternative formulations of the problem. These include the inverse problem of determining the optimal schedule given a desired coverage, as well as other problems based on constraints other than coverage (e.g., latency of data dissemination).*

1. Introduction

The operation of a wireless sensor network (WSNs) node is typically characterized by (long) periods of inactivity, spent in a low-power stand-by state, interleaved with (short) periods where the expected sensing, computation, and communication duties are carried out. This duty-cycling is frequently used to reduce energy consumption to a minimum, and therefore maximize the lifetime of the network at large.

Nevertheless, the performance of the WSN application critically depends on the quality of the duty-cycling schedule, which ensures that the right nodes are active at the right time. A random schedule may lead to highly inefficient deployments. For instance, consider Figure 1, where three nodes cover, with overlaps, a certain area. In this case, having node 2 and node 3 *not* be active at the same time is beneficial, since they share a large portion of the area. A random schedule may miss considerable opportunities for optimization.

In this paper, our objective is to maximize the area monitored by sensors. In contrast with the existing work concisely surveyed in Section 2, we do not achieve this goal by controlling the placement of nodes, which is an input in our case. Instead, we operate on the nodes' wake-up times: by properly *scattering* them, we take advantage of the overlap among the nodes' sensing range. Indeed, since the same point in the target area may be monitored by multiple nodes, it is possible to switch off certain nodes without affecting the coverage, as long as the remaining active nodes cover the area of interest.

In [5] we presented a decentralized *wake-up scattering* protocol, whose effectiveness and practical relevance we demonstrated in several common WSN scenarios, including the coverage problem above. However, although we showed that our protocol yields significant improvements over random schedules, an open question remained about whether further improvements can be obtained.

This paper provides an answer to this question, by giving a means to compute the *optimal* schedule of wake-up times that maximizes the area covered by a set of sensor nodes. Our technique is inherently centralized, as it assumes global knowledge. Therefore, although it can be applied directly in constrained deployments, it is most useful as a theoretical upper bound against which to evaluate distributed solutions. This and other assumptions are discussed in Section 3, along with an overview of our approach, which is

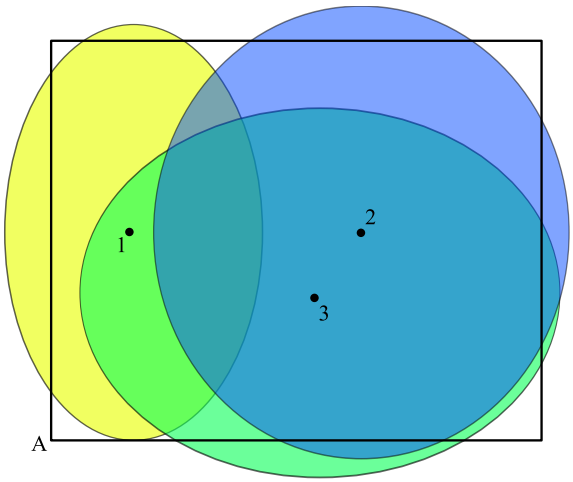

Figure 1. Example of topology where scattering matters

described in detail in Section 4.1 and 4.2. Section 5 reports quantitative results derived using our approach in various WSN scenarios, as well as comparisons with the decentralized, on-line approach described in [5].

The contribution we put forth in this paper is not limited to the coverage problem. The modeling framework we present, using integer programming techniques, is general enough to encompass alternative formulations, including the inverse problem of determining the optimal schedule given a desired coverage, as well as other problems based on constraints other than coverage (e.g., latency of data dissemination). Therefore, it provides a theoretical foundation for tackling problems that are similarly affected by duty-cycling. These aspects are discussed in Section 6, before the concluding remarks in Section 7.

2. Related work

The introduction of duty cycling to increase system lifetime focused research attention on mitigating its negative effects on system behavior both at the MAC layer and above.

At the MAC layer, a variety of solutions either synchronize or spread awake intervals. For example, SMAC [15] and TMAC [11] yield solutions where all nodes are awake during the same, short interval, enabling communication. Applying this approach for sensing leads to an undesirable system with coverage only for the single, short synchronized awake interval. Other approaches such as LMAC [12] and ZMAC [9] spread out communication, assigning TDMA-like slots. However, unlike our approach, a node is assigned only a single slot, and the scheduling is generally random.

Above the MAC, achieving the best spatial sensor placement has been studied [8]. Solution techniques include integer programming [3], greedy heuristics [1, 6, 7] and virtual force methods [16]. These works complement our temporal distribution and can be applied in parallel to achieve further coverage improvements.

In temporal scattering, our distributed wake-up scattering protocol [5], and a similar but more complex solution [2] have been formulated and evaluated. As these solutions are based on local information exchange, they neither achieve the globally optimal scattering schedule, nor can they perform well in environments where the communication range is smaller than the sensing range. In this paper, our previous scattering protocol serves as a point of comparison.

Finally, the notion of applying linear programming to find optimal solutions in WSNs has been applied to a similar problem of multiple-point coverage [14], however, the goal of that work is to select a set of sensors, not to schedule the awake times of all sensors.

3. System model and assumptions

System model We are given[1] a *target area* A to be monitored using a set of *sensor nodes* $\mathcal{N}$. For each node $n \in \mathcal{N}$, we are given the *position* of the node in the target area A as a pair of coordinates (x_n, y_n), as well as its *sensing range* n_A, i.e., the area that is directly monitored by n. In other words, we are given the complete topology of the problem. An example was already shown in the introduction in Figure 1.

The system operates according to a periodic schedule. The period, called the *epoch* and denoted by E, is further discretized into a finite number of *slots*, in which each node may be either active or asleep. In our problem, the lifetime of the system is fixed a priori, and determined by the *awake* or *activation interval*, i.e., the interval during which a node is awake. We assume that the awake interval, potentially spanning multiple slots, is the same across all nodes, and that each node is woken up only once per epoch. Alternative formulations with different awake intervals or multiple activations per epoch are possible at the expense of a more cumbersome treatment, and are therefore omitted here.

A discretized epoch simplifies the optimization task which can be expressed as a slot assignment problem, and formulated as an integer (in fact, boolean) linear program [13]. The granularity of the discretization, i.e., the duration of a slot, obviously affects (adversely) the achievable quality of the final result, but also lowers the complexity of solving the linear program. A trade-off can therefore

[1] For simplicity, we describe the problem in a two-dimensional space: the extension to a three-dimensional space is straightforward.

be established between optimality and the time it takes to compute the solution.

Assumptions In our formulation we make a number of simplifying assumptions. For instance, we assume that the sensitivity of a node in its sensing range is constant, and that the boundaries of the sensing range are sharp. These limitations can be addressed by considering different (e.g., concentric) sensing areas for each node, and by weighing them differently in the optimization constraints. This is, in fact, what we do to overcome the problem of sharp boundaries: since the contribution to a constraint of some area is weighted by the area itself, small errors in the determination of the sensing range will have a low impact on the final result. On the other hand, accurate models would not only make the solution substantially harder, but would likely still be affected by large errors. This justifies our use of a simpler model for the sake of tractability.

Another important assumption, intrinsic to the centralized way we solve the problem, is that the topology of the system should not evolve in time. This condition may not be satisfied by systems that require the mobility of the sensor nodes. A topology change may also be due to the failure of a node. In these cases, a new optimum must be recomputed with the new topology. Similarly, we consider the topology of the system exclusively in terms of sensing, regardless of connectivity. However, we believe these are reasonable assumptions, as our main target is not to use our results directly in real deployments, rather to derive a theoretical optimum useful as a baseline against which to compare distributed, on-line, practically usable protocols such as the one we described in [5].

4. Solution Algorithm

The objective of scattering the wake up times of the nodes is to maximize the area covered during the epoch. At any time t, the set of nodes that are awake at that time defines a covered area $S(t)$, which is equal to the area of the union of the sensing areas of all awake nodes.[2] Our objective is therefore to maximize the integral over the epoch of $S(t)$.

Our method to solve the optimization problem goes through two steps, which are described in detail in Section 4.1 and 4.2, respectively.

1. We consider the problem of determining a *finite* number of regions in the target area that, without loss of accuracy, can be used to model the topology of the system and the area overlaps. We refer to this as the *spatial partitioning* problem.

[2]Note that this is different from taking the sum of the areas of the awake nodes, since overlaps would be counted multiple times.

2. We express the scheduling problem by setting up an integer linear program, with coverage constraints for each node and for each region found in the previous step.

The optimization problem is then solved using standard linear programming and branch and bound techniques, and yields as a result the optimal schedule for the given objective function and constraints. In the following sections we discuss our solution in detail, and present experimental results for a number of different setups.

4.1. Spatial partitioning

The optimization problem can be set up as a linear program by enforcing a certain level of coverage on the target area A, through the use of appropriate constraints (see Section 4.2). This approach is feasible if A is discretized into a finite number of regions, to avoid generating constraints for each of the infinitely many points in A. One method is to discretize A on a regular grid. However, a regular discretization may either be inaccurate, when it is too coarse, or inefficient, when it reaches the minimum granularity to ensure correctness. To avoid these problems, we use an adaptive, non-regular finite discretization that retains all the information of the original topology, while achieving the largest possible granularity.

To do that, we identify regions in the target area that are invariant relative to node coverage. That is, we look for regions such that for any two points in the same region, if one is covered, then the other is covered, and vice-versa. In this case, it is sufficient to consider only one point per region, or equivalently consider the region as a whole. This idea can be expressed as follows. Let $r : A \rightarrow 2^{\mathcal{N}}$ be the function that for each point $p \in A$ associates the set of nodes for which p is in range. We partition the points of A according to the following relation $\sim$:

$$p_1 \sim p_2 \iff r(p_1) = r(p_2).$$

The relation $\sim$ is an equivalence relation since it is reflexive, transitive and symmetric. Hence, the equivalence classes of $\sim$ completely partition the area A into a set of non-overlapping regions. By definition, every point in a region is covered by exactly the same set of nodes, and therefore regions are invariant, as required. Given that there is a one-to-one mapping, we sometimes make no distinction between a region ρ and the set of nodes $r(\rho)$ that cover it. Note also that each region need not necessarily be connected, even when the sensing areas of the nodes are connected and convex. Thus, this kind of spatial partitioning cannot be obtained using a simple regular discretization.

The number of regions determined by our equivalence relation is finite, and equal to $2^{|\mathcal{N}|}$ in the worst case. In

practice, the range of the function r, which determines the number of equivalence classes for a given problem, may be considerably smaller than $2^{|\mathcal{N}|}$, depending on the geometry of the problem. For the case of rectangular sensing areas, that we address below, the number of regions is order $O(|\mathcal{N}|^2)$, as can be determined by analyzing the complexity of Algorithm 1 which computes their number and areas. Similar bounds can be derived for other geometries, such as circles. The number of regions will be even lower when the target area is large, and the sensing area of distant nodes do not overlap, a situation that is common in any moderate size deployment. However, a systematic study of the size of the equivalence relation as a function of node density is still part of our future work.

Determining the invariant regions may involve non-trivial geometrical computations, especially if the sensing area of the nodes have non-regular shapes. In this paper, we present an algorithm for the simplified case in which the sensing area of a node is a rectangle. This allows us to easily compute intersections by performing only sums and differences, and avoid dealing with higher order polynomial as in the case of round areas. This model can be trivially extended to sensing areas composed of a *union* of rectangles, and can therefore be made arbitrarily accurate, including the case of unconnected sensing areas (e.g., blind spots). Experiments in locationing also show that, despite its simplicity, this approach can lead to reliable results [10].

The partitioning algorithm works by scanning the target area horizontally, left to right, stopping at every vertical boundary of a node's range, as shown in Figure 2. At every

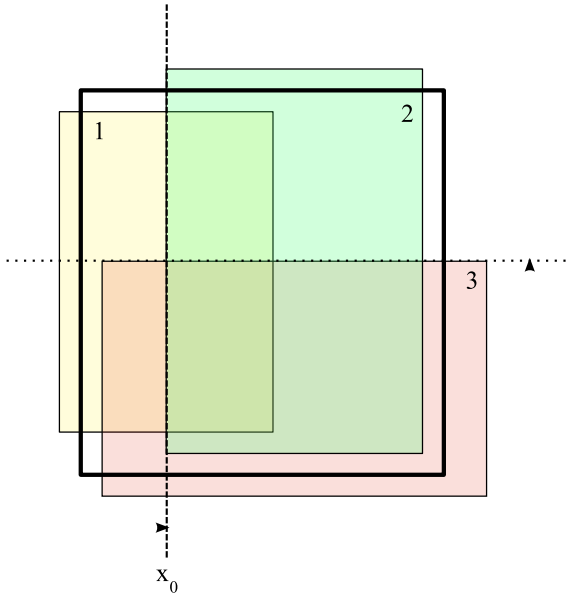

Figure 2. Rectangular topology with scanning sequence

stop, the algorithm performs a vertical scan, from bottom to top, that computes the regions found at that horizontal position, and updates their area. This is sufficient, since the computation of the optimal schedule discussed in Section 4.2 does not require the shape of the invariant regions, but only their area and corresponding set of covering nodes. Each region can conveniently be identified by its set of covering nodes, as discussed above. For example, in Figure 2, the vertical scan at x_0 finds a sequence of regions corresponding to the sets of nodes:

$$\{3\}, \{2,3\}, \{1,2,3\}, \{1,2\}, \{2\}. \quad (1)$$

Areas are accumulated incrementally, and computed up to the following stop in the scanning sequence.

The procedure is shown in detail in Algorithm 1. To

Algorithm 1 Spatial partitioning algorithm

```
1:  h_scan = list of vertical boundaries of the nodes ordered by their horizontal position
2:  v_scan = ∅
3:  R = ∅
4:  for i = h_scan.begin() to h_scan.end() do
5:    if i is left boundary of node n then
6:      Insert the horizontal boundary of n in v_scan ordered by their vertical position
7:    else // i is the right boundary of node n
8:      Erase the horizontal boundary of n from v_scan
9:    end if
10:   h_extent = horizontal extent to next stop
11:   node_set = ∅
12:   for j = v_scan.begin() to v_scan.end() do
13:     if j is bottom boundary of node n then
14:       Insert n in node_set
15:     else // j is the top boundary of node n
16:       Remove n from node_set
17:     end if
18:     v_extent = vertical extent to next stop
19:     area = h_extent · v_extent
20:     R = R ∪ {node_set}
21:     node_set.area = node_set.area + area
22:   end for
23: end for
```

perform the horizontal scan, we first build a sequence *h_scan* of all the vertical boundaries of the nodes, ordered by their horizontal position (line 1). We then loop through this sequence (line 4) and stop at every element. To perform the vertical scan we also build a sequence, denoted *v_scan*, by inserting the horizontal boundaries of the nodes that intersect the scan at the current horizontal position in the order of their vertical position. For efficiency, this sequence is constructed incrementally: it is initialized to empty (line 2) and at every step of the horizontal scan we insert or remove

Region	Nodes	Area	Region	Nodes	Area
ρ_0	$\emptyset$	56	ρ_4	$\{ 1, 3 \}$	96
ρ_1	$\{ 1 \}$	144	ρ_5	$\{ 2 \}$	244
ρ_2	$\{ 1, 2 \}$	140	ρ_6	$\{ 2, 3 \}$	272
ρ_3	$\{ 1, 2, 3 \}$	160	ρ_7	$\{ 3 \}$	112

Table 1. Regions for topology of Figure 2

the horizontal boundaries of the node at that position, according to whether we are entering (from the left) or exiting (to the right) the sensing area of the node (lines 6 and 8). In the example of Figure 2, the vertical scan at x_0 would already have the ordered entries

$$(3_{bot}, 1_{bot}, 3_{top}, 1_{top}).$$

At x_0, since we are entering node 2, we add 2_{bot} and 2_{top} in the correct order, to obtain

$$(3_{bot}, 2_{bot}, 1_{bot}, 3_{top}, 1_{top}, 2_{top}). \quad (2)$$

The vertical scan starts with the loop at line 12. During this phase, we keep track of the set of nodes *node_set* for which the current point is in range. This set is updated at every step of the scan according to whether we are entering the node's range from the bottom (line 14) or exiting it from the top (line 16). For example, scanning the sequence 2 we obtain the regions of sequence 1. After computing the area of the *node_set*, we add it to the set $\mathcal{R}$ (line 20), which was previously initialized to empty at the beginning of the procedure (line 3). If the region was already present in the set, we do not add a new element but simply update its area (line 21). Since regions can span several horizontal positions, we use for the set $\mathcal{R}$ an ordered data structure that provides efficient insertion and retrieval.

The result of applying Algorithm 1 to the topology of Figure 2 is shown in Table 1, where each region is associated with its covering nodes and its area. Note that the area covered by the nodes outside the target area A is ignored.

4.2. Computation of the optimal schedule

Once we have partitioned the target area into non-overlapping regions, we compute the optimal schedule of the wake-up times for the nodes by setting up an optimization program. The input from the spatial partitioning is:

- the set $\mathcal{N}$ of nodes,
- the set $\mathcal{R}$ of regions to be monitored,
- a function $r : \mathcal{R} \rightarrow 2^{\mathcal{N}}$ associating to each region $\rho \in \mathcal{R}$ the subset $r(\rho)$ of nodes that cover the region,

$$\begin{array}{ll} & \max \sum_{k=0}^{L-1} \sum_{\rho \in \mathcal{R}} C_{k,\rho} w(\rho) \\ \forall n \in \mathcal{N} & \sum_{k=0}^{L-1} x_{n,k} = n_s \\ \forall n \in \mathcal{N} & \sum_{k=0}^{L-1} s_{n,k} = 2 \\ \forall n \in \mathcal{N}\ \forall k \in [0, L-1] & s_{n,k} \geq x_{n,k} - x_{n,((k+1)\%L)} \\ \forall n \in \mathcal{N}\ \forall k \in [0, L-1] & s_{n,k} \geq x_{n,((k+1)\%L)} - x_{n,k} \\ \forall \rho \in \mathcal{R}\ \forall k \in [0, L-1] & \sum_{n \in r(\rho)} x_{n,k} \geq C_{\rho,k} \\ \forall \rho \in \mathcal{R}\ \forall k \in [0, L-1] & \forall n \in r(\rho)\ C_{\rho,k} \geq x_{n,k} \end{array}$$

Figure 3. The coverage problem: maximize the coverage assuming that each node is active once per epoch and for an interval equal to n_s slots.

- a function $w : \mathcal{R} \rightarrow \mathbb{R}_+$ that associates to each region its area.

To manage the temporal aspect we discretize the problem by dividing the period E into *slots* of length Δ. For simplicity, we assume that E is an integer multiple of Δ, so that there are exactly $L = \frac{E}{\Delta}$ slots in each epoch. The slots are numbered from 0 to $L - 1$.

Given that each node is awake only for a fixed interval, the output of the optimization is a *schedule* that defines the slot at which the node wakes up. This schedule is computed to maximize the total area covered by the nodes during the epoch. The network lifetime is fixed and defined as first node failure. Note that the ability to change the schedule across epochs could potentially give a better solution. However, we defer an investigation on this possibility to our future work.

As an example, let us go back to the problem in Figure 2, whose output is in Table 1. Assume that we only have two slots in the epoch and that each node can be active for at most one slot. If we allocate all nodes in one slot, we cover a total area of $\sum_{i=1}^{7} w(\rho_i)$ = 1168, where $w(\rho_i)$ is the area of region ρ_i. We can improve on this coverage by moving one node to the other slot. For instance, if we move node 1, the coverage for the first slot is equal to the area of the regions covered by nodes 2 and 3, which is $\sum_{i=2}^{7} w\rho_i = 1024$. For the second slot, it is the area of the regions covered by node 1, which is $\sum_{i=1}^{4} w\rho_i = 540$, for a total of 1564. If we move node 2, we have a total covered area of 1468, and if we move node 3 we obtain a total of 1696. The optimal choice is therefore to move node 3.

For the cases in which the active interval is larger than one slot, we allow the activation interval to wrap around the epoch. For instance, if the active interval is 2 slots and if the number of slots per epoch is 4 (numbered from 0 to 3), a node can be activated at slot 3 of one epoch and remain active in slot 0 of the subsequent epoch.

4.3. The coverage problem

In this section we show how the problem can be set up as a boolean linear program (BLP), which can be solved by standard commercial and open source tools.

To model the schedule, we introduce a set of variables that say if a node is active in a slot or not. Formally, this is a binary variable

$$x_{n,\,k} = \begin{cases} 1 & \text{if node } n \text{ is awake during slot } k \text{ of each epoch} \\ 0 & \text{otherwise.} \end{cases}$$

From the schedule we compute the coverage by introducing a set of variables that says if a region is covered in a slot or not. Formally, this is another binary variable

$$C_{\rho,\,k} = \begin{cases} 1 & \text{region } \rho \text{ is covered during slot } k \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Using the $C_{\rho,\,k}$ variables and the w area function, we can compute the covered area as

$$S = \sum_{k=0}^{L-1} \sum_{\rho \in \mathcal{R}} C_{k,\,\rho} w(\rho).$$

The optimization problem is then expressed as

$$\begin{aligned} &\max S \\ &\text{subj. to} < \text{each node is active for } n_s \text{ slots} > \quad \text{(I)} \\ &\qquad\quad < \text{each node is activated once} > \quad \text{(II)} \end{aligned}$$

These conceptual constraints must be expressed in terms of the schedule variable $x_{n,\,k}$. Formalizing the limit on the number of active slots, constraint (I) can be straightforwardly expressed as:

$$\forall n \in \mathcal{N} \ \sum_{k=0}^{L-1} x_{n,k} = n_s,$$

which counts and limits to n_s the number of slots a node is active. For constraint (II), we must ensure that a node's schedule includes only a single activation (transition from 0 to 1) and deactivation (transition from 1 to 0). The number of activations and deactivations is formally expressed by considering adjacent slots, as in:

$$W_a(n) = \sum_{k=0}^{L-1} |x_{n,\,k} - x_{n,\,(k+1)\,\%\,L}|, \tag{4}$$

which uses the $\%$, or modulo, notation to account for the fact that the schedule repeats in subsequent epochs, and the first slot actually follows the last. To guarantee only a single activation and deactivation the number of transitions is restricted to two, as in:

$$\forall n \in \mathcal{N},\ W_a(n) = 2.$$

We linearize the activation and deactivation counting with the binary variables $s_{n,\,k} \in \{0,\,1\}$ such that $\forall n \in \mathcal{N}\ \forall k \in [0, L-1]$:

$$\begin{aligned} s_{n,\,k} &\geq x_{n,\,k} - x_{n,\,((k+1)\,\%\,L)} \\ s_{n,\,k} &\geq x_{n,\,((k+1)\,\%\,L)} - x_{n,\,k} \end{aligned} \tag{5}$$

which together handle the absolute value computation of $W_a(n)$, and limit the number of transitions with:

$$\forall n \in \mathcal{N}\ W_a(n) = \sum_{k=0}^{L-1} s_{n,\,k} = 2. \tag{6}$$

By requiring these constraints we can prove that $s_{n,\,k} = |x_{n,\,k} - x_{n,\,(k+1)\,\%\,L}|$. In fact, any activation or deactivation, i.e., when $|x_{n,\,k} - x_{n,\,(k+1)\,\%\,L}| = 1$, forces $s_{n,\,k}$ to 1 due to Equation (5). On the other hand, constraint (I) forces at least two transitions during an epoch, from off to on, and on to off. Therefore, $\sum_{k=0}^{L-1} s_{n,\,k} \geq 2$. However, the constraint in Equation (6) requires that this be *equal* to 2, hence it follows that $s_{n,\,k} = 0$ when $|x_{n,\,k} - x_{n,\,(k+1)\,\%\,L}| = 0$.

Finally, we must show how to compute the coverage variable, $C_{n,\,\rho}$, whose value is uniquely determined by the schedule $x_{n,k}$. This is done with the additional constraints:

$$\begin{aligned} &\forall \rho \in \mathcal{R},\ \forall k \in [0, L-1],\ \sum_{n \in r(\rho)} x_{n,\,k} \geq C_{\rho,\,k}; \quad (a) \\ &\forall \rho \in \mathcal{R},\ \forall k \in [0, L-1], \forall n \in r(\rho),\ C_{\rho,\,k} \geq x_{n,\,k}; \quad (b) \end{aligned} \tag{7}$$

where (a) guarantees that a region ρ is not considered covered if none of the nodes n covering it are active during slot k. Specifically, constraint (a) forces $C_{\rho,\,k} = 0$. Conversely, by constraint (b), if at least one covering node is active, the region is considered covered, and $C_{\rho,\,k} = 1$.

Figure 3 summarizes the optimization problem. In this case, all the decision variables $x_{n,k}$, $s_{n,k}$ and $C_{\rho,\,k}$ can only take binary values.

5. Simulation results

To verify the effectiveness and the practical relevance of our approach, we performed an extensive set of simulations comparing the performance achieved by our optimal algorithm against both randomly generated schedules as well as those obtained by our previously published distributed scattering algorithm [5]. The primary performance metric is the covered area averaged over the epoch. Specifically, if $S(k)$ is the area covered during slot k, the covered area is $(1/L)\sum_{k=0}^{L-1} S(k)$. Our initial evaluation uses GLPK [4], an open source linear programming kit, to compute the optimal schedule. Typical runs with less than 50 nodes completed in a few hours. We are currently investigating more efficient implementations using other ILP solvers.

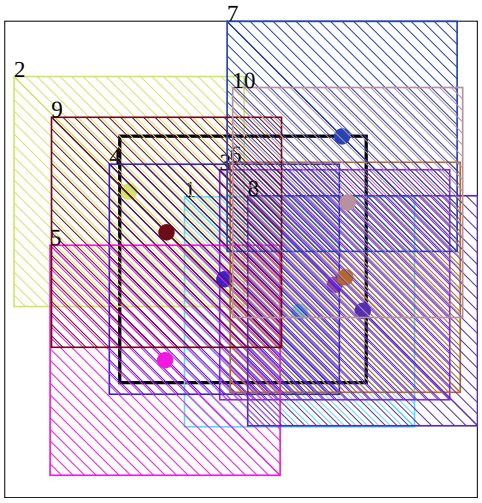

Figure 4. Spatial configuration of the 10 node scenario. The thick line delimits the target area to be monitored. For each node a colored square shows its sensing area.

Our evaluation considers two primary scenarios, outlined in Table 2, with different sizes and amount of spatial overlap. In all cases, we considered square sensing areas and, for the distributed algorithm, circular communication ranges. The first scenario considers 10 nodes randomly deployed in a 150×150 target area. The edge of a single sensing square is 140 and the spatial partitioning algorithm identifies 62 unique regions. Figure 4 visualizes this scenario. The second scenario randomly places 30 nodes in a 600×600 target area with a sensing square edge of 212. In this case, the number of regions is 308.

To compare against random, we generated 3000 random initial schedules, calculating the average coverage achieved. Because the distributed solutions requires communication to share wake-up times among neighboring nodes, the size of the communication radius greatly affects its ability to estimate overlapping sensing regions and scatter awake time with respect to nearby nodes. Therefore, we evaluated three communication ranges for each scenario, representing low, medium, and high degrees of connectivity among nodes. Additionally, the scattering protocol does not change the wake-up order among nodes, and is affected by the initial random wake-up configuration. Therefore, we report results that average 10 different initializations.

Table 3 presents the results, grouping the two scenarios. As optimal and random results do not depend on the communication radius, only one value is reported for each. Instead, the scattering protocol's performance relative to optimal increases with larger communication ranges. The third column reports the *coverable area*, or the area that could be covered if all sensors are active throughout the entire epoch. This is a loose upper bound, which even the optimal duty-cycling schedule cannot attain. To allow easy comparison, we report the random and scattered schedules as a percentage of the optimal solution.

When considering the random schedules, it is worth noting that they are heavily affected by the spatial density of the nodes. Indeed, if the node are distant from one another to a point where their sensing areas do not overlap, all schedules are equivalent. On the contrary, if the sensing areas of the nodes overlap heavily, the performance of the random schedule can be substantially lower than the optimal (an average value around 75% of the optimum may correspond to a worst case coverage far below the 50% of coverable area). The performance of the scattering algorithm is consistently better than the random schedule, even in cases with moderate connectivity. As communication increases, its performance similarly increases, achieving up to 93% of the optimum.

6. Extensions and future directions

The formulation proposed in Section 4.2 can be applied to a much wider class of problems than coverage. For example, we can easily handle the inverse problem: given a required coverage, find a schedule that *maximizes the lifetime* of the network. Since we have defined lifetime as first node failure, this goal can be achieved by minimizing the maximum energy consumed by any node in one epoch. In this case we allow nodes to activate/deactivate multiple times. For power consumption, we adopt a model in which a node incurs an energy cost proportional to its active interval paying an additional price for each activation. A conceptual formulation of the problem is the following

$$\min \max_{n \in \mathcal{N}} \alpha I(n) + \beta W_a(n)$$
$$\text{subj. to} < \text{the assigned area is covered} > \quad \text{(I)}$$

The cost function is the element-wise maximum of a vector of functions. Each element of the vector, $\alpha I(n) + \beta W_a(n)$, is related to a node n. In particular: α is (an estimation of) the energy consumed by a node during a slot of its active interval (i.e., for sensing, computation and communication), $I(n)$ is the active interval, β is the energy spent for activation, $W_a(n)$ is the number of times n is activated/deactivated during the epoch.

The vector notation in the cost function can be expressed in scalar form by introducing an additional decision variable

Number of Nodes	Target Area	Sensing Area	Awake Interval n_s	Number of Slots E
10	150×150	140×140	3	15
30	600×600	212×212	3	12

Table 2. Key parameters of our simulation scenarios

Number of Nodes	Communication Range	Coverable Area	Coverage of the Optimal Schedule	Average Coverage of Random Schedule (percent of optimal)	Average Coverage of the Scattered Schedule (percent of optimal)
10	45	22500.02	18030.741	76.909%	84.63 %
	60				84.51 %
	99				93.25%
30	53	337329.98	254354.11	79.18%	80.8%
	106				87.12%
	130				90.58%

Table 3. Simulation results for the different scenarios. The last two columns report the ratio between the average coverage respectively obtained by a random schedule and by the scattering algorithm and the coverage achieved by the optimal algorithm.

μ constrained as follows:

$$\begin{aligned} &\min \mu \\ &\forall n \in \mathcal{N},\ \mu \geq \alpha I(n) + \beta W_a(n) \end{aligned} \tag{8}$$

The function $I(n)$ is given by:

$$I(n) = \sum_{k=0}^{L-1} x_{n,\,k}.$$

Function W_a can be computed as shown in Equation (4). Once again, it is possible to linearize the constraints in Equation (8) by introducing the binary variables $s_{n,\,k}$. Since we want to allow for multiple activations, which would be inhibited by (6), we impose Equation (5) as the only constraint on $s_{n,\,k}$. Even without Equation (6), when a slot has no activation or deactivation the value of $s_{n,\,k}$ is guaranteed to be 0 in the final solution because the optimal solution must also be minimal.

For Constraint (I) we require that for each slot the sum of the areas of the covered regions is greater than a given value A_0:

$$\forall k \in [0, L-1], \sum_{\rho \in \mathcal{R}} w(\rho) C_{\rho,\,k} \geq A_0$$

where the $C_{\rho,k}$ variable has been introduced in Equation (3) and computed in Equation (7). The complete formulation for the problem is shown in Figure 5. In this case we have both binary decision variables ($x_{n,\,k}$ and $s_{n,\,k}$) and a real decision variable (μ). Therefore the problem is a mixed boolean linear program (MBLP).

$$\begin{array}{ll} & \min \mu \\ \forall n \in \mathcal{N} & \mu \geq \alpha \left(\sum_{k=0}^{L-1} x_{n,\,k} \right) + \\ & +\beta \left(\sum_{k=0}^{L-1} s_{n,\,k} \right) \\ \forall n \in \mathcal{N}\ \forall k \in [0, L-2] & s_{n,\,k} \geq x_{n,\,k} - x_{n,\,((k+1)\%L)} \\ \forall n \in \mathcal{N}\ \forall k \in [0, L-2] & s_{n,\,k} \geq x_{n,\,((k+1)\%L)} - x_{n,\,k} \\ \forall k \in [0, L-1] & \sum_{\rho \in \mathcal{R}} w(\rho) C_{n,\,k} \geq A_0 \\ \forall \rho \in \mathcal{R}\, \forall k \in [0, L-1] & \sum_{n \in r(\rho)} x(n,\, k) \geq C_{\rho,\,k} \\ \forall \rho \in \mathcal{R}\, \forall k \in [0, L-1] & \forall n \in r(\rho)\ C_{\rho,\,k} \geq x_{n,\,k} \end{array}$$

Figure 5. The lifetime problem: minimize the maximum power consumption incurred in one epoch requiring that an area at least equal to A_0 be always covered.

Additionally, we will explore adding constraints to manage latency in delivery of information to a sink node. This is naturally expressed as additional constraints on the awake times between parent and children nodes in a distribution tree. Finally, we will refine the power consumption model to realistically explore the option of multiple awake intervals inside the epoch.

7. Conclusions

This paper presented evaluated a modeling framework for scheduling node wake-up times to achieve the maximal coverage in a WSN. Our model is solvable with linear programming, and the results, in comparison to a simple, distributed solution, demonstrate that the optimal solution achieves significant improvement. Our interpretation of this result is that additional gains can be achieved with the distributed solution, for example by including aspects such as signal strength to estimate the sensing overlap among nodes. We also intend to evaluate other optimizations that can be explored within our specification framework, for example, calculating the maximum lifetime for a given coverage.

References

[1] N. Bulusu, D. Estrin, and J. Heidemann. Adaptive beacon placement. In *Proceedings of the 21st International Conference on Distributed Computing Systems (ICDCS)*, pages 489–498, Phoenix, AZ, USA, April 2001.

[2] Q. Cao, T. Abdelzaher, T. He, and J. Stankovic. Towards optimal sleep scheduling in sensor networks for rare-event detection. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks (IPSN)*, Los Angeles, CA, USA, April 2005.

[3] K. Chakrabarty, S. S. Iyengar, H. Qi, and E. Cho. Grid coverage for surveillance and target location in distributed sensor networks. *IEEE Transactions on Computers*, 51(12):1448–1453, 2002.

[4] F. S. Foundation. The gnu linear programming kit. http://www.gnu.org/software/glpk/.

[5] A. Giusti, A. Murphy, and G. Picco. Decentralized Scattering of Wake-up Times in Wireless Sensor Networks. In *Proceedings of the 4th European Conference on Wireless Sensor Networks (EWSN)*, LNCS 4373, pages 245–260. Springer, January 2007.

[6] A. Howard, M. J. Mataric, and G. S. Sukhatme. An incremental self-deployment algorithm for mobile sensor networks. *Auton. Robots*, 13(2):113–126, 2002.

[7] A. Howard, M. J. Mataric, and G. S. Sukhatme. Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. In *Proceedings of the 7th International Symposium on Distributed Autonomous Robotic Systems (DARS)*, June 2002.

[8] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. Srivastava. Coverage problems in wireless ad-hoc sensor networks. In *Proceedings of 20th IEEE INFOCOM*, pages 1380–1387, Anchorage, Alaska, USA, April 2001.

[9] I. Rhee, A. Warrier, M. Aia, and J. Min. Z-MAC: A hybrid MAC for wireless sensor networks. In *Proceedings of the 3rd ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 90–101, San Diego, CA, USA, Nov. 2005.

[10] A. Savvides, H. Park, and M. Srivastava. The bits and flops of the N-hop multilateration primitive for node localization problems. In *First ACM International Workshop on Wireless Sensor Networks and Application (WSNA)*, pages 112–121, Atlanta, GA, 2002.

[11] T. van Dam and K. Langendoen. An adaptive energy-efficient MAC protocol for wireless sensor networks. In *Proceedings of the 1st ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 171–180, Los Angeles, CA, USA, Nov. 2003.

[12] L. van Hoesel and P. Havinga. A lightweight medium access protocol (LMAC) for wireless sensor networks. In *Proceedings of 1st International Workshop on Networked Sensing Systems*, Tokyo, Japan, June 2004.

[13] W. L. Winston. *Introduction to Mathematical Programming: Applications and Algorithms*. Duxbury Press, 1995.

[14] S. Yang, F. Dai, M. Cardei, and J. Wu. On multiple point coverage in wireless sensor networks. In *Proceedings of the 2nd IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, Washington D.C., USA, November 2005.

[15] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient MAC protocol for wireless sensor networks. In *Proceedings of the 21st IEEE INFOCOM*, pages 1567–1576, June 2002.

[16] Y. Zou and K. Chakrabarty. Sensor deployment and target localization in distributed sensor networks. *Transactions on Embedded Computing Systems*, 3(1):61–91, 2004.