Distributed Critical Location Coverage in Wireless Sensor Networks with Lifetime Constraint

Changlei Liu and Guohong Cao
Department of Computer Science & Engineering
The Pennsylvania State University
University Park, PA 16802
E-mail: {chaliu, gcao}@cse.psu.edu

***Abstract*—In many surveillance scenarios, there are some known critical locations where the events of concern are expected to occur. A common goal in such applications is to use sensors to monitor these critical locations with sufficient quality of surveillance within a designated period. However, with limited sensing resources, the coverage and lifetime requirement may not be satisfied at the same time. Thus, sometimes the sensor needs to reduce its duty cycle in order to satisfy the stringent lifetime constraint. In this paper, we model the critical location coverage problem using a point coverage model with the objective of scheduling sensors to maximize the event detection probability while meeting the network lifetime requirement. We show that this problem is NP-hard and propose a distributed algorithm with a provable approximation ratio of 0.5. Extensive simulations show that the proposed distributed algorithm outperforms the extensions of several state-of-the-art schemes with a significant margin while preserving the network lifetime requirement.**

I. Introduction

Field surveillance is an important application of wireless sensor network that plays a major role in many civilian and military scenarios. In many cases, the surveillance application has stringent lifetime requirement, which demands the network to operate for at least a designated period of time. In a dynamic environment, this requirement is not easy to be continually fulfilled since the sensor density may drop over time and the mission duration may change unexpectedly. Therefore, the preplanned strategy may not work, instead, the sensor activity should be dynamically adjusted according to the lifetime requirements and network conditions.

The tradeoff between sensor coverage and network lifetime has been studied extensively in the sensor network literature [1], [2], [3], [4], [5], [6], [7], [8]. However, most existing works target for only one direction, that is, to elongate the network lifetime while meeting the coverage requirement. The other direction, which trades the coverage for network lifetime, is often overlooked. In this paper, we consider the lifetime as a constraint but optimize the coverage. Similarly, the work in [9] also considers the network lifetime as a constraint, but it focuses on the area coverage instead of location coverage. In addition, the distributed algorithm proposed in [9] is essentially a heuristic with no performance guarantee, but here we present a distributed algorithm with a provable constant approximation ratio.

This work was supported in part by the US National Science Foundation (NSF) under grant number CNS-0916171.

In this paper, we study the critical location coverage problem, with each location modeled as a point[1]. Thus, our work falls into the category of point coverage [10], [11]. Our goal is to schedule the sensor activity within the designated period to monitor the critical locations as many as possible and as long as possible. Since the number of sensors may not be enough to cover all critical locations for the whole period, we define a new metric called *effective coverage time* (or coverage, for short) to quantify the performance of coverage. In particular, the effective coverage time of a critical location is defined as the product of the importance factor of the location and the length of time during which the location is covered. Then our objective becomes how to schedule the sensor's on-period to maximize the total effective coverage time, calculated as the sum of individual effective coverage time at all critical locations. When the events uniformly occur in time at each critical location, it can be proved that the effective coverage time can be directly translated to the weighted event detection probability, which is of central importance in the surveillance applications.

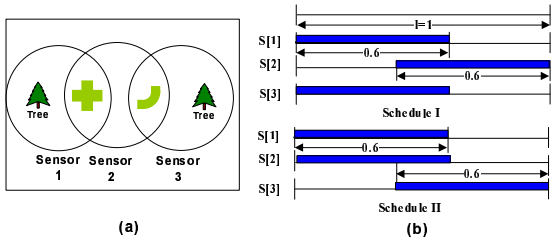

Fig. 1. A surveillance example, where three sensors monitor four critical locations.

Fig. 1 uses a simplified example to illustrate how to calculate the total effective coverage time given the network and sensor schedule. In Fig. 1(a), three sensors are required to monitor four targets (i.e., a curve, a crossing, and two trees) for 10 hours. The importance factor of the tree, curve, and crossing is assumed to be 1, 2, and 4, respectively. Since each sensor battery can only last for 6 hours, the existing sensors are not able to provide full coverage while meeting the network lifetime requirement. Most existing coverage algorithms in such a case would activate the three sensors simultaneously for 6 hours, without considering the network lifetime constraint. However, we trade the coverage for lifetime

[1]We will use the term critical location and critical point interchangeably throughout the paper.

by dividing the mission duration of 10 hours into ten cycles, and within each cycle, each sensor is active for at most $6/10 = 0.6$ hour. Then the scheduling issue becomes how to place the "on" period (no greater than 0.6 hour) in each cycle so that the total effective coverage of the critical locations are maximized. Fig. 1(b) shows two possible solutions. In Schedule I, the two trees are monitored for 0.6 hour; the curve and the crossing are monitored for the full cycle of 1 hour. Thus, the total effective coverage of Schedule I is $1\times0.6+1\times0.6+4\times1+2\times1 = 7.2$. Similarly, the total effective coverage of Schedule II is $1\times0.6+1\times0.6+4\times0.6+2\times1 = 5.6$.

At first sight, the above two schedules make no difference because each sensor covers the same target for the same amount of time in different schedules. However, Schedule I performs better because the important target (i.e., the crossing) in Schedule I is collaboratively monitored for a longer time than that in Schedule II. In fact, when a large number of events occur uniformly in time at each critical location and each location is assigned an importance factor as the weight, the optimization of the effective coverage time can be directly translated to the maximization of the weighted event detection probability[2]. However, it is non-trivial to find the optimal schedule. Although the example in Fig. 1 can be easily solved (schedule I is actually the optimal solution), in a complex network where hundreds of sensors are arbitrarily deployed, we need a systematic way to address the problem.

The organization of the paper is as follows. Section II formulates the critical location coverage problem with lifetime constraint, and proves it is NP hard. Section III proposes an approximation algorithm that works in a distributed manner. Section IV analyzes the algorithm and proves its constant approximation ratio. Performance evaluations are done in Section V. Finally, Section VI summarizes the related works and Section VII concludes the paper.

II. Problem Formulation

In a surveillance application, when the sensor density is not enough to monitor the critical locations throughout the designated lifetime, the coverage has to be traded for lifetime. In this paper, we study this tradeoff by formulating an optimization problem (*MaxEct*) with the objective to maximize the total effective coverage time. We assume the event occurrences are uniformly distributed in time and in space at the critical locations. As a result, the optimization of the effective coverage time can be directly translated to the maximization of average weighted event detection probability. To achieve this, we divide the network lifetime L into cycles and turn on each sensor within each cycle for a time period proportional to its battery life. We further designate that the same schedule repeats in each cycle, such that the sleep schedule can be implemented, e.g., using the Power Saving Mode (PSM) of 802.11. Then the purpose of the scheduling is to place the on-periods within each cycle, such that the total effective coverage time can be maximized. The network lifetime is defined to be the time period until a sensor runs out of battery, which is widely used in the literature. This definition is motivated by the observation that when batteries are fairly used among sensors, the first sensor death due to depleted battery indicates that more sensors will run out of battery shortly. With these assumptions, the $MaxEct$ problem can be formalized as follows.

Problem MaxEct*: Given a unit-disk graph $G(N, E, P)$ with n sensors, a set of critical locations P whose importance factors are known, the battery life of each sensor $B_i, i = 1...n$, and a designated lifetime L, where $B_i \leq L$, we want to calculate an "on" schedule per cycle for each sensor such that the overall effective coverage time is maximized.*

To compute the total effective coverage time, we need to get the sum of individual effective coverage time at all critical locations. As mentioned before, when the importance factor of each location is given and the schedule of each sensor is known, the effective coverage time can be calculated.

Alternatively, we can compute the total effective coverage time in another way. First define *elementary region* as the minimum region formed by the intersection of a number of sensing disks. For example, Fig. 1 has five elementary regions, two of which are covered by two sensors (the overlapping area) and the other three are covered by a single sensor. Notice that different critical locations belonging to the same elementary region are covered by the same set of sensors, and thus covered for the same amount of time. Thus, we can associate an importance factor with an elementary region by summing up the importance factors of all critical locations in that region. If a region does not contain any critical location, its importance factor is zero. Then, the total effective coverage time can also be calculated by the sum of individual effective coverage time over all elementary regions. Below lists some symbols and notations that will be used in our problem formulation.

- N, E: the node set and the edge set of the graph.
- $s_i, N(i)$: sensor i and the neighbor set of sensor i, i.e., $N(i)$=$\{j \mid$ sensor s_j is the neighbor of sensor $s_i\}$.
- R: the index set of the elementary regions, i.e., R=$\{i \mid$ region i is a elementary region$\}$.
- $R(i)$: the index set of the elementary regions within the sensing disk of sensor i.
- p_i: the ith critical location.
- P: the index set of the overall critical locations.
- $P(i)$: the index set of the critical locations within the elementary region i.
- w_i: the importance factor of critical location i.
- $W(i)$: the accumulated importance factors of the critical locations within elementary region i, i.e., $W(i) = \sum_{j\in P(i)} w_j$.
- T_i: the coverage time of the location p_i, during which p_i is covered by at least one sensor.
- $T(i)$: the coverage time of region i. $T(i) = T_j, \forall j \in P(i)$.
- L, l: L is the network lifetime, and l is the length of the cycle, so there are $\frac{L}{l}$ cycles, assuming $\frac{L}{l}$ is an integer. l should not be too small so that the switch overhead between the on/off state is negligible.

[2]Through simple analysis, it can be seen that the two metrics differ by a constant factor, which is also confirmed by the simulation results in Section V. Due to space limitation, the analysis is omitted.

- B_i, b_i: B_i is the battery life of s_i, and b_i is the length of s_i's on-period per cycle. Since there are $\frac{L}{l}$ cycles, we have $b_i \times \frac{L}{l} \leq B_i$.
- $s_i.start$, $s_i.end$: the start and end time of s_i's on-period within the cycle, where $s_i.end - s_i.start = b_i$.
- C: the total effective coverage time.

With these notations, we can calculate the total effective coverage time as the weighted sum of the individual coverage time over all critical locations, or alternatively, over all elementary regions. The problem can be formalized as follows.

$$\text{Max } C = \sum_{i \in P} w_i \times T_i = \sum_{i \in R} W(i) \times T(i) \quad (1)$$

$$\text{ST: } 0 \leq s_i.start \leq l, i \in N \quad (2)$$

$$s_i.end - s_i.start = b_i, i \in N \quad (3)$$

$$b_i \leq B_i \times \frac{l}{L}, i \in N \quad (4)$$

The objective of the above optimization is to determine the variables $s_i.start$ (or $s_i.end$) to maximize the total effective coverage time subjecting to the constraints (2), (3), (4). In Eqn. (1), the objective is written in two forms. In the first form, the coverage time is summed over all critical locations, where w_i is the importance factor of location i and T_i is the time during which the ith location is covered by *at least* one sensor, which depends on the schedules of all neighboring sensors. In the second form, the sum is taken over all elementary regions, with $W(i)$ as the accumulated importance factor and $T(i)$ as the coverage time of region i. Constraint (2) shows that the on-period may fall on the boundary of the cycle, so $s_i.start$ ranges from 0 to l. Constraint (3) establishes the relationship between $s_i.end, s_i.start$ and the length of its on-period. Constraint (4) shows that the upper bound of each node's on-period within a cycle is proportional to its battery life.

In the reminder of the section, we prove that $MaxEct$ is NP-hard, and then propose an approximation algorithm in the next section.

Theorem 1: The problem $MaxEct$ is NP-hard.

Proof: First, the total effective coverage time is upper bounded by $U = \sum_{p_i \in P} w_i \times |N(p_i)| \times b$, where $|N(p_i)|$ is the number of sensors covering location p_i and $b = \max_{i \in N} b_i$. The upper bound can be achieved when each node is on for b duration per cycle and there is no schedule overlap among the sensors covering the same location.

Second, we will prove the NP complexity of $MaxEct$ by considering a decision version of the problem, i.e., given a surveillance scenario, does there exist any sensor schedule based on which the total coverage can reach the upper bound $U = \sum_{p_i \in P} w_i \times |N(p_i)| \times b$?

Third, we show that the $MaxEct$ decision problem is NP-hard via a reduction from the graph k-coloring problem, which asks whether a given graph G can be colored using k colors such that no two neighboring vertexes have the same colors [12]. Given an instance I of graph k-coloring problem, we can construct an equivalent instance I' of the $MaxEct$ *decision* problem in polynomial time, such that instance I has a solution if and only if instance I' has a solution. To construct instance I', assume the node battery life B and network life L satisfy $\frac{L}{B} = \frac{l}{b} = k$. Then we can divide each cycle into k time slots, with each time slot mapped to a distinct color in instance I. Just as instance I assigns k colors, instance I' allocates one of the k time slots to each node. It can be observed that if there exists a k-coloring scheme where each node is assigned a color different from that of its neighbors', there must exist a corresponding scheduling scheme where no two neighbors are allocated the same time slot. This means that any set of sensors covering the same critical location does not overlap in time, and hence I' has a solution whose total coverage is the upper bound U.

On the other hand, suppose there is a scheduling solution for instance I' where the total coverage is U, we can always find a solution to instance I for the graph k-coloring problem. Since each on-period in the solution of I' may not exactly occupy a time slot, we need to first align the on-period of each node with the closest left time slot. By doing this, no additional overlap is introduced and the aligned schedule is still the solution for instance I'. After that, to construct a solution to instance I simply becomes equivalent to "color" each vertex using one of the k time slots, such that no neighboring vertexes have the same colors. In this regard, the solution of instance I' produces a corresponding solution for instance I.

To sum up, G can be k-colored if and only if there is a scheduling scheme for G whose total coverage reaches the upper bound U, which means that the graph coloring problem can be reduced to the $MaxEct$ decision problem in polynomial time. Since the graph k-coloring problem is NP-hard, the $MaxEct$ decision problem is NP-hard. On the other hand, the $MaxEct$ optimization problem is at least as hard as its decision problem, and thus is also NP-hard. ■

III. A Distributed Approximation Algorithm

To solve $MaxEct$, we need to select a subset of sensors which covers the given critical locations and determine their corresponding schedules. Since this problem is NP-hard, we propose a distributed approximation algorithm, which involves only local computation and local message exchanges.

A. Algorithm Description

To simplify the presentation, we divide the algorithm execution into rounds. Note that in reality the algorithm can be executed asynchronously and the concept of rounds can be removed. In our algorithm, during each round, a subset of sensors called *local optimal sensors*, are elected, and their schedules are determined. The local optimal sensors are those sensors that can provide the maximum additional effective coverage time among their neighborhood. Although the algorithm is only based on the computation of local optimal, its performance is within a constant factor of the global optimal, as demonstrated in Section IV. However, several questions need to be answered. For example, how to let each sensor know whether it is the local optimal? How to update the sensor schedule when a sensor is labeled, and how to guarantee the

proper termination of the algorithm? We will address these challenges in this following.

Our algorithm is based on a new definition of the neighborhood graph, where two sensors are neighbors to each other if they monitor the same critical location. As a common assumption adopted in the literature, we assume the communication range is at least twice of the sensing range [2]. As a result, the neighboring sensors could directly communicate with each other in the neighborhood graph.

Before presenting our algorithm, we first give several notations that will be used in our description.

- ΔC_i: the additional effective coverage time that sensor s_i provides when the schedule of s_i is known.
- ΔC_i^{max}: the maximum additional effective coverage time that sensor s_i can provide among all its possible schedules.
- ΔT_j^i: the additional coverage time of the jth elementary region as contributed by the inclusion of s_i's schedule.

The control message is defined as $msg(id, type, sch, \Delta C)$, where the fields id and sch are the id and schedule (i.e., $s.start$ or $s.end$) of the sender, and $type$ is either LAB or UPD, which corresponds to the label notice or the update report.

Algorithm 1 Distributed Algorithm (at each node $s_i, i \in N$)

Input: the neighbor set $N(s_i)$, the neighboring schedules, the critical location set P_i that s_i covers, the importance factor of each location $p_i, i \in P_i$, network lifetime L, battery life B_i, s_i is initially unlabeled
Output: the state (label or unlabel), the optimal schedule if s_i is labled
Procedure:

1: calculate the maximum additional coverage it can provide, i.e., $\Delta C_i^{max} = \max_{s_i.start} \sum_{j \in R(i)} W(j) \times \Delta T_j^i$, and the optimal schedule
2: broadcast $msg(i, UPD, Null, \Delta C_i^{max})$ to its neighbors
3: **while** $\Delta C_i^{max} > 0$ **do**
4: **if** s_i has the largest ΔC_i^{max} among its neighbors **then**
5: s_i labels itself, broadcasts $msg(i, LAB, sch, \Delta C_i^{max})$, and exits.
6: **end if**
7: **if** $msg(k, LAB, sch, \Delta C_k^{max})$ is received **then**
8: update the schedule of its neighbor s_k, recalculate ΔC_i^{max}, and broadcast $msg(i, UPD, Null, \Delta C_i^{max})$. Go to 3
9: **end if**
10: **if** $msg(k, UPD, Null, \Delta C_k^{max})$ is received **then**
11: update the coverage metric of its neighbor s_k and Go to 3.
12: **end if**
13: **end while**

The pseudo code is listed in Algorithm 1. In the beginning, all sensors are unlabeled and have a null schedule. After exchanging their initial schedule, each sensor s_i starts to compute ΔC_i^{max}, the maximum additional effective coverage time it can potentially contribute, and broadcasts this information to its neighbors (line 1-2). On receiving the replies from its neighbors, each sensor will compare its ΔC_i^{max} with that of its neighbors. If it has the largest ΔC_i^{max} among its unlabeled neighbors, it will label itself as the local optimal (line 4-6). Then the label decision (with $type$ field set to be LAB) together with the corresponding optimal schedule will be sent to the local neighborhood, based on which each affected neighbor will recalculate ΔC_i^{max} and send an update report (with $type$ field set to be UPD) to its neighbors (line 7-9). On receiving the update report, each node updates the coverage metric of its neighbors and reevaluates the local optimum condition (line 10-12). This process will repeat rounds after rounds, until no additional coverage can be provided or no more unlabeled sensors are available.

From the algorithmic point of view, the sensors elected per round constitute an Independent Set, which is defined as a subset of nodes among which there is no link between any two nodes. The set is a maximal independent set (MIS) if no more nodes can be added to generate a bigger independent set [13], [14]. Therefore, our algorithm is equivalent to finding a MIS in a distributed manner by labeling the local optimal sensors in each round. At the end of each round, the labeled sensors are removed from consideration in its neighborhood, and then the algorithm is executed among the remaining unlabeled sensors.

The above localized labeling and scheduling process can guarantee that the algorithm terminates. This is because at any moment, there must be at least one local optimal sensor in the network, which is qualified to label itself. As the labeled sensors are removed from consideration, additional sensors can rise up to become the new local optimal for the coming round. Eventually, the algorithm will stop when the coverage can not be further improved.

B. Details of Computing ΔC_i^{max} and ΔC_i

During each iteration, each sensor s_i needs to calculate the maximum additional effective coverage time ΔC_i^{max}. ΔC_i^{max} is the maximum additional coverage that s_i can provide among all its possible schedules, thus $\Delta C_i^{max} = \max_{s_i.start} \Delta C_i$. However, $s_i.start$ falls in the continuous range of $[0, l]$, so there are infinite number of possible values for $s_i.start$. Thus it is infeasible to compute ΔC_i^{max} and the corresponding optimal schedule of s_i through enumeration. We address this challenge by shrinking the search space and focusing on only a finite number of values, while still maintaining the optimality. This simplification is based on the observation that the maximum ΔC_i can be achieved when either $s_i.end$ meets the start point of the neighboring schedule or $s_i.start$ meets the end point of the neighboring schedule.

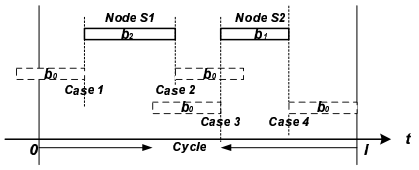

Fig. 2. An example to show how to calculate s_0's optimal schedule based on its neighbors' schedules. With two neighbors s_1, s_2, only four cases need to be considered.

Fig. 2 uses a simple example to illustrate how to calculate the optimal schedule for a node during one iteration. In Fig. 2, s_0 has two neighbors s_1, s_2 which have been labeled and whose schedules are already determined. Then to calculate the optimal schedule for s_0, we only need to consider the following four cases: $s_0.end = s_1.start$, $s_0.start = s_1.end$, $s_0.end = s_2.start$, $s_0.start = s_2.end$. This is because ΔC_0 shall not increase when s_0's on-period moves a tiny step either to the right or to the left at these points. Therefore, the

collection of these cases represent all local maximum points, from which the global optimal point can be chosen.

In general, to calculate ΔC_i^{max}, we need to first identify all start and end points for each neighbor of s_i, then produce the set $E_i = \bigcup_{j \in N(i)} \{s_j.end \mod l, (s_j.start - b_i) + l \mod l\}$, and select $s_i.start$ from E_i to optimize ΔC_i. E_i is derived based on the modular operation since $s_i.start$ is in the range $[0, l]$. As can be seen, the total computational complexity is $O(|N(i)|)$, where $|N(i)|$ is the number of neighbors of s_i. To further reduce the computational overhead, we can also combine the schedules of the neighbors that cover the same critical location, which will reduce the number of local optimal points under consideration.

Finally, we explain how to calculate ΔC_i given the schedule of s_i. Define $\Delta C_i = \sum_{j \in R(i)} W(j) \times \Delta T_j^i$, where $W(j)$ is the accumulated importance factor and ΔT_j^i is the additional coverage time of the jth elementary region as contributed by the inclusion of s_i's schedule. Note that different regions may have different combined monitoring schedule. Then, given a specific region j, the additional coverage time as contributed by s_i can be calculated by $\Delta T_j^i = b_i - \overline{R_j S_i}$, where $\overline{R_j S_i}$ is the on-period overlap between the combined schedule of region j and the schedule of s_i. Given the monitoring schedule of region j and s_i, it is easy to compute $\overline{R_j S_i}$, where the on-period of the combined schedule of region j could be either continuous or discontinuous. Due to space limitation, we omit the detail here.

C. A Numerical Example

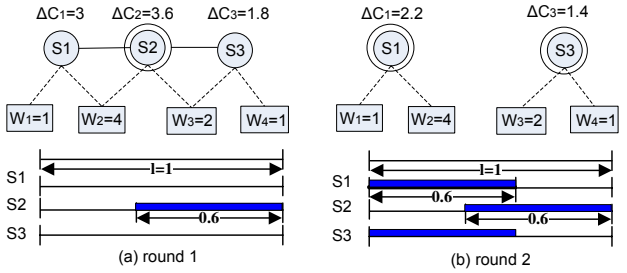

Fig. 3. An example to illustrate the distributed algorithm based on the simple network in Fig. 1. The neighborhood graph is shown in the upper part, with circle denoting the sensor and rectangle denoting the critical location, and the determined schedule is shown in the lower part. (a) round 1, where s_2 is labeled (b) round 2, where s_1, s_3 are labeled.

Fig. 3 uses an example to illustrate how the distributed algorithm works. Based on the network shown in Fig. 1, the neighborhood graph in Fig. 3 consists of three sensors (circle nodes), covering four critical locations (rectangle nodes). Each location j is assigned an importance factor w_j. Fig. 3 shows that the algorithm runs in two rounds, with the remaining neighborhood graph shown in the upper part and the determined schedule shown in the lower part. In the beginning, each node calculates its $\Delta C_i^{max} = \max_{s_i.start} \sum_{j \in R(i)} W(j) \times \Delta T_j^i$ and exchanges this information with its neighbors. After calculation, it is known that $\Delta C_1^{max} = 3, \Delta C_2^{max} = 3.6, \Delta C_3^{max} = 1.8$. Since ΔC_2^{max} is the maximum, s_2 is labeled and broadcasts the label notice together with the optimal schedule to s_1 and s_3. In the second round, after s_1 and s_3 update their coverage, they find themselves to be the local optimal and thus label themselves. Note that Fig. 3(b) shows just one of the optimal schedules, and there may exist other optimal schedules.

IV. Distributed Algorithm Analysis

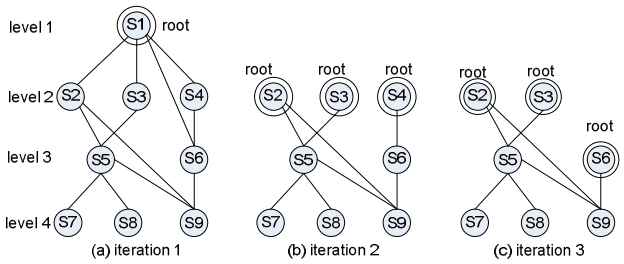

Fig. 4. An example of the dependency graphs corresponding to different iterations of the reordering algorithm. The initial dependency graph in the first iteration is built upon the whole sensor set selected in the distributed algorithm.

Our distributed algorithm has low overhead. The computation complexity for each node is $O(d \cdot R)$, where d is the maximum number of neighbors for a node and R is the number of rounds which is usually small as shown in the simulation result. The message complexity per node is $O(R)$. To compare the performance of the algorithm with the optimal, we derive the approximation ratio of the distributed algorithm in the remainder of the section. To do this, we need to first preprocess the outcome of the distributed algorithm by rearranging the labeling order of the sensors. This is equivalent to first run a reordering algorithm upon the sensors labeled in the distributed algorithm.

A. Reordering the Labeled Sensors

The reordering process is based on a newly defined *dependency graph*, which is built upon the sensors selected in the distributed algorithm. In the dependency graph, the labeled sensors are classified into different levels, corresponding to different rounds of the distributed algorithm. For example, the dependency graph in Fig. 4(a) has four levels, where level 1 consists of the root node s_1 which is labeled in the first round, level 2 has nodes s_2, s_3, and s_4 which are labeled in the second round, and so on. Two nodes have a link between each other if they are neighbors in the neighborhood graph, i.e, they share a common critical location to cover. A node is called a root node if it has no high-level neighbor.

The reordering algorithm is straightforward, which consists of multiple iterations. During each iteration, it picks the root with the largest value of C_i^{max}. If there is a tie, it can be broken based on the id. The picking order will be the new index of that sensor. The same operation is repeated until all nodes on the dependency graph are reordered. Taking Fig. 4 as an example, in the first iteration, the root node s_1 is picked and assigned index 1. In the second iteration, the root node s_4 is picked, which has the largest C_i^{max} among the root nodes, and assigned index 2, and so on.

Next, we give several results regarding the dependency graph and the reordering algorithm.

Lemma 1: In the dependence graph, the nodes belonging to the same level cannot be neighbors.

Proof: If the nodes belong to the same level, they are the local optimal during the same round of the algorithm execution, which means that they cannot be neighbors to each other. ■

Lemma 2: the maximum additional effective coverage time for each sensor, i.e., $\Delta C_i^{max} = \max_{s_i.start} \sum_{j \in R(i)} W(j) \times \Delta T_j^i$, is a non-increasing function of time as more sensors are labeled.

Proof: In the expression of ΔC_i^{max}, the accumulated importance factor $W(j)$ is a constant value but the additional coverage time ΔT_j^i is a variable depending on the combined monitoring schedule of region j and the value of $s_i.start$. $s_i.start$ is always in the range $[0, l]$, but the combined schedule could be enlarged as more sensors are labeled. Therefore, given a fixed value of $s_i.start$, ΔT_j^i is a non-increasing function, which means that ΔC_i is non-increasing as well. ■

Theorem 2: For each iteration of the reordering algorithm, the local optimal nodes must be among the root nodes of the dependency graph.

Proof: It can be proved by contradiction. Suppose for some iteration of the reordering algorithm, there exists a non-root local optimal sensor s_o, then we will prove that this leads to a contradiction. Note that s_o can either belong to the dependency graph or not.

If the non-root s_o belongs to the dependency graph, according to the definition of the root node, s_o must have neighbors at higher level. Among those neighbors, suppose the level of s_k (at level k) is the highest. Since s_o is the local optimal, $\Delta C_o^{max} > \Delta C_k^{max}$ for the current iteration. However, this contradicts with the fact that in the kth round of the distributed algorithm, s_k is the local optimal and labeled earlier than s_o, which means that $\Delta C_o^{max} < \Delta C_k^{max}$[3]. This shows that a non-root node in the dependency graph cannot be the local optimal.

If s_o does not belong to the dependency graph, again, this can lead to two possibilities. First, if s_o has neighbors in the dependency graph, the proof is similar to the above case. Second, if s_o has no neighbor in the dependency graph, it should be labeled together with the root nodes at certain level, which means that it should belong to the dependency graph. Therefore, any node not belonging to the dependency graph cannot be the local optimal either. ■

B. Approximation Ratio

The analysis of the approximation ratio is based on the new ordering of the sensors labeled in the distributed algorithm. Due to the unique challenges of our problem, we cannot simply borrow the traditional techniques such as those used in the set cover model [15]. This is because our algorithm not only needs to select sensors but also needs to determine their corresponding schedules. In addition, the scheduling decision needs to be made in a continuous space while there are infinite possibilities to place the on-period in a cycle. Therefore, new analytical techniques are needed.

Theorem 3: The distributed algorithm achieves an approximation factor of 0.5 for the $MaxEct$ problem.

Proof: Suppose there are total n sensors labeled in the distributed algorithm. We first rearrange the index of the n sensors by the aforementioned reordering algorithm. By simply changing the labeling order, with the whole set of sensors and their schedules intact, the total effective coverage time will not be affected. Let s_i^d denote the sensor picked in the ith iteration in the distributed algorithm after reordering, then i is the new index of the sensor. We further use ΔC_i^d to denote the maximum additional effective coverage time that s_i^d can provide, given the previous $i-1$ sensors and their schedules.

The vector $V_i^d = (v_{i1}^d, v_{i2}^d ..., v_{i|R|}^d)$ is used to denote the coverage vector of s_i^d, with each component corresponding to one of the $|R|$ elementary regions. Each component v_{ij}^d denotes the time periods during which the region j is covered by s_i^d. For example, $v_{ij}^d = \{[1,3],[5,6]\}$ means region j is covered by sensor s_i^d during the periods [1,3] and [5,6]. In general, each period in v_{ij}^d is determined by a start time and an end time, with no overlap between the different periods. If region j is not covered by sensor s_i^d, set $v_{ij}^d = \{[0,0]\}$. $|v_{ij}^d|$ is used to denote the total length of time during which region j is covered by s_i^d. For example, if $v_{ij}^d = \{[1,3],[5,6]\}$, then $|v_{ij}^d| = 2+1 = 3$.

Define the norm of the vector V_i^d in the form of a weighted sum, i.e., $|V_i^d| = \sum_{j=1}^{|R|} W(j) \times |v_{ij}^d|$, where the weight is the accumulated importance factor. For example, if $V_i^d = (\{[1,3],[5,6]\},[0,0],\{[2,4]\})$, then $|V_i^d| = 3W(1) + 2W(3)$. It can be seen that $|V_i^d|$ is the total effective coverage time provided by s_i^d.

Further suppose $V_i = (v_{i1}, v_{i2} ..., v_{i|R|})$ and $V_j = (v_{j1}, v_{j2} ..., v_{j|R|})$ are two different vectors, then we define the vector addition/substraction operation as follows:

$$V_i + V_j = (v_{i1} \cup v_{j1}, v_{i2} \cup v_{j2}, ..., v_{i|R|} \cup v_{j|R|}) \quad (5)$$

$$V_i - V_j = (v_{i1} - v_{i1} \cap v_{j1}, ..., v_{i|R|} - v_{i|R|} \cap v_{j|R|} \quad (6)$$

Thus, the total coverage vector generated by the distributed algorithm is $V^d = \sum_{i=1}^{n} V_i^d$, and the total effective coverage time is

$$C_{dis} = \sum_{i=1}^{n} \Delta C_i^d = |V^d| = |\sum_{i=1}^{n} V_i^d| \quad (7)$$

Now, let's turn to the optimal algorithm. Suppose there are m iterations in the optimal algorithm, and one sensor is picked in each iteration. Then we rearrange the order of sensors selected in the optimal algorithm according to the new index of the sensors labeled in the distributed algorithm, such that if a sensor is picked by both algorithms, it is picked in the same iteration. With the schedule of each sensor intact, simply changing the order of sensor selection will not affect the outcome of

[3]For a more detailed analysis, we need to identify the fact that the value of ΔC_k^{max} in the current iteration is no smaller than its value in the kth round, and the value of ΔC_o^{max} is no larger as compared with the kth round. The deduction of these conclusions will require the result of Lemma 1, 2.

the optimal algorithm. Similar to the distributed algorithm, we also use the notations $s_i^o, V_i^o, |V_i^o|, v_{ij}^o, |v_{ij}^o|, \Delta C_i^o, V^o, C_{opt}$ for the optimal algorithm.

According to the new ordering, in the ith iteration, the distributed algorithm picks sensor s_i^d and enhances the total coverage by ΔC_i^d. Thus we have $\Delta C_i^d = |V_i^d - \sum_{j=1}^{i-1} V_j^d|$. According to Theorem 2, during each iteration the reordering algorithm picks the sensor that can provide the global maximum additional coverage, thus $|V_i^d - \sum_{j=1}^{i-1} V_j^d| \geq |V_i^o - \sum_{j=1}^{i-1} V_j^d|$, where V_i^o is the coverage vector of the sensor s_i^o picked in the same iteration by the optimal algorithm. In other words, the distributed algorithm picks s_i^d instead of s_i^o because s_i^d can provide more additional coverage than s_i^o. On the other hand, $V^d \supseteq \sum_{j=1}^{i} V_j^d$ for $\forall i = 1...n$. Therefore we have

$$\Delta C_i^d \geq |V_i^o - \sum_{j=1}^{i-1} V_j^d| \geq |V_i^o - V^d|, \quad \forall i = 1...n \tag{8}$$

Since n sensors are picked in the distributed algorithm and m sensors are picked in the optimal algorithm, to make the above relationship also applicable to the case when $m < n$, let $V_i^o = ([0,0], ..., [0,0])$ for $\forall i \in (m, n]$. Then adding all the inequalities in Eqn. 8 gives

$$\sum_{i=1}^{n} \Delta C_i^d = C_{dis} \geq \sum_{i=1}^{n} |V_i^o - V^d| \tag{9}$$

Note that the effective coverage time of each sensor is denoted by a vector and each component of a vector is a collection of time periods. Then based on the vector analysis and the addition/substraction defined in Eqns. 5 & 6, it is not hard to see

$$\sum_{i=1}^{n} |V_i^o - V^d| \geq |\sum_{i=1}^{n} (V_i^o - V^d)| \geq |\sum_{i=1}^{n} V_i^o| - |V^d| \tag{10}$$

As $C_{opt} = |\sum_{i=1}^{n} V_i^o|$ and $C_{dis} = |V^d|$, combining Eqn. 9 and Eqn. 10 gives

$$C_{dis} \geq C_{opt} - C_{dis} \tag{11}$$

The above relationship shows that the approximation ratio of Algorithm 1 is 0.5. ■

V. Performance Evaluation

A. Schemes for Comparison

In this section, we use simulations to compare the proposed distributed algorithm (termed as distributed *ECT*) with three other schemes. Since most existing works focus on the area coverage without considering the lifetime constraint, we have extended these schemes to consider location coverage and lifetime requirement, such that they can be applied in our context.

- Randomized Scheme: each node generates a random schedule, i.e., placing its on-period randomly in the cycle. Then the redundant nodes are turned off, and only the nodes which can contribute to the coverage will remain power on.
- Coverage Configuration Protocol (CCP) [2]: the original CCP was proposed for area coverage, which was shown to outperform other schemes in most scenarios. Its objective is to select the minimum number of sensors to provide full coverage. We extend CCP to location coverage, and enable it to operate continuously even when some sensors die due to limited battery. Specifically, when a sensor dies, some other sleeping sensors may be activated if they can contribute to the total coverage time of the critical locations.
- Minimum Redundancy Protocol (MRP) [9]: MRP is a distributed heuristic that aims to minimize the overlapping time of the neighboring schedules at each node. The redundant node will be turned off if its coverage has been provided by the other working nodes.

B. Simulation Setup

In the simulation, n sensors and p locations are randomly distributed in a 10×10 square area, with n varying from 100 to 500 and p from 10 to 50. As in most sensor platforms, the communication range is at least twice of the sensing range; we set the sensing range to be 1 and the communication range to be 2. Each location is associated with a weight w, the importance factor of the location. $w = 1$ unless otherwise specified. The battery/network lifetime ratio is denoted by ν, if the initial battery capacity is homogeneous. However, in the heterogeneous case, the initial battery is represented by a random variable ν_i, which is uniformly distributed in $[\frac{\nu}{2}, \frac{3\nu}{2}]$ where ν is the average ratio. We assume the required network lifetime is 10 units, and the length of the cycle is normalized to be 1 unit.

Different schemes are evaluated in terms of the total effective coverage time, the average weighted event detection probability, the number of working nodes, and the network lifetime. Randomized event is considered whose occurrence is uniformly distributed in the cycle among the critical locations. The length of the event is assumed to be half of the on-period per cycle in our experiment. The event detection probability is measured by the fraction of time during which the event can be monitored by any sensor node. 1000 randomized events are simulated, and the simulations are done with a customized C++ simulator.

C. Simulation Results

Figs. 5-9 study the effect of varying the number of sensor nodes on the performance, when $p = 20, \nu = \frac{1}{5}$. Fig. 5 shows that the total effective coverage time increases as the number of nodes increases in all algorithms. Among different schemes, the distributed ECT algorithm has the best performance and the randomized algorithm has the worst performance. CCP and MRP stay close to each other, with MRP starting off better but lagging behind CCP when n increases beyond 200. This is because unlike the other four schemes, CCP does not divide the node activity into cycles. Once a node in CCP is turned on, it will not be off until its battery is run out. Therefore, the

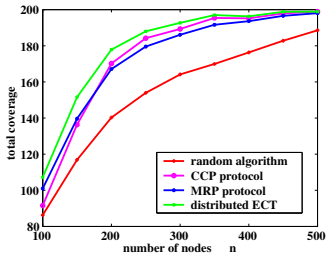

Fig. 5. Total coverage Vs. number of nodes ($p = 20, \nu = \frac{1}{5}$)

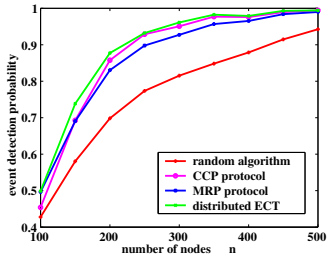

Fig. 6. Event detection probability Vs. number of nodes ($p = 20, \nu = \frac{1}{5}$)

number of nodes affects the performance of CCP more than the other schemes.

Fig. 6 shows the influence of the number of sensors on the event detection probability, which has the same trend as Fig. 5. This is expected since larger coverage normally implies higher event detection probability. In fact, when a large number of events are uniformly distributed in time and space, the total weighted event detection probability converges to the total effective coverage time, differing by a factor l, the length of the cycle. Therefore, we show the result of either coverage or probability in the following experiments, but not both, unless necessary.

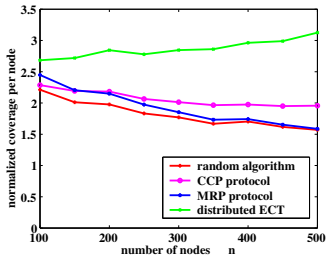

Fig. 7. Normalized coverage Vs. number of nodes ($p = 20, \nu = \frac{1}{5}$)

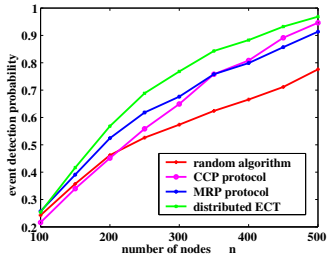

Fig. 8. Event detection probability Vs. number of nodes ($p = 20, \nu = \frac{1}{10}$)

Fig. 7 compares the different schemes in terms of the normalized coverage per node, calculated by dividing the total coverage by the number of working nodes. It can be seen that the distributed ECT algorithm outperforms the other schemes by 20% to 50% when n exceeds 100, and the performance gap is enlarged as n increases. When referring back to Figs. 5, 6, we can see that the normalized performance improvement in distributed ECT is much more noticeable than its performance without being normalized. This is because when n is large, distributed ECT selects much less working nodes than the other schemes, and thus has much better normalized performance.

Fig. 8 examines the average event detection probability again, but reduces the battery/network lifetime ratio ν from $\frac{1}{5}$ to $\frac{1}{10}$. Compared with Fig. 6, the distributed ECT algorithm outperforms CCP by a greater margin, e.g., the improvement is up to 20% when $n = 200$. This shows that the performance of CCP not only relies on the number of nodes but also on the battery/network lifetime ratio.

Fig. 9 shows that the distributed ECT algorithm consistently satisfies the lifetime constraint, irrespective of the number of nodes, but CCP fails to meet the network life time requirement

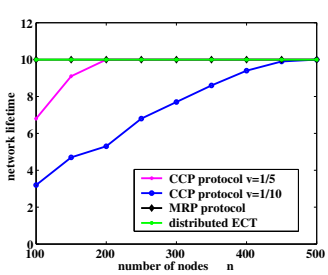

Fig. 9. Network life Vs. number of nodes ($p = 20$)

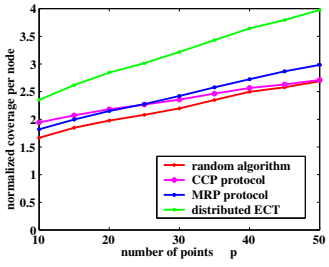

Fig. 10. Normalized coverage Vs. number of locations ($n = 200, \nu = \frac{1}{5}$)

when the number of nodes falls below a threshold. The threshold is 200 when $\nu = \frac{1}{5}$ and 450 when $\nu = \frac{1}{10}$. The performance distinction is due to the different design philosophies. CCP aggressively uses each node until its battery is consumed, but in ECT (also in MRP and randomized schemes), node activity is divided into cycles and the battery consumption is fairly distributed among different cycles according to the lifetime requirement. The lifetime of CCP is measured as the time when 80% of the nodes fail.

Figs. 10-11 study the effect of different number of locations on the performance of the normalized coverage, and the number of working nodes, when $n = 200, \nu = \frac{1}{5}$. As the number of locations in the field increases, the normalized coverage also increases. Similar to Fig. 7, in Fig. 10, the distributed ECT algorithm features 20%-50% improvement over the other schemes. This is because, as evidenced by Fig. 11, ECT requires much less nodes to satisfy the lifetime constraint, thus improves the normalized performance substantially.

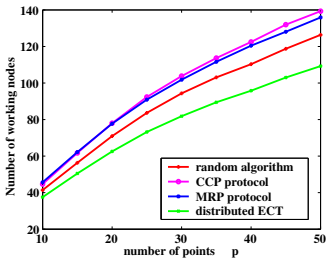

Fig. 11. Number of working nodes Vs. number of locations ($n = 200, \nu = \frac{1}{5}$)

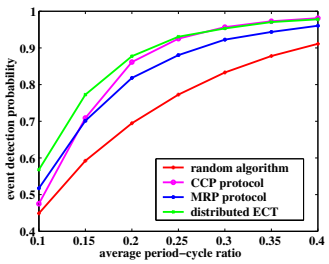

Fig. 12. Event detection probability Vs. battery/lifetime ratio ($n = 200, p = 20$, heterogeneous battery)

Fig. 12 shows the relationship between the average event detection probability and the average battery/network lifetime ratio ν, with the heterogeneous batteries. For example, we can read from the figure that given $n = 200, p = 20$, in order to reach 90% probability, ν needs to be greater than 0.2. Again, CCP starts as low as the randomized algorithm, but when ν exceeds 0.2, its performance curve approaches ECT closely. This confirms our former observation.

VI. Related Work

Sensor coverage has been extensively studied in the literature. Depending on the subject covered, existing works can be classified into area coverage, point coverage, and barrier

coverage [16]. In terms of area coverage, many works focus on how to select the minimum number of sensors to preserve the coverage degree (e.g., 1-degree or *k*-degree) [1], [2], [3], [4], [5], [6], [17], but provide no network lifetime guarantee. In [18], a centralized scheduling algorithm is proposed to sequentially activate the sensors and guarantees a $O(logn)$ factor of the maximum network lifetime, where n is the total number of nodes. In [8], a distributed scheduling algorithm is proposed which achieves a $O(logn * lognB)$ performance factor, where B is the upper bound of the initial battery. In [19], [20], full-view coverage which considers the sensor's viewing direction is introduced for camera sensor networks, and techniques have been proposed to achieve full-view coverage.

The point coverage has been studied in [10], [11], with the objective to maximize the network lifetime given a set of targets to be covered. With the assumption that one sensor covers only one target a time, the optimal schedule can be derived based on the technique of matrix decomposition [11]. However, in real life most sensors monitor multiple targets at the same time. Our work falls into the category of point coverage. However, considering the network lifetime as a constraint, the problem becomes NP-hard, thus the approximation scheme is the best we can achieve.

There are also some other works focusing on the point coverage [21], [22], whose purpose is to minimize the probability of undetected penetration through the barrier.

While all of the above works treat lifetime as an objective, we consider the network lifetime as a constraint, with the aim to schedule the sensor activity to maximize the total effective coverage time of the critical locations. The work in [9] also considers network lifetime as a constraint, but it focuses on the area coverage and requires the knowledge of the area size of each elementary region, which is infeasible to compute in reality. To solve this, a distributed heuristics is proposed in [9] with the aim of minimizing the overlap of neighboring schedules, but no approximation ratio is given as compared with the global optimal performance.

Theoretically, we have designed approximation schemes based on a new metric of effective coverage time. Our model is different from the traditional maximum coverage problem, which is known to have the $\frac{e-1}{e}$-approximation bound [15]. This is because in our problem, we not only need to select sensors but also need to determine their corresponding schedules in a continuous cycle. Therefore, the $\frac{e-1}{e}$ ratio cannot be applied here. By contrast, we model the effective coverage time as a vector and propose a 0.5-approximation algorithm.

VII. CONCLUSIONS

In this paper, we have studied the critical location coverage problem in wireless sensor networks with lifetime constraints. By dividing the sensor activity into cycles, we formulate it as a scheduling problem which devises the sensor active schedule such that the total effective coverage time is maximized. Through analysis, we show that the problem is NP-hard and then propose a distributed algorithm. Based on dependency graph and vector analysis, the distributed algorithm is proved to have a constant approximation ratio of 0.5.

Through extensive simulations, we compare the proposed algorithm with the extensions of several state-of-the-art schemes, and observe significant performance enhancements. Specifically, in sparse network (i.e., sensor density is low or the battery/lifetime ratio is small), the proposed algorithm renders around 20% more coverage and event detection probability than existing schemes, and in dense network (i.e., sensor density is high or battery/lifetime ratio is large), although the proposed algorithm shows similar performance of coverage or event detection probability to other schemes, it requires much less working nodes to meet the lifetime requirement. In either case, our algorithm demonstrates 20%-50% improvement in terms of normalized coverage and event detection probability.

REFERENCES

[1] D. Tian and N. Georganas, "A coverage-preserving node scheduling scheme for large wireless sensor networks," in *ACM international workshop on Wireless sensor networks and applications*, 2002.

[2] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill, "Integrated coverage and connectivity configuration in wireless sensor networks," in *ACM SenSys*, 2003.

[3] T. Yan, T. He, and J. A. Stankovic, "Differentiated surveillance for sensor networks," in *ACM SenSys*, 2003.

[4] H. Zhang and J. C. Hou, "Maintaining sensing coverage and connectivity in large sensor networks," *Wireless Ad Hoc and Sensor Networks*, 2005.

[5] Y. Zou and K. Chakrabarty, "A distributed coverage and connectivity centric technique for selecting active nodes in wireless sensor networks," *IEEE Tran. Computer*, 2005.

[6] S. Kumar, T. H. Lai, and J. Balogh, "On k-coverage in a mostly sleeping sensor network," in *ACM MOBICOM*, 2004.

[7] S. Funkey, A. Kesselman, F. Kuhn, and Z. Lotker, "Improved approximation algorithms for connected sensor cover," *Wireless Networks*, 2007.

[8] G. S. Kasbekar, Y. Bejerano, and S. Sarkar, "Lifetime and coverage guarantees through distributed coordinate-free sensor activation," in *ACM Mobicom*, 2009.

[9] C. Liu and G. Cao, "Spatial-temporal coverage optimization in wireless sensor networks," *IEEE Trans. On Mobile Computing*, 2011.

[10] M. Cardei, M. Thai, Y. Li, and J. Wu, "Energy-efficient target coverage in wireless sensor networks," in *IEEE INFOCOM*, 2005.

[11] H. Liu, P. Wan, C.-W. Yi, X. Jia, S. Makki, and P. Niki, "Maximal lifetime scheduling in sensor surveillance networks," in *IEEE INFOCOM*, March 2005.

[12] M. H. Alsuwaiyel, *Algorithms Design Techniques and Analysis*. World Scientific Publishing Company, 1999.

[13] S. Basagni, "A distributed algorithm for finding a maximal weighted independent set in wireless networks," in 11^{th} *International Conference on Parallel and Distributed Computing and Systems (PDCS)*, 1999.

[14] C. Liu and G. Cao, "Distributed monitoring and aggregation in wireless sensor networks," in *IEEE INFOCOM*, 2010.

[15] S. Khuller, A. Moss, and J. Naor, "The budgeted maximum coverage problem," *Information Processing Letter*, vol. 70, no. 1, pp. 39–45, 1999.

[16] M. Cardei and J. Wu, "Energy-efficient coverage problems in wireless ad hoc sensor networks," *Journal of Computer Communications on Sensor Networks*, 2005.

[17] M. Hefeeda and H. Ahmadi, "An integrated protocol for maintaining connectivity and coverage under probabilistic models for wireless sensor networks," *Ad Hoc & Sensor Wireless Networks*, 2009.

[18] P. Berman, G. Calinescu, C. Shah, and A. Zelikovsly, "Efficient energy management in sensor networks," *Ad hoc and sensor networks*, 2005.

[19] Y. Wang and G. Cao, "On Full-View Coverage in Camera Sensor Networks," *IEEE Infocom*, 2011.

[20] ——, "Barrier coverage in camera sensor networks," in *ACM Mobihoc*, 2011.

[21] B. Liu, O. Dousse, J. Wang, and A. Saipulla, "Strong barrier coverage of wireless sensor networks," in *ACM Mobihoc*, 2008.

[22] A. Saipulla, C. Westphal, B. Liu, and J. Wang, "Barrier coverage of line-based deployed wireless sensor networks," in *IEEE INFOCOM*, 2009.