Taming the Exponential State Space of the Maximum Lifetime Sensor Cover Problem

Akshaye Dhawan
Department of Mathematics and Computer Science
Ursinus College
Collegeville, PA 19426
Email: adhawan@ursinus.edu

Sushil K. Prasad
Department of Computer Science
Georgia State University
Atlanta, GA 30030
Email: sprasad@gsu.edu

***Abstract*—A key problem in Wireless Sensor Networks is that of scheduling sensors into sleep-sense cycles that maximize the lifetime of the network while ensuring coverage of a set of targets. This is a known NP-complete problem, due to the exponential number of possible ways to select a subset of sensors to turn on. Our earlier work had presented a unique model for this problem by introducing a lifetime dependency (LD) graph in which these possible cover sets are nodes and edges represent shared sensors between them. Using the graph properties, we presented a range of effective distributed heuristics. Even though the local subgraphs are practically tractable, their theoretically exponential growth has remained a persistent issue. In this paper, we present a theoretical model for representing this exponential space of possible cover sets that groups related covers together, thereby reducing this exponential space virtually to a linear one. We then present an equivalence class (EC) graph as a model for this reduction. We use these underlying theoretical properties to develop a smart linear time sampling algorithm of this exponential space. To demonstrate the effectiveness of our sampling technique, we employ the sampled localized LD subgraph as input to our previous heuristics. Simulation studies show about two-fold speedup while reducing solution quality (network lifetime) only by under 10% when compared to our previous heuristics that operate on the exponential space. We also compare our algorithms to two other state-of-art greedy algorithms and show that ours still manage to achieve improvements of around 8-10% over them.**

I. INTRODUCTION

Wireless Sensor Networks (WSNs) consist of a number of low cost sensors that are equipped with a radio interface. These devices are deployed in large numbers over an area of interest and they monitor the targets in this region and send information to a base station or a gateway node [1]. Due to their broad applications in disaster relief, homeland security, military and other important applications, these networks have attracted intense research interest.

A key constraint of sensor nodes is a limited battery that cannot be replenished. Once the battery has been exhausted, the sensor is useless. Hence, efficiently managing energy at every layer of the network stack is critical. In this paper we study the problem of maximizing the lifetime of a dense network of static sensors while ensuring that they cover certain predefined targets. The lifetime of the network is defined as the amount of time that the network can cover its *area* or *targets* of interest. Having all the sensors remain “on” trivially ensures coverage but also significantly reduces the lifetime of the network as the nodes would discharge quickly.

A standard approach taken to maximizing the lifetime is to make use of the overlap in the sensing regions of individual sensors caused by the high density of deployment. Hence, only a subset of all sensors need to be in the “on” or “sense” state, while the other sensors can enter a low power “sleep” or “off” state. The members of this active set, also known as a *sensor cover* or the *cover set*, are then periodically changed so as to keep the network alive for longer duration. In using such a scheduling scheme, there are two problems that need to be addressed. First, we need to determine how long to use a given cover set and then we need to decide which set to use next. This problem has been shown to be NP-complete [2], [3].

Existing work on this problem has looked at both centralized and distributed algorithms to come up with such a schedule (See Section VI for a more detailed description of related work). The distributed algorithms typically operate in rounds. At the beginning of each round, a sensor exchanges information with its neighbors, and makes a decision to either switch on or go to sleep. In most algorithms, the sensor with some simple greedy criteria like the largest uncovered area [4], maximum uncovered targets [5], etc. is selected to be on.

More recently, we have proposed several distributed algorithms in [6]–[9] based on the concept of a *Lifetime Dependency* (LD) Graph. Our motivation was to study underlying properties of the problem and develop distributed heuristics that make use of these properties. The definition of the LD graph comes from understanding that the problem with scheduling is that a sensor can be a part of multiple covers which have an impact on each other, as using one cover set reduces the lifetime of another set that has sensors common with it. By making greedy choices, the impact of this dependency is not being considered. Briefly, the lifetime dependency graph consists of the cover sets as nodes with any two nodes connected if the corresponding covers intersect (See Section II for a formal definition).

If we consider the LD graph, it is quickly obvious that even creating this graph will take exponential time since there are 2^n cover sets to consider where, n is the number of sensors. However, the target coverage problem has a useful property - if the local targets for every sensor are covered, then globally, all targets are also covered. In [6], we make use of this property to look at the LD graph locally (fixed 1-2 hop neighbors), and are able to construct all the local covers for the local targets and then model their dependencies. Based on these dependencies, a sensor can then prioritize its covers and negotiate these with its neighbors. Simple heuristics based on properties of this graph were presented in [6] and showed a 10-15% improvement over comparable algorithms in the literature. [7] built on this work by examining how an optimal sequence would pick covers in the LD graph and designing heuristics that behave in a similar fashion. Though the proposed heuristics are efficient in practice, the running time is a function of the number of neighbors and the number of local targets. Both of these are relatively small for most graphs but theoretically are exponential in the number of targets and sensors.

Contributions: A key issue that remains unresolved is the question of how to deal with this exponential space of cover sets. In this paper we present a reduction of this exponential space to a linear one based on grouping cover sets into *equivalence classes*. We use $[C_i]$ to denote the equivalence class of a cover C_i. The partition defined by the equivalence relation on the set of all sensor covers stems from the understanding that from the possible exponential number of sensor covers, several covers are very similar, being only minor variations of each other. In Section III, we present the definition of the relation $\Re$, based on a grouping that considers cover sets equivalent if their lifetime is bounded by the same sensor. We then show the use of this relation to collapse the exponential LD Graph into an *Equivalence Class* (EC) Graph with linear number of nodes. This theoretical insight allows us to design a sampling scheme that selects a subset of all local covers based on their equivalence class properties and presents this as an input to our simple LD graph degree-based heuristic. Simulation results show that class based sampling cuts the running time of these heuristics by nearly half, while only resulting in a less than 10% loss in quality.

The remainder of this paper is organized as follows. In Section II, we introduce the notation that we use in this paper and also provide a definition for the LD Graph. Section III presents our theory for grouping cover sets into related partitions and presents the Equivalence Class (EC) Graph as a model for these partitions. We also show some key properties of the EC graph. Section IV uses these ideas to select a sample of cover sets based on what partitions they are located in. We then present a modification of our LD Graph based heuristics that operate on this sample as opposed to the set of local covers. In Section V we simulate the performance of these sampling based heuristics and compare them to other approaches. We briefly examine related work in the literature in Section VI. Finally, we conclude in Section VII.

II. Preliminaries

a) Definitions: Let us start with a few definitions and notations to be employed throughout. Let the sensor network be represented using graph SN where, $S = \{s_1, s_2, \ldots, s_n\}$ is the set of sensors, and an edge between sensor s_i and s_j exists if the two sensors are in communication range of each other. Note that in a network with fixed communication ranges for each node, this model becomes a unit disk graph.

Let the set of targets be $T = \{t_1, t_2, \ldots, t_m\}$. In this paper we consider the problem of covering a stationary set of targets. This can easily be translated into the area coverage problem by mapping the area to a set of points which need to be covered [10], [11].

In addition to this, we define the following notation:

- $b(s)$: The battery available at a sensor s.
- $T(s)$: The set of targets in the sensing range of sensor s.
- $N(s, k)$: The neighbors of sensor s at k or fewer communication hops (including s).
- Cover C: Cover $C \subseteq S$ to monitor targets in T is a minimal set of sensors such that each target $t \in T$ has a nearby sensor $s \in C$ which can sense t, i.e., $t \in T(s)$.
- $lt(C)$: Maximum lifetime of a cover C is $lt(C) = min_{s \in C} b(s)$.
- $lt(t_i)$: The lifetime of a target $t_i \in T$ is given by $lt(t_i) = \sum_{\{s|t_i \in T(s)\}} b(s)$.
- Bottleneck Sensor: Bottleneck sensor s of cover C is the sensor $s \in C$ with minimum battery, i.e., it is the sensor s that upper bounds $lt(C)$.
- Bottleneck Target (t_{bot}): The target with the smallest lifetime $lt(t_{bot})$.

b) The Lifetime Dependency (LD) Graph [6]:: The Lifetime dependency graph $LD = (V, E)$ where V is the set of all possible covers to monitor targets in T and and two covers C and C' are joined by an edge in E if and only if $C \cap C' \neq \emptyset$.

The LD graph effectively captures the dependency between two cover sets by representing their intersection by the edge between them. Further, we define,

- $w(e)$: Weight of an edge e between covers C and C' is $w(e) = min_{s \in C \cap C'} b(s)$.
- $d(C)$: Degree of a node or cover C is $d(C) = \Sigma_{e\ incident\ to\ C}\ w(e)$.

The reasoning behind this definition of the edge weight comes from considering a simple two-node LD graph (see Fig. 1) with two covers C_1 and C_2 sharing an edge e. The lifetime of the graph is upper bounded by $min(lt(C_1)+lt(C_2), w(e))$.

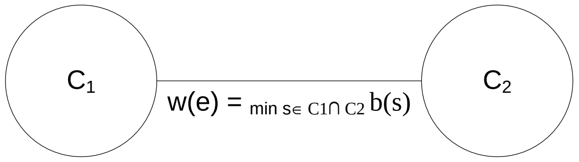

Fig. 1. A simple 2-node LD graph

Similarly, the reasoning behind the definition of the degree of a cover C is that by summing the weights of all the edges incident on the cover C, we are getting a measure of the impact it would have on all other covers with which it shares an edge.

III. Dealing with the Exponential Space

In this section, we present our approach of dealing with the exponential solution space of possible cover sets. The next section utilizes these ideas to develop heuristics for maximizing the lifetime of the network. As explained in Section I, even though the total number of cover sets for the network may be exponential in the number of sensors, for any given cover set, there are several other sets that are very *similar* to this set. We begin by attempting to define this notion of similarity by expressing it as an equivalence relation.

Definition 1: Let $\Re$ be an equivalence relation defined on the set of all sensor covers such that $C_i \Re C_j$ if and only if C_i and C_j share the same bottleneck sensor s_{bot}.

Clearly, $\Re$ is an Equivalence Relation. Every equivalence relation defined on a set, specifies how to partition the set into subsets such that every element of the larger set is in exactly one of the subsets. Elements that are related to each other are by definition in the same partition. Each such partition is called an *equivalence class*. Hence, the relation $\Re$ partitions the set of all possible sensor covers into a number of *disjoint* equivalence classes.

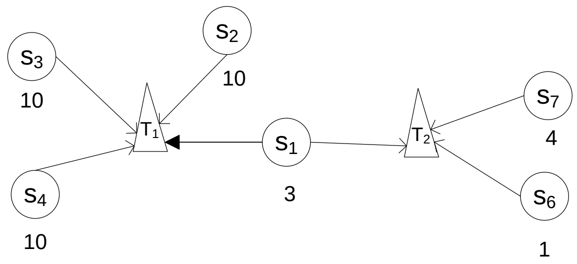

Fig. 2. Example Sensor Network

Notation: Henceforth, we represent the equivalence class of covers sharing a bottleneck sensor s_i by $[s_i]$. Note that this is a slight abuse of notation since s_i is not a member of this class, but is instead the property that is common to all members of this class. Hence, $[s_i]$ can be read as the equivalence class for all covers having sensor s_i as their bottleneck sensor.

We now define what we would call the Equivalence Class (EC) Graph. Each node of this graph represents an equivalence class. Just as the LD graph models the dependency between sensor covers, the EC Graph models the dependency between classes of covers.

Definition 2: Equivalence Class Graph (EC). The Equivalence Class graph $EC = (V', E')$ where, V' is the set of all possible equivalence classes defined by $\Re$ and two classes $[s_i]$ and $[s_j]$ are joined by an edge for every cover in each class that share some sensor in common. Hence, the graph EC is a multi-edge graph.

The cardinality of the vertex set of the Equivalence Class Graph is at most n. This result follows from the observation that for any network of n sensors, there can be at most one equivalence class corresponding to each sensor, since every cover can have only one of the n sensors as its bottleneck (in case two or more sensors all have the same battery and are the bottleneck, sensor id's can be used to break ties).

To better understand these definitions, let us consider an example. Consider the sensor network shown in Figure 2. The network comprises of seven sensors, $s_1, ..., s_7$ and two targets, T_1, T_2. Observe that T_2 is the bottleneck target for the network since it is the least covered target (8 units of total coverage compared to 33 for T_1). Also note that only one sensor, s_1 can cover both targets.

For the given network, the set of all possible minimal sensor covers, S is,

$$S = \{\{s_2, \mathbf{s_6}\}, \{s_2, \mathbf{s_7}\}, \{s_3, \mathbf{s_6}\}, \{s_3, \mathbf{s_7}\}, \{s_4, \mathbf{s_6}\}, \{s_4, \mathbf{s_7}\}, \{\mathbf{s_1}\}\}$$

For each individual cover in this set, the bottleneck sensor is the sensor shown in bold face.

Figure 3 shows the Lifetime Dependency graph for these covers. As defined, an edge exists between any two covers that share at least one sensor in common and the weight of this edge is given by the lifetime of the common sensor having the smallest battery (the bottleneck). For example, an edge of weight 4 exists between C_1 and C_2 because they share the sensor s_2 having a battery of 4.

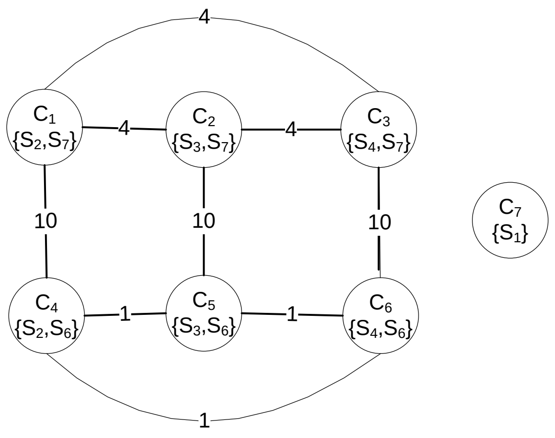

Fig. 3. LD Graph for the example network

To obtain the EC Graph from this LD Graph, we add a node to represent the equivalence class for each sensor that is a bottleneck sensor for any cover. For the above example, given all sensor covers in the set S, there are three sensors s_1, s_6, s_7 that are each the bottleneck for one or more covers in S. Hence, the EC Graph is a three node graph. Figure 4 shows the complete EC Graph for the covers in S. There is a node corresponding to the equivalence class for each of the three sensors s_1, s_6, s_7 and for each cover in the class we retain edges to the class corresponding to the bottleneck sensor of the cover on which the edge terminated in the LD graph. Hence, we have three edges between the nodes s_6 and s_7.

It is key to realize that the EC graph is essentially an encapsulation of the LD Graph that can have at most n nodes.

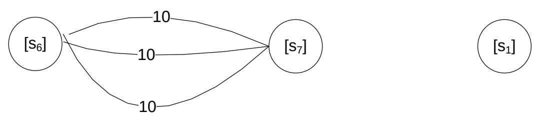

Fig. 4. EC Graph for the example network

This view is presented in Figure 5, where we show the LD Graph that is embedded into the EC Graph. Each rectangular box shows the nodes in the LD graph that are in the same equivalence class. This figure also illustrates our next theorem.

c) Theorem: For sensor covers in the same equivalence class, the induced subgraph on the LD Graph is a clique

Proof: This theorem states that for the nodes in the LD graph that belong to the same class, the induced subgraph is a clique. Since by definition, all sensor covers in a class $[s]$ share the sensor s as their bottleneck sensor, the induced subgraph will be a complete graph between these nodes. $\square$

Also, a subtle distinction has been made between inter-class edges and intra-class edges in going from the LD graph to the EC graph.

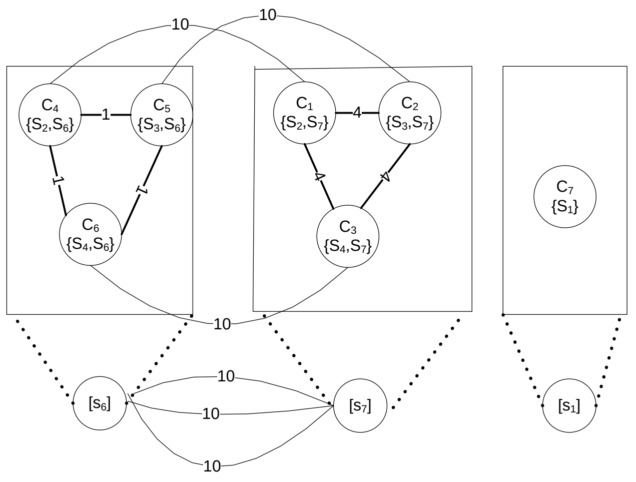

Fig. 5. EC Graph for the example network along with the LD Graph embedded in it

IV. Sampling Based on the Equivalence Class Graph

The previous section defined the concepts behind reducing the exponential space of covers in the LD Graph to the linear space of the EC Graph. In this section, we build on these concepts to discuss techniques for generating a limited number of covers for the LD Graph. Specifically, our goal is to improve the timing performance of the distributed algorithms we presented in [6], [7]. As presented, the EC Graph is not very useful since it still requires the exponential LD graph to be populated, before it can be constructed. However, by realizing that the exponential space of cover sets can be expressed in this linear space of equivalence classes, we can generate only a subset of the set of all covers.

Recall from Section I that even though the number of global sensor covers is exponential in the number of sensors, our heuristics presented in [6], [7] worked by constructing *local* covers. After exchanging one or two hop coverage information with neighboring sensors, a sensor can exhaustively construct all possible local covers. A local cover here is a sensor cover that covers all the local targets. The number of local covers is also exponential but is determined by the maximum degree of the graph and the number of local targets, typically much smaller values than the number of all sensors or targets. The heuristics then construct the LD graph over these local covers. The choice of which cover to use is determined by looking at properties of the LD graph such as the degree of each cover in the LD graph.

By making use of the idea of related covers in the same equivalence class, our goal is to use our existing heuristics from [6], [7] but to modify them to run over a subset of the local covers as opposed to all local covers. This should give considerable speedup and if the subset is selected carefully, it may only result in a slight reduction of the overall lifetime. We present such a local cover sampling scheme in Section IV-A and then present the modified basic algorithm of [6], [7] to operate on this sample in Section IV-B. Finally, we evaluate the effectiveness of sampling in Section V.

A. Local Bottleneck Target based generation of local cover sets

Understanding the underlying equivalence class structure, we now present one possible way of generating a subset of the local cover sets. Our approach is centered around the bottleneck target. For any target t_i, the total amount of time this target can be monitored by any schedule is given by:

$$lt(t_i) = \sum_{\{s \mid t_i \in T(s)\}} b(s)$$

Clearly there is one such target with the smallest $lt(t_i)$ value, and is hence a bottleneck for the entire network [12]. Without global information it is not possible for any sensor to determine if the global bottleneck is a target in its vicinity. However, for any sensor s, there is a least covered target in $T(s)$ that is the *local* bottleneck. A key thing to realize is the fact that the global bottleneck target is also the local bottleneck target for the sensors in its neighborhood. Hence, if every sensor optimizes for its local bottleneck target, then one of these local optimizations is also optimizing the global bottleneck target. We use t_{bot} to denote this local bottleneck target. Let C_{bot} be the set of sensors that can cover this local bottleneck target.That is,

$$C_{bot} = \{s \mid t_{bot} \in T(s)\}$$

d) Implementation: This understanding of bottleneck targets, along with our definition of equivalence classes, now gives us a simple means to generate local covers. Since no coverage schedule can do any better than the total amount of time that the global bottleneck can be covered, instead of trying to generate all local covers, what we really need are covers in the equivalence classes corresponding to each sensor $s_i \in C_{bot}$, such that each class can be completely exhausted. Also, to only select covers that conserve the battery of the sensors in C_{bot}, we want to ensure that the covers we generate are disjoint in C_{bot}. In terms of equivalence classes, for any two classes $[s_i]$ and $[s_j]$ such that $s_i, s_j \in C_{bot}$, we want to generate cover sets that are in these classes but do not include *both* s_i and s_j.

To generate such cover sets, we can start by picking only one sensor s'_{bot} in C_{bot}. This ensures that the local bottleneck target is covered. For each target t_i in the one/two-hop neighborhood being considered, we can then randomly pick a sensor s, giving preference to any $s \notin C_{bot}$. Note that this does not necessarily create a sensor cover in the class $[s'_{bot}]$, since any one of our randomly picked sensors could be the bottleneck for the cover generated. However, replacing that sensor with another randomly picked sensor that covers the same target ensures that the we finish by using a cover in $[s'_{bot}]$. Such a selection essentially ensures that we burn the entire battery of this sensor s'_{bot} in C_{bot} through different covers, while trying to avoid using other sensors in C_{bot}. This process is then repeated for every sensor in C_{bot}. Hence, instead of generating all local covers, we only generate a small sample (constant number) of these corresponding to the equivalence class for each sensor covering the bottleneck target and some related randomly picked covers. We already showed that there can be at most n equivalence classes for the network. Thus, the sampled graph generated has O(n) nodes. If we consider the maximum number of sensors covering any target as a constant for the network, sampling only takes cumulative time of $O(n\tau)$, where $\tau = max_{s \in S}|T(s)|$, since we do this for n sensors, each of which has a maximum of τ targets to cover, which are in turn covered by a constant number of sensors (as per our assumption). Even if this assumption is removed, in the worst case, all n sensors could be covering the same target making the time complexity $O(n^2\tau)$. Next, we run our basic heuristic from [6] on this sampled LD graph as explained in the next section.

B. Modified Basic Algorithmic Framework [6]

Our distributed algorithms consist of a initial setup phase followed by rounds of predetermined duration during which sensors negotiate with their neighbors to determine their sense/sleep status.

Setup: In the setup phase, each sensor s communicates with each of its neighbor $s' \in N(s, 1)$ exchanging battery levels $b(s)$ and $b(s')$, and the targets covered $T(s)$ and $T(s')$.

In [6] our heuristic finds all the local covers using the sensors in $N(s, 1)$ for the target set being considered. The latter can be solely $T(s)$ or could also include $T(s')$ for all $s' \in N(s, 1)$. It then constructs the local LD graph $LD = (V, E)$ over those covers, and calculates the degree $d(C)$ of each cover $C \in V$ in the graph LD.

Instead of this approach, we generate only a sample of all the local covers using the ideas explained in the previous subsection. Our goal is to see if we can achieve a significant speedup with some reasonable trade off in network lifetime. The remainder of the heuristics phases remain unchanged from before keeping in mind that they operate on the LD graph of the sample of all local covers.

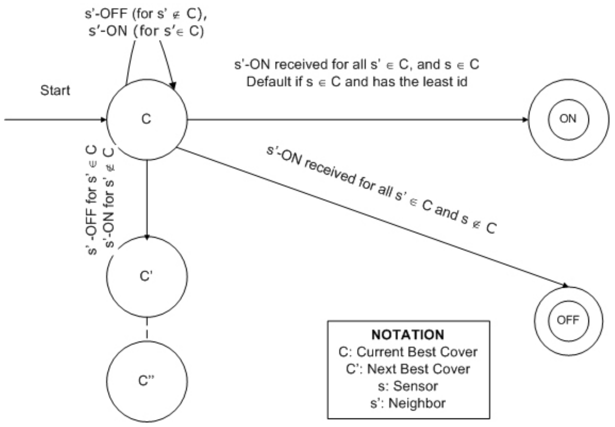

Fig. 6. The state transitions to decide the sense/sleep status

Prioritize and Negotiate solutions:

Once the LD graph has been sampled by each sensor, it needs to decide which cover to use. In order to do this, a *priority function* can be defined to prioritize the local covers. We base the priority of cover C on its degree *d(C)*. A lower degree is better since this corresponds to a smaller impact on other covers. Note that the priority function is computed at the beginning of every round by exchanging current battery levels among neighbors since the degrees may have changed from the previous round.

After calculating the priority function, the goal is to try and satisfy the highest priority cover. However, a cover comprises of multiple sensors and if one of these switches off, this cover cannot be satisfied. Hence, each sensor now uses the automaton in Fig. 6 to decide whether it can switch off or if it needs to remain on. The automaton starts with every sensor s in its highest priority cover C. The sensor s keeps trying to satisfy this cover C and eventually if the cover C is satisfied, then s switches on if $s \in C$ else s switches off. If a cover C cannot be satisfied, then the sensor s transitions to its next best priority cover C', C'' and so on, until a cover is satisfied. We also show the correctness and deadlock/livelock free properties of this heuristic in [6].

As shown in [6], the number of covers in the local LD graph is given by $O(\Delta^\tau)$, where $\Delta = max_{s \in S}|N(s, 1)|$ and $\tau = max_{s \in S}|T(s)|$. This is exponential in m, but not in n because of limited number of covers generated for practical reasons by the local LD graph generator. In practice, the number of covers in the local LD graph is small, since Δ and τ are relatively small. But by using sampling based on classes, we are able to reduce this to polynomial in both n and m.

V. Performance Evaluation

In this section, we evaluate the performance of the proposed sampling scheme and evaluate it against our degree based heuristics of [6]. By not constructing all local covers and instead constructing a few covers for key equivalence classes, we should achieve considerable speedup. But the effectiveness of sampling can only be evaluated by analyzing its tradeoff between faster running time for possible reduced performance. The objective of our simulations was to study this tradeoff. For completeness, we create both one-hop and two-hop versions of our sampling heuristic and also compare its performance to two other algorithms in the literature, the 1-hop algorithm LBP [5] and the 2-hop algorithm DEEPS [13]. Details of both these algorithms are given in Section VI.

In order to compare the equivalence class based sampling against our previous degree based heuristics, LBP, and DEEPS, we use the same experimental setup and parameters as employed in [5]. We carry out all the simulations using C++. For the simulation environment, a static wireless network of sensors and targets scattered randomly in $100m \times 100m$ area is considered. We conduct the simulation with 25 targets randomly deployed, and vary the number of sensors between 40 and 120 with an increment of 20 and each sensor with a fixed sensing range of $60m$. The communication range of each sensor assumed to be two times the sensing range [14], [15]. For these simulations, we use the linear energy model wherein the power required to sense a target at distance d is proportional to d. We also experimented with the quadratic energy model (power proportional to d^2). The results showed similar trends to those obtained for the linear model.

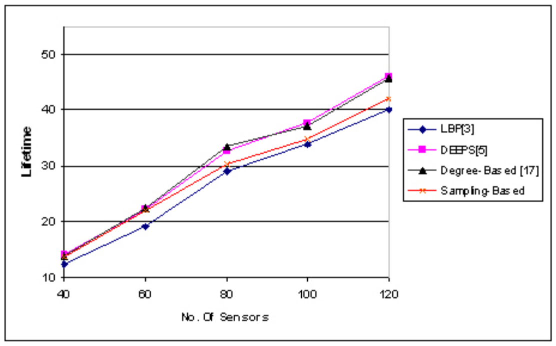

Fig. 7. Comparison of Network Lifetime with 25 Targets

Figure 7 shows the Network Lifetime for the different algorithms. As can be seen from the figure, the sampling heuristics is only between 7-9% worse than the degree based heuristic. Sampling also outperforms the 1-hop LBP algorithm by about 10%. It is interesting to observe that for smaller network sizes, sampling is actually much closer to the degree-based heuristics in terms of performance.

Now that we have seen that sampling works well when compared to the degree based heuristic, the question that remains to be answered is how much faster is the sampling algorithm? Figure 8 compares head-to-head the running time for the degree based heuristic (potentially exponential in m) and the linear time sampling algorithm. As can be seen from the figure the running time for the sampling algorithm is about

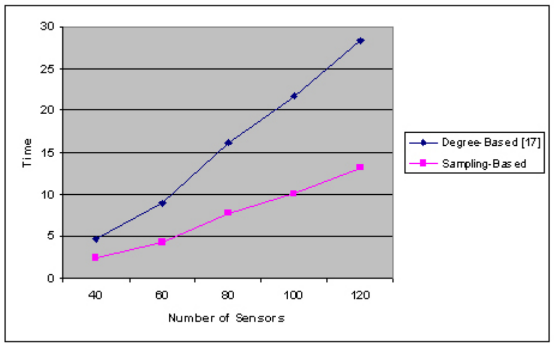

Fig. 8. Comparison of Running Time with 25 Targets

Algorithm	n=40	n=80	n=120
LBP [5]	12.4	29.1	40.1
Degree-Based [6]	13.8	33.4	45.6
Sampling-Based	13.7	30.3	42.1
Randomized-Sampling	10.1	17.6	30.1

TABLE I
COMPARISON OF NETWORK LIFETIME FOR 1-HOP ALGORITHMS

half of the running time for the degree-based heuristic. This makes it clear that for a reduction in lifetime of 8-10%, the new heuristic using equivalence class based sampling achieves a two-fold speedup in running time.

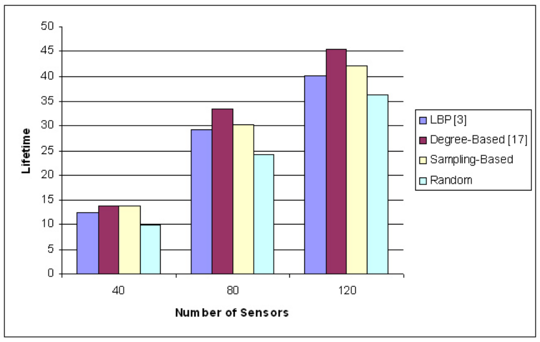

Fig. 9. Chart for comparison of Network Lifetime for 1-hop algorithms

Finally, we individually study the 1-hop (Figure 9, Table I) and 2-hop (Figure 10, Table II) sampling heuristics with comparable algorithms. For the 1-hop algorithms, we also include a randomized-sampling algorithm that makes completely random picks for each target, without cosidering properties of the equivalence classes. The intention is to ensure that the performance of our sampling-heuristic can be attributed to the selection algorithm. For the 2-hop versions of our proposed sampling heuristic, the target set $T(s)$ of each sensor is expanded to include $\cup_{s' \in N(s,1)} T(s')$ and the neighbor set is expanded to all 2-hop neighbors, i.e., $N(s, 2)$. Covers are now constructed over this set using the same process as before. As can be seen from the figure, both the 1-hop and 2-hop version are under 10% worse than the comparable degree-based heuristics. Also, the 2-hop sampling slightly outperforms the DEEPS by a 5% improvement in network

Algorithm	n=40	n=80	n=120
DEEPS [13]	14.1	32.7	46.1
Degree-Based (2-hop) [6]	15.2	36.2	49.6
Sampling-Based (2-hop)	14.4	33.4	47.5

TABLE II
COMPARISON OF NETWORK LIFETIME OF 2-HOP ALGORITHMS

lifetime.

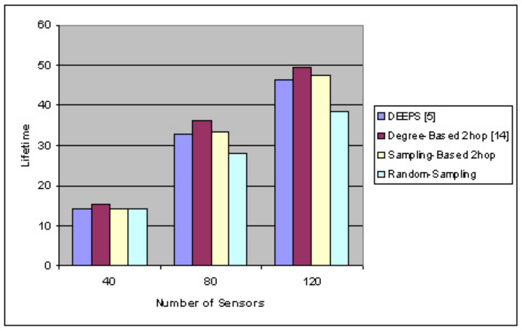

Fig. 10. Chart for comparison of Network Lifetime for 2-hop algorithms

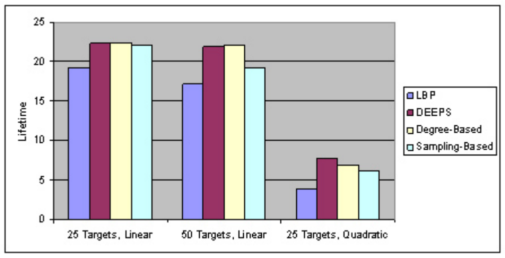

Fig. 11. Chart for comparison of Network Lifetime for linear and quadratic energy models

Additional experiments were also carried out with the quadratic model. We compared different variations in the number of targets (25 and 50) and the energy model (linear and quadratic) while keeping the number of sensors constant. The graph in Fig. 11 shows an experimental setup with the number of sensors fixed to $n = 60$ while the number of targets and energy model vary as shown. The trends of a $7 - 9\%$ degradation in performance over the Degree-based algorithm while still showing an improvement over LBP is preserved even with the quadratic model. We also include the 2-hop algorithm DEEPS in the figure to provide an additional point of comparison. The running time with the quadratic energy model also shows a similar two-fold improvement to the linear-energy model plot in 8.

VI. RELATED WORK

In this section, we briefly survey existing approaches to maximizing the lifetime of a sensor network, while meeting certain coverage objectives. [16] gives a more detailed survey on the various coverage problems and the scheduling mechanisms they use. We end this section by focusing on two algorithms, LBP [5] and DEEPS [13], which we used for comparisons against our algorithms in Section V.

Coverage problems have long existed in other fields of work. The Art Gallery Problem [17], looks at the question of deploying observers in an art-gallery such that every point in the room is seen by at least one observer. There is a linear time solution provided for the 2D case but the 3D version of the problem is shown to be NP=hard and an approximation algorithm is given in [18]. [19] looks at problem of covering an ocean for satellited based monitoring of phytoplankton abundance. They show that having three satellites observe the region and merging their data results in 50 % improvements in coverage but adding more satellites produce minimal improvements over this. Coverage has also been looked at in robot-systems by [20]. The author introduces the notions of blanket, barrier and sweep coverage. Blanket coverage attempts to maximize the total area covered. Barrier coverage arranges nodes in a manner that minimizes the probability of undetected intrusion. Sweep coverage is a mobile barrier detection problem. In particular, the definition of barrier coverage introduced here has been used in the context of barrier coverage for wireless sensor networks in [21]–[25].

The coverage problem has been shown to be NP-complete in [2], [3]. The approach taken in the literature has evolved around algorithmic techniques commonly used to work with NP-complete problems. This is indicated by the use of solutions to related problems such as that of finding domatic partitions, coloring, set covers, etc, as applied to coverage. Cardei et al. [16] give a detailed survey on the various coverage problems and the scheduling mechanisms they use. [26] also surveys the coverage problem along with other algorithmic problems relevant to sensor networks.Initial approaches to maximizing the lifetime in [2], [3], [11] considered the problem of finding the maximum number of *disjoint* covers. However, [27], [28] and others showed that using non-disjoint covers allows the lifetime to be extended further and this has been adopted since. We briefly survey the existing work by classifying it into the categories of centralized and distributed algorithms. [29] present some insights into different factors influencing the lifetime.

A common approach taken with centralized algorithms is that of formulating the problem as an optimization problem and using linear programming (LP) to solve it [3], [28], [30], [31]. [30] formulates the area coverage problem using a Integer LP and relax it to obtain a solution. [3] proves that the target coverage problem is NP-complete and model their solution as a Mixed Integer Program. [28] formulates a packing LP for the coverage problem. Using the Garg-Könemann approximation algorithm [32], they provide a $(1+\epsilon)(1+2lnn)$ approximation of the problem, where n is the number of sensors. A similar problem is solved by us for sensors with adjustable ranges [31]. [2] also gives a centralized greedy algorithm that picks sensors based the largest uncovered area.

The distributed algorithms in the literature can be further classified into greedy, randomized and other techniques. The greedy algorithms [2], [4], [5], [11], [13], [27] all share the common property of picking the set of active sensors greedily based on some criteria. [11] considers the area coverage problem and introduces the notion of a *field* as the set of points that are covered by the same set of sensors. The basic approach behind the picking of a sensor is to first pick the one that covers that largest number of previously uncovered fields and to then avoid including more than one sensor that covers a sparsely covered field. [2] builds on this work. The greedy heuristic they propose works by selecting the sensor that covers the largest uncovered area. [4] defines the sensing denomination of a sensor as its contribution, i.e., the area left uncovered when the sensor is removed. Sensors with higher sensing denomination have a higher probability of remaining active. [29]

Some distributed algorithms use randomized techniques. Both OGDC [14] and CCP [15] deal with the problem of integrating coverage and connectivity. They show that if the communication range is at least twice the sensing range, a covered network is also connected. [2], [14], [15] also present randomized algorithms. [10] gives an algorithm for a node to compute its sponsored area. To prevent the occurrence of blind-points by having two sensors switch off at the same time, a random back off is used. [33] present distributed algorithms that ensure with very high probablity that all targets are covered, provided some initial assumptions on the shape of the deployment area are met.

Now, we look at the two protocols closest to our approach. We compared our heuristics against these in the preceding section. The load balancing protocol (LBP) [5] is a simple one-hop protocol which works by attempting to balance the load between sensors. Sensors can be in one of three states sense, sleep or vulnerable/undecided. Initially all sensors are vulnerable and broadcast their battery levels along with information on which targets they cover. Based on this, a sensor decides to switch to off state if its targets are covered by a higher energy sensor in either on or vulnerable state. On the other hand, it remains on if it is the sole sensor covering a target. This is an extension of the work in [28]. LBP is simplistic and attempts to share the load evenly between sensors instead of balancing the energy for sensors covering a specific target.

The second protocol we consider is a two-hop protocol called DEEPS [13]. The maximum duration that a target can be covered is the sum of the batteries of all its nearby sensors that can cover it and is known as the life of a target. The main intuition behind DEEPS is to try to minimize the energy consumption rate around those targets with smaller lives. A sensor thus has several targets with varying lives. A target is defined as a $sink$ if it is the shortest-life target for at least one sensor covering that target. Otherwise, it is a $hill$. To guard against leaving a target uncovered during a shuffle, each target is assigned an in-charge sensor. For each sink, its in-charge sensor is the one with the largest battery for which this is the shortest-life target. For a hill target, its in-charge is that neighboring sensor whose shortest-life target has the longest life. An in-charge sensor does not switch off unless its targets are covered by someone. Apart from this, the rules are identical as those in LBP protocol.

VII. Conclusions

In this paper we introduced some key ideas on dealing with the exponential space of sensor covers for the maximum lifetime sensor scheduling problem. Our approach was based on defining a relation that partitions the set of all sensor covers into a linear number of disjoint partitions. We use this underlying theory to sample this exponential space efficiently and show that heuristics that use this sample achieve significant speedup with only slight reduction in quality. Unlike existing work based on simple greedy algorithms on the sensor network graph, our algorithms are theoretically grounded and make use of the dependency information in the LD graph as well as the reductions of the EC graph, to achieve significant improvements in network lifetime.

As part of our future work we are looking at other variations of sampling heuristics driven by different selection criteria.

References

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Commun. Mag.*, pp. 102–114, 2002.

[2] Z. Abrams, A. Goel, and S. Plotkin, "Set k-cover algorithms for energy efficient monitoring in wireless sensor networks," *Third International Symposium on Information Processing in Sensor Networks*, pp. 424–432, 2004.

[3] M. Cardei and D.-Z. Du, "Improving wireless sensor network lifetime through power aware organization," *Wireless Networks*, vol. 11, pp. 333–340(8), 2005.

[4] J. Lu and T. Suda, "Coverage-aware self-scheduling in sensor networks," *18th Annual Workshop on Computer Communications (CCW)*, pp. 117–123, 2003.

[5] P. Berman, G. Calinescu, C. Shah, and A. Zelikovsky, "Efficient energy management in sensor networks," *In Ad Hoc and Sensor Networks, Wireless Networks and Mobile Computing*, 2005.

[6] S. K. Prasad and A. Dhawan, "Distributed algorithms for lifetime of wireless sensor networks based on dependencies among cover sets," in *HiPC: 14th International Conference on High Performance Computing, LNCS 4873*, 2007, pp. 381–392.

[7] A. Dhawan and S. K. Prasad, "Energy efficient distributed algorithms for sensor target coverage based on properties of an optimal schedule," in *HiPC: 15th International Conference on High Performance Computing, LNCS 5374*, 2008.

[8] ——, "A distributed algorithmic framework for coverage problems in wireless sensor networks," *Procs. Intl. Parallel and Dist. Processing Symp. Workshops (IPDPS), Workshop on Advances in Parallel and Distributed Computational Models (APDCM)*, pp. 1–8, 2008.

[9] ——, "A distributed algorithmic framework for coverage problems in wireless sensor networks," *To Appear in International Journal of Parallel, Emergent and Distributed Systems(IJPEDS)*, 2009.

[10] D. Tian and N. D. Georganas, "A coverage-preserving node scheduling scheme for large wireless sensor networks," in *WSNA: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*. New York, NY, USA: ACM, 2002, pp. 32–41.

[11] S. Slijepcevic and M. Potkonjak, "Power efficient organization of wireless sensor networks," *IEEE International Conference on Communications (ICC)*, pp. 472–476 vol.2, 2001.

[12] S. Schmid and R. Wattenhoffer, "Maximizing the lifetime of dominating sets."

[13] D. Brinza and A. Zelikovsky, "Deeps: Deterministic energy-efficient protocol for sensor networks," *Proceedings of the International Workshop on Self-Assembling Wireless Networks (SAWN)*, pp. 261–266, 2006.

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

[15] G. Xing, X. Wang, Y. Zhang, C. Lu, R. Pless, and C. Gill, "Integrated coverage and connectivity configuration for energy conservation in sensor networks," *ACM Trans. Sen. Netw.*, vol. 1, no. 1, pp. 36–72, 2005.

[16] M. Cardei and J. Wu, "Energy-efficient coverage problems in wireless ad hoc sensor networks," *Computer Communications*, vol. 29, no. 4, pp. 413–420, 2006.

[17] J. O'Rourke, *Art gallery theorems and algorithms*. New York, NY, USA: Oxford University Press, Inc., 1987.

[18] M. Marengoni, B. A. Draper, A. Hanson, and R. Sitaraman, "A system to place observers on a polyhedral terrain in polynomial time," pp. 773–780, 2000.

[19] W. Gregg, W. Esaias, G. Feldman, R. Frouin, S. Hooker, C. McClain, and R. Woodward, "Coverage opportunities for global ocean color in a multimission era," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 36, no. 5, pp. 1620–1627, Sep 1998.

[20] D. W. Gage, "Sensor abstractions to support many-robot systems," in *Proceedings of SPIE Mobile Robots VII*, 1992, pp. 235–246.

[21] X.-Y. Li, P.-J. Wan, and O. Frieder, "Coverage in wireless ad hoc sensor networks," *IEEE Transactions on Computers*, vol. 52, no. 6, pp. 753–763, 2003.

[22] B. L. Don, "On the coverage and detectability of large-scale wireless sensor networks," 2003.

[23] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. B. Srivastava, "Coverage problems in wireless ad-hoc sensor networks," in *in IEEE INFOCOM*, 2001, pp. 1380–1387.

[24] S. Meguerdichian, F. Koushanfar, G. Qu, and M. Potkonjak, "Exposure in wireless ad-hoc sensor networks," in *Procs. of 7th Annual International Conference on Mobile Computing and Networking (MobiCom'01)*, 2001, pp. 139–150.

[25] C. Zhang, Y. Zhang, and Y. Fang, "Localized algorithms for coverage boundary detection in wireless sensor networks," *Wirel. Netw.*, vol. 15, no. 1, pp. 3–20, 2009.

[26] S. Sahni and X. Xu, "Algorithms for wireless sensor networks," *Intl. Jr. on Distr. Sensor Networks*, vol. 1, 2004.

[27] M. Cardei, M. Thai, Y. Li, and W. Wu, "Energy-efficient target coverage in wireless sensor networks," *INFOCOM 2005*, vol. 3, March 2005.

[28] P. Berman, G. Calinescu, C. Shah, and A. Zelikovsky, "Power efficient monitoring management in sensor networks," *Wireless Communications and Networking Conference (WCNC)*, vol. 4, pp. 2329–2334 Vol.4, 2004.

[29] Q. Xue and A. Ganz, "Maximizing sensor network lifetime: analysis and design guidelines," vol. 2, Oct.-3 Nov. 2004, pp. 1144–1150 Vol. 2.

[30] S. M. Miodrag, "Low power 0/1 coverage and scheduling techniques in sensor networks," *UCLA Technical Reports 030001*, 2003.

[31] A. Dhawan, C. T. Vu, A. Zelikovsky, Y. Li, and S. K. Prasad, "Maximum lifetime of sensor networks with adjustable sensing range," *Proceedings of the International Workshop on Self-Assembling Wireless Networks (SAWN)*, pp. 285–289, 2006.

[32] N. Garg and J. Koenemann, "Faster and simpler algorithms for multicommodity flow and other fractional packing problems." in *FOCS '98: Proceedings of the 39th Annual Symposium on Foundations of Computer Science*. Washington, DC, USA: IEEE Computer Society, 1998, p. 300.

[33] G. Calinescu and R. B. Ellis, "Monitoring schedules for randomly deployed sensor networks," in *DIAL M-POMC '08: Proceedings of the fifth international workshop on Foundations of mobile computing*. New York, NY, USA: ACM, 2008, pp. 3–12.