Energy Efficient Monitoring in Sensor Networks *

Amol Deshpande† Samir Khuller‡§ Azarakhsh Malekian‡ Mohammed Toossi

Abstract

In this paper we study a set of problems related to efficient energy management for monitoring applications in wireless sensor networks. We study several generalizations of a basic problem called Set k-Cover. The problem can be described as follows: we are given a set of sensors, and a set of regions to be monitored. Each region can be monitored by a subset of sensors. The goal is to partition the sensors into k sets (or time-slots), so that by activating the set of sensors in a time-slot, we can maximize coverage of the regions. By activating each sensor in only one of the k time slots, we decrease its battery consumption and extend its battery life significantly (by a factor of k). This problem is known to be NP-hard. Our goal is to develop improved approximation algorithms for this problem. Moreover, we are able to demonstrate that this algorithm is practical, and yields almost optimal solutions in practice.

We also consider generalizations of this problem in several different directions. First, we allow each sensor to be active in α different sets (time-slots). This means that the battery life is extended by a factor of $\frac{k}{\alpha}$, and allows for a richer space of solutions. We also consider different coverage requirements, such as requiring that the regions be covered in each time slot, or a certain number of regions be monitored in each time slot. In the Set k-Cover formulation, there is no requirement that a region be monitored at all, or in any number of time slots. We develop a randomized rounding algorithm for this problem.

We also consider extensions where each sensor can monitor only a bounded number of regions, and not all the regions adjacent to it. This kind of problem may arise when a sensor has a directional camera, or some other physical constraint might prevent it from monitoring all adjacent regions even when it is active. We develop the first approximation algorithms for this problem.

*Department of Computer Science. University of Maryland, College Park, MD 20742, USA. Email: {amol,samir,malekian,toossi}@cs.umd.edu

†Research supported by NSF grants CNS-0509220 and IIS-0546136

‡Research supported by NSF grant CCF-0430650.

§Institute for Advanced Computer Studies. University of Maryland, College Park, MD 20742, USA.

1 Introduction

Efficient energy management in sensor networks is a primary challenge as we expect wireless devices to communicate and to continue to function effectively for long periods of time. In this paper we study a basic set of problems dealing with energy efficient monitoring in sensor networks. One particular question of this type was first formalized in a paper by Slijepcevic and Potkonjak [11], in which they asked for a collection of disjoint set covers.

Let $H = (S \cup R, E)$ denote a bipartite graph in which nodes in set S correspond to sensors, and nodes in set R correspond to regions. There is an edge between node s_i and node r_j if sensor s_i can monitor region r_j. By keeping all sensors activated all the time, clearly all the regions can be monitored continuously (assuming no nodes in R have zero degree). The problem with this solution is that the sensors may not last very long. One might now wonder if a better solution can be obtained. There are multiple ways in which we could formulate this problem. One approach is to partition the nodes in S into k sets $S_1, \ldots, S_k$, such that the sensors in set S_i cover all the regions in R. This is also referred to as the domatic set problem [6] for which a randomized approximation algorithm with factor $O(\log n)$ has been proposed. An alternative formulation (called Set k-cover), studied by Abrams et al. [1], instead asks for a partitioning that maximizes the total regions covered. More formally, let $R_i \subseteq R$ denote the regions covered by the sensors in S_i; in other words, $R_i = \{r | (s,r) \in E \wedge s \in S_i\}$. The goal is to maximize $\sum_{i=1}^{k} |R_i|$.

Given such a partitioning, the idea then is to cycle through the k sets S_i in a round-robin fashion. When we activate all sensors in set S_i, we

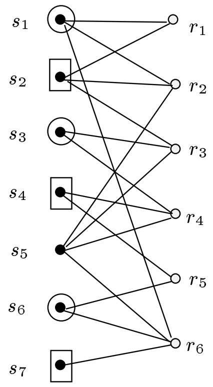

Figure 1: Graph H and solution for $k = 3$ is $S_1 = \{s_2, s_4, s_7\}, S_2 = \{s_1, s_3, s_6\}$ and $S_3 = \{s_5\}$ and $R_1 = \{r_1, r_2, r_3, r_4, r_5, r_6\}$, $R_2 = \{r_1, r_2, r_3, r_4, r_5, r_6\}$ and $R_3 = \{r_2, r_3, r_4, r_6\}$.

cover the regions in R_i. Thus the objective function tries to maximize the coverage of the regions in the different time slots. We will consume significantly less energy this way, as each sensor is activated in only one of the k sets. Moreover there is evidence to suggest that the battery life of sensors is increased significantly when batteries are used in short bursts, rather than being used continuously [3].

At the same time we will maximize the coverage over time. *Ideally*, each region will belong to R_i for each value of i. Notice that we have relaxed the requirement that each region is always monitored. If k is not very large it is entirely possible that we can actually monitor all regions at all times. Of course, the larger the value of k, the longer we extend the lifetime of the system, while paying a penalty of lowering the coverage level within each time-slot. It also depends on the redundancy of coverage, in other words, it also depends on how many sensors are monitoring each region.

In this paper, we identify and address several generalizations of this problem, many of which, to our best knowledge, have not been addressed before. One generalization we define is called the Set (k, α)-Cover problem. We would like to find k sets, $S_1, S_2, \ldots, S_k$. As before, $S_i \subset S$. We require that each vertex in S belong to at most α such sets. More formally, for all $s_j \in S$ we require that $|\{i | s_j \in S_i\}| \leq \alpha$. A solution for this problem can be mapped to a sensor schedule in which each sensor is active in α of the k time-slots. We have relaxed the requirement that the S_i sets are disjoint which corresponds to the case when $\alpha = 1$. Since this is a generalization of the set k-cover problem, it is also NP-hard.

Since each sensor now belongs to α sets, the battery life is extended by a factor of $\frac{k}{\alpha}$. Our main goal really is to maximize the battery life, subject to adequate coverage requirements. However, for the development of the algorithms, it is easier to fix the parameters k and α and to develop algorithms that work with these parameters. In practice one would consider a spectrum of solutions produced by our algorithms for different choices of k and α.

We then consider a generalization where sensors have *capacity constraints*. In many scenarios, even though a sensor may be able to cover multiple regions, at any given time, it may only be able to actually monitor one or a small number of them. For example, a pan-and-tilt camera can monitor only one region (or a few regions) at any time. We call this a capacity constraint, and denote by $c(s_i)$ the maximum number of regions sensor s_i can cover in one time slot. The goal is to solve the Set k-Cover or Set (k, α)-cover problems given such capacity constraints.

Finally, given a k or a (k, α) pair, a sensor cover problem asks for the best partition of the sensors (along with a designation of which regions to monitor for the capacity-constrained version) that optimizes some coverage property. The optimization goal itself could be one of the following three:

- **avg-coverage:** The average coverage over the k time slots. Formally this would be $\frac{\sum_{i=1}^{k} |R_i|}{k}$.
- **min-coverage(time):** The minimum value of the fractional coverage in any time slot. Formally, this would be $\min_{i=1}^{k} \frac{|R_i|}{|R|}$.
- **min-coverage(region):** The minimum over all regions, of the fraction of time a region is covered. This is important for application

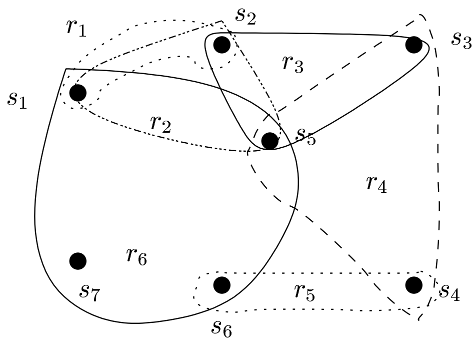

Figure 2: Hyper edge structure for the example from Fig. 1.

where high coverage of regions is required over time.

Different applications may demand support for different optimization goals. For example, for the min-coverage(time) version, we could fix a coverage requirement, by specifying that $|R_i| \geq \gamma|R|$ for some $1 \geq \gamma > 0$ and ask to minimize α. Under this classification, Abrams et al. [1] study the {Set k-cover, NC, avg-coverage} version of the problem (where NC denotes that there are no capacity constraints).

We now consider a slightly different view of the Set k-Cover problem. Given the bipartite graph H describing the sensor-region relationship, we construct the following hypergraph $G = (V, E)$. Each node in V corresponds to a sensor. For each region r, we create a hyperedge e that contains the set of sensors that cover the region. We assume that each region is covered by at most d sensors, i.e., the hyper edge has size d. The goal now is to color the nodes of the graph with k colors (this is simply a way to view the partitioning into k sets). The objective is to maximize the total *benefit* of all hyper edges. The benefit of a hyper edge is the number of different colors that the nodes in the hyper edge are colored with. If all nodes in this hyper edge have the same color, then the benefit is 1 since they are all in the same set. If the nodes have k different colors, then this region is always monitored and its benefit is k.

The practical problem is of interest for small d, so it is worth studying this case in more detail, since we do not expect too many sensors to cover the same region, otherwise this suggests that the density of sensors is too high. For $d = 2$ this problem is clearly related to the well studied max k-cut problem for which a SemiDefinite Programming (SDP) based algorithm does very well [7, 8]. The max k-cut problem asks for a partitioning of the vertices of a graph into k groups so as to *maximize the number of edges across the cut* (edges connecting vertices in two different groups). While both problems ask for a partitioning of the vertices, the precise objective functions are different.

Outline of Contributions:

Our first algorithm (see Section 3) shows how to "reduce" the set k-cover problem to the max k-cut problem and then apply known approximation methods for the latter. In fact, this approach gives rise to an extremely fast and practical method to solve the problem. We also prove some improved approximation factors for small d using this approach (see Section 3). For $d \leq 3$, this gives significantly improved worst case approximation factors compared to the randomized approach in [1]. However, the key point is that this approach gives almost optimal solutions in practice, even for larger values of d. We also present a worst case analysis of this method for large d.

In addition, we are also able to develop a *direct* SDP based algorithm (see Appendix A and A.1) for this problem (the same approach was used to develop algorithms for max k-cut). This gives rise to an algorithm that runs in polynomial time for constant d. The solutions produced are almost as good as the solutions produced by the max k-cut approach but the algorithm is slower compared to the direct reduction to max k-cut. However we believe that this approach will eventually give a better worst case approximation bound for the case when the hyper-edges are large.

In Section 4 we develop an LP based randomized rounding algorithm for approximating the lifetime of the network for the {set (k, α)-cover, NC, min-coverage (time)} version of the problem (an extension to the min-coverage (region) case is straightforward). We fix a k, α pair. Assuming that a feasible integral solution exists, we are guaranteed to find a feasible fractional solution. Once we obtain the fractional solution, we round it. We use a scaling parameter s to do the rounding. This may increase the number of sets a node belongs to, with the expected number being $s\alpha$. We can prove that the probability that each region is covered is very high and at least $1 - \frac{1}{e^s}$.

For the case where each sensor can only cover one region (unit capacity) when it is active, we develop a polynomial time algorithm that computes an optimal solution for the capacitated set (k, α)-cover problem (see Section 5). For the case of *arbitrary* capacities we develop a polynomial time $(1 - \frac{1}{e})$ approximation since we can show that the problem is NP-hard even when the capacities are three.

In Section 6 we illustrate the data sets as well as the performance of our set k-cover method. On the data sets in which we were able to compute the optimal solution, we found that our approach gave almost optimal solutions almost every time.

In addition, we also implemented and tested our algorithms for the version of the problem with capacities. For this case, we simply compare the quality of our solution to an *upper* bound on the optimal solutions since we were unable to compute the optimal solutions. Again, we found that the algorithms performance is significantly better than the worst case bounds we derived in Section 5.

One interesting feature about our algorithms is that they take the structure of the problem into account. The randomized approach in [1] is oblivious to the actual graph structure.

2 Prior Work

Designing sleep schedules to maximize the network lifetime while guaranteeing coverage has been one of the most active research areas in wireless sensor networks. Cardei and Wu [5] survey the work in this area, and identify two types of coverage problems, *area* coverage (where the goal is to cover maximally cover the area the sensor network is deployed in), and *target* coverage (where the goal is to cover a set of targets).

The problem we address in this paper can be seen as a target coverage problem, and we briefly review the prior work on this problem. Slijepcevic and Potkonjak [11] pose the problem with full coverage requirement; given a sensor network, the goal is to identify mutually exclusive sets of sensor nodes such that the members of each set cover the monitored regions (targets) completely. They provide several heuristics for this problem. Abrams, Goel and Plotkin [1] develop approximation algorithms for the Set k-Cover problem, where k is provided, and the goal is to find a partitioning that maximizes the coverage. They present a simple randomized algorithm, where each sensor is assigned to one of the k sets, and they show that the resulting solution approximates the optimal solution within a factor of $1-\frac{1}{e}$. In fact their bound is 0.75 when $k = 2, d = 2$ and approaches $(1 - \frac{1}{e})$ for large d and k. They also show that it is NP-hard to get a polynomial time approximation algorithm with a factor better than $\frac{15}{16} + \epsilon$ for any $\epsilon > 0$. This is shown by a direct reduction from the E4-SET SPLITTING problem for which a $\frac{7}{8} + \epsilon$ hardness has been shown by Hastad [9]. However, the gap between $1 - \frac{1}{e}$ and $\frac{15}{16}$ is significant and our goal is to try and consider other approaches that can be used to narrow this gap further. They also present a distributed greedy algorithm that is a $\frac{1}{2}$ approximation for the problem. Cardei et al. [4] consider a version of the problem that is similar to the Set (k, α)-cover problem, where they allow sensors to belong to multiple sets, and allow the sets to be active for different durations. They present several heuristics for solving the problem. Another related paper is [10] in which the unit capacity case is considered.

3 Max k-cut approach

We now discuss the Set k-Cover problem. We are given a hyper graph $G = (V, E)$, where each node in V corresponds to a sensor. For each region r

we create a hyper edge e that contains the set of sensors that cover the region. The goal now is to color the nodes of the graph with k colors. The objective is to maximize the total *benefit* of all hyper edges. The benefit of a hyper edge is the number of different colors that the nodes in the hyper edge are colored with.

Our algorithm works as follows. We replace each hyper edge $e = \{a_1, \dots, a_p\}$ by edges (a_i, a_j) for $i \neq j$, essentially replacing each hyper edge by a clique (see Fig. 3). We then apply the SDP based Max k-Cut approximation algorithm [7] that tries to maximize the number of edges across the cut after partitioning the vertices into k sets. We use the partitioning produced by this algorithm, even though our original objective function is different. Let α_k be the approximation ratio of the SDP based algorithm [7] for Max k-Cut. Frieze and Jerrum showed that α_k satisfies the following.

(i) $\alpha_k > 1 - \frac{1}{k}$

(ii) $\alpha_k - (1 - \frac{1}{k}) \sim 2\frac{\ln k}{k^2}$

(iii) $\alpha_2 \geq 0.878$, $\alpha_3 \geq 0.8$, $\alpha_4 \geq 0.85$, $\alpha_{10} \geq 0.926$, $\alpha_{100} \geq 0.99$.

In Appendix A we give a description of how the Frieze and Jerrum algorithm works, and the precise randomized rounding method we employed for our experiments. This will make it easier for the reader to undertstand our SDP formulation since it uses similar notation to the one used by Frieze and Jerrum.

In the next two subsections we present a worst case analysis of this method. We first give a simple analysis for the case $d = 2$ and this will convey some intuition about why this scheme works well. The analysis for general d is more complex, and we present it subsequently.

3.1 Analysis for $d = 2$

In the Max k-Cut problem the goal is to partition the vertices into k sets to maximize the number of edges whose end points are in different sets. By using an α_k approximation for Max k-Cut, we are able to obtain an approximation guarantee of $\frac{1}{2}(1 + \alpha_k)$.

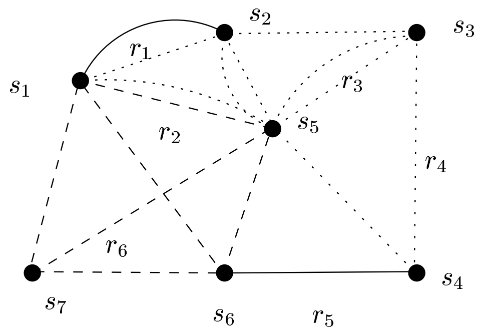

Figure 3: Converting the hyper graph into a multi-graph for the example from Fig. 2.

We know that we will get at least $\alpha_k E^*$ edges across the cut once we run the Max k-Cut algorithm, where E^* is the number of edges across the cut in an optimal solution for Max k-Cut (which is derived from an optimal solution for set k-cover, each region that is monitored in both time slots is essentially an edge across the cut). The optimal solution has total benefit $(E - E^*) + 2E^* = E + E^*$. We get a benefit of at least $2\alpha_k E^* + (E - \alpha_k E^*) = E + \alpha_k E^*$. Taking the ratio, and using the fact that $E^* \leq E$ gives the desired bound. This is significantly better than the oblivious approach of randomly coloring the nodes, regardless of the structure of the graph. If we plug in $\alpha_2 = 0.878$ [8] then we obtain a bound of 0.939. If we plug in $\alpha_3 = 0.8$ [7], we get a bound of 0.9. (In contrast the randomized method of [1] gives a bound of $(1 - \frac{1}{2k})$, which is 0.75 and 0.83 for $k = 2, 3$. Note that α_k also improves as k increases [7].

3.2 Analysis for general d

Recall that we constructed a new multi-graph $G' = (V', E')$ based on the hyper graph $G = (V, E)$. $V' = V$ but instead of each hyper edge $e \in E$, we add edges between all pair of vertices belonging to hyper edge e. We then use the max k-cut algorithm due to Frieze and Jerrum [7] on the graph G'. The next theorem presents the approximation ratio obtained by the max k-cut approach.

The proof is more complicated for when all the hyper-edges have arbitrary sizes. We illustrate the proof for the simpler case when all hyper-edges have the same size d. We show the proof for the case $k \geq d$ since the proof is simpler. Some modifications are required for arbitrary k.

Theorem 1. *The benefit obtained from the method based on Max k-cut is at least $\frac{1}{d} + \frac{\alpha_k}{2}(1 - \frac{1}{d})$ fraction of the maximum benefit for the set k-cover problem.*

Before starting the analysis, we need to define some notation.

Definitions:
E_i: set of hyper edges that have i different colors in the optimal solution.
E'_i is the set of hyper edges that have i different colors in the solution based on Max k-cut.
E represents the set of hyper edges in G.
C^* is the Max k-cut in G', with E^* the edges across the cut.
C_o is the cut in G' obtained by the optimal solution for set k-cover, with B_o the corresponding benefit function.
C' is the cut obtained by the approximation algorithm for finding a max k-cut in G' and $|C'|$ the benefit for set k-cover, and E' the corresponding edges across the cut in G'.

We first prove that the optimal benefit for Set k-Cover can be upper bounded. In fact, we show that the total benefit in the optimal solution is at most $|E| + \frac{2|E^*|}{d}$ (Lemma 1). We also show that the benefit obtained from the Max k-cut approach is at least $|E| + \frac{\alpha_k |E^*|}{d}$ (Lemma 2). By using these two observations, we can guarantee that the approximation ratio is least $\frac{1}{d} + \frac{\alpha_k}{2}(1 - \frac{1}{d})$.

Lemma 1. *The total benefit (B_o) in the optimal solution for the set k-cover problem is at most $|E| + 2\frac{|E^*|}{d}$.*

Proof. The benefit obtained from the optimal solution is simply $\sum_{i=1}^{d} i|E_i|$. Since $|E| = \sum_{i=1}^{d} |E_i|$ this can be rewritten as $B_o = \sum_{i=1}^{d} |E_i|(i-1) + |E|$. The solution for set k-cover partitions the vertices into k sets. Consider a hyper edge $e \in E$ that has i different colors on the end points in an optimal solution. The minimum number of edges that it can contribute to the cut obtained by the solution is $\binom{d}{2} - \binom{d-i+1}{2}$. This is when we put all $d-(i-1)$ nodes into one partition, and each of the remaining $(i-1)$ nodes into a separate partition. Hence we have: $|C_o| \geq \sum_{i=1}^{d} (\binom{d}{2} - \binom{d-i+1}{2})|E_i|$. Comparing the corresponding coefficients, it can be seen that the largest ratio is $\frac{2}{d}$. So $B_o = \sum_{i=1}^{d} |E_i|(i-1) + |E| \leq \frac{2|C_o|}{d} + |E| \leq \frac{2|E^*|}{d} + |E|$. □

Let us briefly consider a diversion for the case $d = 3$. Lemma 1 shows that $B_o \leq |E| + \frac{2}{3}|E^*|$. Before we prove Lemma 2 in general, we prove a lower bound on the quality of the obtained solution. There are two main reasons for this. The first is that in fact we get a slightly better bound when $d = 3$ as follows (in contrast the bound obtained by [1] is 0.70). The second reason is that it provides some intuition for the case when d is arbitrary, but this proof is easier to understand.

Theorem 2. *For the case when $d = 3$, we obtain an approximation factor of $\frac{1}{3} + \frac{1}{2}\alpha_k$.*

Proof. Essentially each hyper edge of size three is reduced to a triangle. Note that when all three nodes belong to different time slots, the benefit function is 3, and we have 3 edges crossing the cut. When the nodes of this hyper edge belong to two time slots, we also have two edges crossing the cut. This is better than the bound obtained by randomized rounding given in [1] which is 0.703.

Let C' be the cut produced by the max k-cut algorithm, and $|E'|$ the number of edges across the cut. The benefit from the max k-cut approach is $\sum_{i=1}^{3} |E'_i|(i-1) + |E|$. The number of edges across the cut in G' is $|E'|$. Clearly $|E'| = 2|E'_2| + 3|E'_3| = 2(|E'_2| + \frac{3}{2}|E'_3|)$. Hence $\frac{|E'|}{2} = |E'_2| + \frac{3}{2}|E'_3|$. So we can conclude that $|C'| = |E| + |E'_2| + 2|E'_3| \geq |E| + \frac{|E'|}{2} \geq |E| + \frac{\alpha_k |E^*|}{2}$. We also know that $|E| \geq \frac{|E^*|}{3}$. Putting these equations together gives us a lower bound on $\frac{|C'|}{B_o}$. Using the bound from Lemma 1, the benefit from the solution based on the Max

k-cut approach is at least $(\frac{1}{3} + \frac{1}{2}\alpha_k)$ of the maximum benefit in the optimal solution. □

When $d = 3$ our bound depends on α_k and is at least 0.828 for large k. When $k = 3$, α_k is 0.8 and we get a bound slightly better than 0.733.

Lemma 2. *The benefit obtained from the Max k-cut approach is at least $|E| + \frac{\alpha_k|E^*|}{d}$.*

Proof. The benefit from the Max k-cut approach can be formulated as $|C'| = \sum_{i=1}^{d} |E'_i|(i-1) + |E|$. Also $|E'| \le \sum_{j=2}^{d} \sum_{i=\lceil \frac{d}{j} \rceil}^{\lfloor \frac{d}{j-1} \rfloor} (\binom{d}{2} - d(j-1) + \binom{j}{2} i)|E'_i|$. This is obtained basically by spreading out all d nodes into i sets as evenly as possible, with each group having either $\lceil \frac{d}{i} \rceil$ nodes or $\lfloor \frac{d}{i} \rfloor$ nodes. The ratio of the corresponding coefficients of $|E'_i|$'s is at least $\frac{1}{d}$. So we can conclude that $|C'| \ge |E| + \frac{|E'|}{d} \ge |E| + \frac{\alpha_k|E^*|}{d}$. □

Proof Of Theorem 1. Using the above two lemmas and considering the fact that $|E| \ge \frac{|E^*|}{\binom{d}{2}}$ it can be shown that the benefit from the solution based on the Max k-cut approach is at least $\frac{1}{d} + \frac{\alpha_k}{2}(1 - \frac{1}{d})$ fraction of the maximum benefit in the optimal solution. □

4 Set (k, α)-cover Problem

We start with the following Integer Program (IP) formulation (for fixed k, α).

We define a 0/1 variable x_{ij}, for $i = 1 \dots k$, $j = 1 \dots n$. When $x_{ij} = 1$ it means that sensor j is active in time-slot i.

Constraints are as follows

$$\forall j \in S \sum_{i=1}^{k} x_{ij} \le \alpha$$

$$\forall r \in R \ \ \forall i = 1 \dots k \sum_{(r,p) \in E} x_{ip} \ge 1$$

$$x_{ij} \in \{0, 1\}$$

The first constraint simply states that each sensor may belong to at most α time-slots. The second constraint states that each region j is covered in each time slot.

Clearly there is no benefit to increasing a variable beyond 1. This gives us a linear program (LP) which can be solved efficiently.

We now use randomized rounding to obtain an integral solution in which with high probability each region is covered in all the time slots. This is a bicriteria approximation algorithm in a sense that the approximation factor increases when we look for solutions with higher probability for coverage. To do the randomized rounding, we first scale all the x_{ij} variable by a scale factor s and then we round up x_{ij} to 1 with probability equal to the scaled x_{ij}, call the new variable $x'_{ij} = \min(1, s \cdot x_{ij})$. Each sensor will be active in $s\alpha$ time-slots (in expectation); and each region will be covered in each time-slot with probability at least $(1 - \frac{1}{e^s})$. For each sensor $j \in S$, the expected value of $E(\sum_{i=1}^{k} x'_{ij}) = \sum_{i=1}^{k} s x_{ij} = s \sum_{i=1}^{k} x_{ij} \le s\alpha$. So the expected cost of the new integral solution is at most $s \cdot OPT$. Now we show that each region will be covered in all the time slots with high probability. Probability that a given region r in a given time slot i is not covered is equal to the probability that none of the x_{ip} variables $\forall (r,p) \in E$ is rounded up to one. In other words, $\Pr[r \text{ is not covered at time } i] = \prod_{(r,p) \in E} (1 - s x_{ip}) \le \frac{1}{e^s}$. So each region in each time slot will be covered with probability at least $1 - \frac{1}{e^s}$.

4.1 Non-constant integrality gap

One might wonder if there is a rounding that satisfies all the constraints with constant s, thus giving a constant approximation. However, it can be shown there is an non-constant integrality gap for this problem. Consider a matrix M of size $\binom{k}{k/2} \times k$. The rows contain all the binary strings of length k that have exactly $\frac{k}{2}$ 1's. Consider each column as a sensor and each row as a region. So if $M_{ij} = 1$, sensor j will cover region i and otherwise not. Suppose we have k time slots. One (fractional) solution is to activate each sensor fractionally for $\frac{2}{k}$ in each time slot so that each location is covered since we have $\frac{k}{2}$ sensors covering each region. This gives us $\alpha = 2$ for the fractional solution. At each time step, at least $(\frac{k}{2}+1)$ sensors have to be active (otherwise a region is uncovered). Thus the total number of active sensors (summing over all times) is $\Omega(k^2)$. Hence at least one sensor is active in $\Omega(k)$ time slots, and has $\alpha = \Omega(k)$. This shows an integrality gap of $\Omega(k)$, and we can make k as large as $\Omega(\frac{\log n}{\log \log n})$, while still having a polynomial number of regions.

Extensions for the min-coverage(region) are straightforward.

5 Sensors with Capacity Constraints

In many applications, sensors have constraints as to how many locations they can monitor even when the sensor is active. For example, a camera sensor s_i may have the *capability* to monitor a set of regions $N(s_i)$, but in a time slot when s_i is active it can only monitor at most $c(s_i)$ regions. For a fixed camera sensor, in fact $c(s_i)$ may just be 1. For a moving sensor, it is possible that the sensor can cover multiple regions. In this case, we also have to come up with an assignment of each sensor to at most $c(s_i)$ regions in a time-slot when the sensor is active.

5.1 NP-Completeness

We examine several cases and either provide polynomial time algorithms, or approximation algorithms when the problem can be shown to be NP-complete.

We first show that even when the capacities are as low as 3, even the basic problem is NP-complete for $k = 2$ and each region being covered by exactly two sensors.

First recall that Max-Cut[1] is NP-complete for graphs with degree at most 3 [12]. The question is - is there a way to partition the nodes of a graph $G = (V, E)$ into two groups so that at least Δ edges cross the cut?

[1]This is the Max k-cut problem when $k = 2$.

We give a sketch of the reduction. Let $S = \{s_i | v_i \in V\}$. Each edge $(v_i, v_j) \in E$ corresponds to a region in R that has neighbors s_i and s_j. Since each node has degree at most 3, in any case a sensor covers at most 3 regions so with a capacity of 3 each sensor can cover all regions adjacent to it. Set $k = 2$. There is a partition in which the total coverage is at least $|E| + \Delta$ if and only if there is a solution to Max Cut with at least Δ edges across the cut.

5.2 General Capacity

In this case, we consider the set (k, α)-cover problem. Each sensor can be activated in at most α sets and the goal is to maximize the average coverage. Unlike the previous case, each sensor s_i can cover $c(s_i)$ regions in each time slot in which it is active. We develop a randomized $(1 - \frac{1}{e})$-approximation algorithm for this problem. As in the previous section we first construct a bipartite graph. Let $H_c = (S_c, R_c, E'_c)$. For each sensor s_i we create α vertices $s_i^1, \ldots, s_i^\alpha \in S_c$. We put an edge from each of the α copies of a sensor to a region node if that the sensor can cover that region. Next, we select a bounded degree subgraph of H_c with the maximum number of edges. The bound on the degree of each $s_i \in S_c$ is $c(s_i)$ and the bound on the degree of each $r \in R_c$ is k. As in the previous case, we can find the subgraph with the maximum number of edges in polynomial time using network flow.Call the subgraph H_c^*. It is easy to prove that the number of edges in H_c^* is an upper bound on OPT. To actually find the schedule we use the following randomized algorithm:
For each sensor, randomly, choose α of the k available time slots (without replacement).

Theorem 3. *The expected value of the coverage given by the randomized algorithm is at least $1-\frac{1}{e}$ of the number of edges in H_c^*.*

Proof. The argument used here is similar to the one given in [1] with some adaptations to work for the new method. We first compute the probability that a region r is not covered in a given time slot t. We use N_r to denote the set of neighbors of r in H_c^*. Also N_{rs} refers to the copies of sensor s that belongs to N_r.

We first show that the probability that r is not covered in a specified time slot t is $(1-\frac{1}{k})^{N_r}$. For each sensor s, the probability that none of the copies of s belonging to N_{rs}, have been covered it, is at most $(1-\frac{|N_{rs}|}{k})$. For $k \geq 1$, $(1-\frac{|N_{rs}|}{k}) \leq (1-\frac{1}{k})^{|N_{rs}|}$. The probability that a specified region r is not covered at a given time slot t is the probability that it is not covered by any of its neighbors. Since for sensors i, j, the two events that N_{ri} is not covering r and N_{rj} is not covering r are independent events, the probability that r is not covered in t is the product of all these probabilities for all N_{rs} sets. So the probability that r is not covered is bounded by $(1-\frac{1}{k})^{\sum_{s\in S_c}|N_{rs}|} = (1-\frac{1}{k})^{|N_r|}$. Since we know that N_r is the union of all N_{rs} sets for $s \in S$. In other words, $N_r = \cup_{s\in S} N_{rs}$.

The probability that r is covered is at least in each time slot $1-(1-\frac{1}{k})^{N_r}$. Let l_r be the number of sensors covering region r, in our solution. We can see that $E(l_r) \geq k - k((1-\frac{1}{k})^{|N_r|}$. The optimal solution can be bounded by the number of edges in the H_c^* which is $\sum_r |N_r|$. As shown in [1], for each region r, $\frac{E(l_r)}{|N_r|} \geq (1-\frac{1}{e})$ which completes the proof. □

Since $H_c^* \geq OPT$, we have a $(1-\frac{1}{e})$ approximation.

5.3 Unit Capacity

We consider the set (k,α)-cover problem with the objective to maximize average coverage.

We show that this problem can be solved optimally in polynomial time. Our goal is to maximize the number of regions that can be covered. Recall that we need to define k subsets S_i such that each sensor belongs to at most α subsets.

We first construct the following bipartite multi-graph. Let $H = (S, R, E')$. We create a vertex in S for each sensor and a vertex for each region in R. We put α parallel edges between each sensor and region pair if that the sensor can cover the region. We now select a maximum *bounded degree subgraph*[2] of H with the maximum number of edges. The degree bound on sensor nodes in S is exactly α and the degree bound on region nodes in R is k. This problem can be solved in polynomial time for bipartite graphs using network flows.Once we find a maximum subgraph (it is not necessarily unique) H^*, we then find an edge coloring [2] of H^* using at most k colors where k is the maximum degree (since $\alpha \leq k$). An edge coloring of a graph is an assignment of colors to the edges such that no pair of edges that are incident on a common vertex have the same color. Each color class forms a matching in the bipartite graph and corresponds to a time slot. The sensor nodes will be members of the α color classes corresponding to the colors of the edges incident on the sensor nodes.

Theorem 4. *The running time of the algorithm is constrained by the time taken to compute the bounded degree subgraph with the maximum number of edges. This takes $O(n^3)$ time in the worst case where n is the number of vertices in the graph.*

6 Experimental evaluation

In this section, we present the results of a comprehensive experimental study that compares the performance of the algorithms we proposed. The salient points of our study can be summarized as follows:

- The Max k-Cut algorithm for solving the Set k-Cover problem achieves schedules very close to optimal, and much better than the prior randomized algorithm.
- Allowing sensors to belong to multiple sets (Set (k,α)-cover) results in increased lifetime in several cases.
- Our Max-Flow based algorithms for the capacity-constrained sensor cover problem

[2]This problem can be viewed as a generalization of the problem of finding a matching in a graph. Each node v has a upper bound of $b(v)$ on the number of chosen edges in the subgraph. With this restriction we wish to compute a subgraph with the maximum number of edges.

consistently perform significantly better than the theoretical worst case bounds.

We ran experiments on a wide variety of synthetic datasets, and report the results for a representative sample of them (details of datasets in Appendix B).

6.1 Set k-Cover

With our first set of experiments, we compare the performance of the new Set k-Cover algorithms that we developed in this paper. For each of the datasets we generated 20 random instances, and compared the performance of the Max-Cut algorithm and the randomized algorithm by Abrams et al. [1] (called Random henceforth), with the the optimal solution found by solving the integer program using the CPLEX software. Note that solving the integer program is not feasible in most cases; for some of the problem instances we tested, the algorithm took days to finish, and in several cases, it did not terminate at all. Max-Cut runs in a few seconds, and Random is even faster since the algorithm runs in linear time.

For Random, as with all the randomized algorithms that we test, we ran the algorithm 100 times and took the best obtained solution over these runs.

In Figure 4, we plot the percentage of the regions *not covered* by the solution found using these algorithms. As we can see from Figure 4, the solution found by Max-Cut is consistently almost as good as the optimal solution in all but one case (for dataset DS1), for which we could not find the optimal solutions for all graphs.

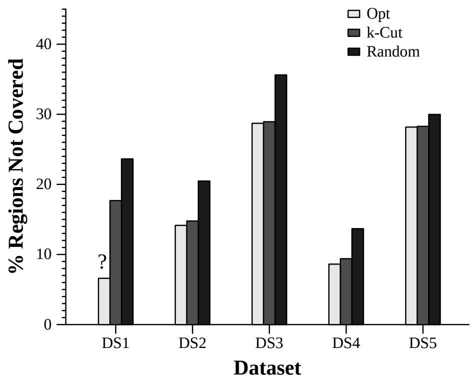

Figure 4: Comparing Max-Cut, Random, and Opt for different datasets. The “?” for DS1 for Opt indicates that the optimal solution could not be found; we instead plot a weak lower bound.

	$\alpha = 1$	$\alpha = 2 \dots 5$
G_1	2	2.5
G_2	3	3.5
G_3	2	2.25
G_4	3	3.5

Table 1: Lifetime increase of sensor network ($\frac{k}{\alpha}$)

6.2 Effect of α

We study the effect of allowing for larger values of α. To try and understand the benefits of allowing larger values of α, we ran the following experiment. We compute (using an IP formulation) the largest possible value of k while asking for full coverage (in other words, we would like each region covered in every time slot) after fixing $\alpha = 1$. In other words, we require a partitioning of the sensors into k sets such that each partition covers all the sensors.

We next compute the maximum value of k for each choice of $\alpha = 2 \dots 5$ such that we can create k subsets $S_1, \dots, S_k$ with each sensor belonging to at most α sets. The maximum value of $\frac{k}{\alpha}$ gives us the maximum lifetime that is possible. We only show a sample of the results in which we got some improvment. In geometric and colored graphs in many cases, there was surprisingly no improvement. For random graphs there are many cases with an improvement. The table is shown above.

6.3 Sensor with Capacity Constraints

With our final set of experiments, we compare the performance of our randomized algorithm for the capacity-constrained version of this problem. Recall that our algorithm finds the optimal solution in the case when capacities are equal to 1; when capacities are greater than or equal to

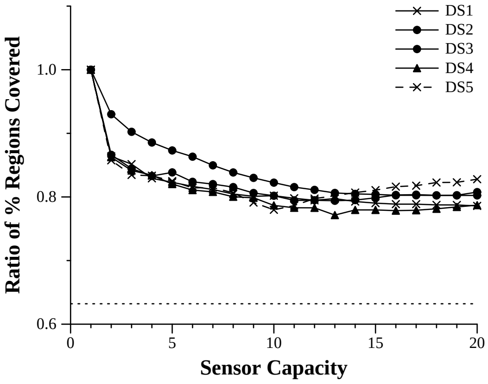

Figure 5: Comparing the solution found by our algorithm for the capacity-constrained case with an optimistic upper bound. Dashed line denotes the approximation ratio guaranteed by our algorithm.

3, the problem is NP-hard, and our algorithm is a $(1 - \frac{1}{e})$ approximation to the optimal solution. We compare the performance of our algorithm against an optimistic bound on the solution – number of edges in H_c^* (see Section 5.2). We plan on implementing an exhaustive algorithm that finds the optimal solution, and compare against it for problems with small sizes.

Figure 5 shows the results of our experiments as we increase the sensor capacities for the five datasets described above. As we can see, the solution achieved by our algorithm is within 80% of the upper bound, much better than the worst case approximation bound. Note that the upper bound that we compute is highly optimistic and is unlikely to be achieved by the optimal solution either.

7 Conclusions

Most of the prior work on energy minimization for sensor networks is based on algorithms based on random partitioning of sensors into sets. For the *first time*, we have shown a better scheme which actually takes the structure of the graph into account by converting the formulation to a hyper-graph formulation, then reducing that to a graph formulation and applying a Max k-Cut algorithm on that. And as the experiments suggest this approach gives almost optimal solutions in practice. We also propose a direct SDP based approach, that currently is slow (albeit polynomial) and we believe would be the approach to use for developing worst case approximation bounds.

General capacitated versions of this problem are introduced and studied for the first time as well. The natural approach for this problem would be to first partition the sensors into sets, and then figure out the assignment of which regions each sensor should monitor. The use of network flows as a pre-processing step is crucial in making this algorithm work. In addition, it is slightly surprising that the problem can be solved in polynomial time with unit capacities but with capacity three it is already NP-complete.

Acknowledgements: We thank David Kempe for useful discussions and Matt McCutchen for useful comments on the writeup.

References

[1] Z. Abrams, A. Goel, and S. Plotkin. Set k-cover algorithms for energy efficient monitoring in wireless sensor networks. In *IPSN '04: Proceedings of the third international symposium on Information processing in sensor networks*, pages 424–432, 2004.

[2] N. Alon. A simple algorithm for edge-coloring bipartite multigraphs. *Inf. Process. Lett.*, 85(6):301–302, 2003.

[3] L. Benini, G. Castelli, A. Macii, E. Macii, M. Poncino, and R. Scarsi. A discrete-time battery model for high-level power estimation. In *DATE '00: Proceedings of the conference on Design, automation and test in Europe*, pages 35–41, New York, NY, USA, 2000. ACM Press.

[4] M. Cardei, M.T. Thai, Yingshu Li, and Weili Wu. Energy-efficient target coverage in wireless sensor networks. In *IEEE Infocom*, 2005.

[5] M. Cardei and Jie Wu. Energy-efficient coverage problems in wireless ad-hoc sensor networks. *Computer Communications*, pages 413–420, 2006.

[6] U. Feige, M. Halldorsson, G. Kortsarz, and A. Srinivasan. Approximating the domatic number. *SIAM J. on Comput.*, 32:172–195, 2002.

[7] A. M. Frieze and M. Jerrum. Improved approximation algorithms for max k-cut and max bisection. In

Proceedings of the 4th International IPCO Conference, pages 1–13. Springer-Verlag, 1995.

[8] M. X. Goemans and D. P. Williamson. .879-approximation algorithms for max cut and max 2sat. In *STOC '94: Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 422–431, New York, NY, USA, 1994. ACM Press.

[9] J. Hastad. Some optimal inapproximability results. *JACM*, 48:798–859, 2001.

[10] H. Liu, X. Jia, P. Wan amd C. Yi, S. Makki, and N. Pissinou. Maximizing lifetime of sensor surveillance systems. *IEEE/ACM Trans. on Networking*, 15:172–195, 2007.

[11] S. Slijepcevic and M. Potkonjak. Power efficient organization of wireless sensor networks. In *IEEE International Conference on Communications (ICC'01)*, 2001.

[12] M. Yannakakis. Node-and edge-deletion np-complete problems. In *STOC '78: Proceedings of the tenth annual ACM symposium on Theory of computing*, pages 253–264, New York, NY, USA, 1978. ACM Press.

A Appendix: SDP Background

Our approach will be to formulate this as a Semi definite programming (SDP) problem. Consider the following SDP for constant d (and arbitrary k). This formulation is inspired by the Max k-Cut work by Frieze and Jerrum [7] (and the description in this section is taken from their paper). However in our direct SDP formulation (Subsection A.1), as we will see shortly, the constraints are significantly more complex.

Let y_j be one of k vectors $a_1, \ldots, a_k$ defined as follows. Consider an equilateral simplex Σ_k in R^{k-1} with vertices $b_1, \ldots b_k$. Let $c = \frac{\sum_{i=1}^{k} b_i}{k}$ be the centroid of Σ_k and let $a_i = b_i - c$. Assume that Σ_k is scaled so that $|a_i| = 1$ for all i.

Frieze and Jerrum show that $a_i \cdot a_j = \frac{-1}{k-1}$, when $i \neq j$. If $i = j$, clearly $a_i \cdot a_j = 1$. (We would like $y_j = a_p$ if and only if $v_j \in S_p$.)

In this framework, let Y_{ij} represent the dot product of the two vectors y_i and y_j. The dot product of the two vectors is 1 if the vertices i and j belong to the same group. The dot product is $\frac{-1}{k-1}$ if they belong to different groups.

The Max k-Cut problem can be formulated as follows:

$$\max \quad \frac{k-1}{k} \sum_{i<j} w_{i,j}(1 - y_i \cdot y_j)$$

$$\text{such that} \quad y_j \in \{a_1, \ldots, a_k\}$$

Note that

$$1 - y_i \cdot y_j = \begin{cases} 0 & \text{if } y_i = y_j \\ \frac{k}{k-1} & \text{if } y_i \neq y_j \end{cases}$$

In this formulation, the graph has weights on the edges specified by $w_{i,j}$. Since the graph we compute G' is a multi-graph, we define the weight of an edge to be the number of copies of the edge in the multi-graph. For example in Fig. 3 our edges will have weight chosen from $\{0, 1, 2\}$.

To obtain the SDP relaxation, now replace y_i by v_i where v_i is any n-dimensional unit vector. We add the constraint $v_i \cdot v_j \geq -\frac{1}{k-1}$. Thus we obtain an SDP (semidefinite program) of the following form:

$$\max \quad \frac{k-1}{k} \sum_{i<j} w_{i,j}(1 - v_i \cdot v_j)$$

$$\text{such that } v_j \in S_n$$

$$v_i \cdot v_j \geq -\frac{1}{k-1} \forall i \neq j$$

The (fractional) solution obtained from solving this SDP is now rounded by a simple randomized rounding algorithm to obtain a solution for the Max k Cut problem. Rather than using exactly the same rounding as given by Frieze and Jerrum [7] we developed another procedure that had excellent performance.

After finding the solution to the convex program, we perform the following procedure to find the partitioning into k sets. First we compute the $\mathcal{V} = v_1, \ldots v_n \in S_n$ from the decomposition of matrix Y. We choose k vectors at random from $\mathcal{V}$, call them $z_1, \ldots, z_k$. Each of the vectors in Z represents one of the k sets. Vector v_i (which represents sensor i) will be assigned to the set j if and only if z_j is the closest vector to v_i. In other words v_i is assigned to j iff $|v_i - z_j| \leq |v_i - z_{i'}|, \forall i' \in \{1 \ldots k\}$.

A.1 Direct SDP Formulation

In this section we take a different route and explore a *direct* SDP formulation for the set k-cover problem without first reducing to max k-cut. Consider a hyper edge $e = \{v^e_{i_1}, v^e_{i_2}, \ldots, v^e_{i_{d_e}}\}$. For each hyper-edge e, we define a new variable α_e. We can formulate the problem as follows.

$$\max \sum_{e \in E} [(\frac{k-1}{k}((d_e - 1) - \alpha_e)) + 1]$$

such that for each hyper edge e we have the following set of constraints.

Consider all possible Hamilton paths P_e among the set of vertices in the hyper edge e. We will generate a constraint for each possible path. This formulation has an exponential number of constraints, but this is not a serious problem. In practice, each region is only covered by a small number of sensors (for constant d, it is polynomial in any case).

For each hyper edge e, we define a new variable α_e and the following set of constraints (one constraint for each path P_e).

$$\alpha_e \geq \sum_{(i_p, i_q) \in P_e} Y_{i_p i_q} \text{such that } y_j \in \{a_1, \ldots, a_k\}$$

If a hyper edge e has size d_e and intersects p_e groups, then we show that the contribution to the objective function is exactly p_e. This can be seen as follows: Consider the path $P' \in P_e$ with the following structure – the path visits all the nodes in a group before visiting the nodes in another group. Note that this path has exactly $p_e - 1$ edges (each dot product contributes $-\frac{1}{k-1}$ for these edges) that go across two groups, and $d_e - p_e$ local edges (each such dot product clearly contributes 1). For this path the rhs of the constraint is exactly $-\frac{p_e - 1}{k-1} + (d_e - p_e)$ (and this is the maximum value of the rhs). Note that for every other path, if there fewer local edges, the sum is only smaller. Putting this value for α_e (the smallest valid choice) gives the correct objective function value of p_e. $(\frac{k-1}{k}((d_e - 1) + \frac{p_e - 1}{k-1} - d_e + p_e)) + 1 = p_e$.

We use the same rounding approach as described in the previous subsection. We do not report on the computational results as the solutions obtained in practice were slightly worse than the ones obtained by reducing to max k-cut, and the algorithm was much slower.

B Datasets

We ran experiments on a wide variety of synthetic datasets, and report the results for a representative sample of them.

- **DS1, DS3** - Random Uniform Datasets: A set of n sensors and m locations were created (for varying values of n and m). For each location v, a degree d_v is chosen uniformly in $[d_{\min}, d_{\max}]$. Each of the d_v sensors covering v is then randomly selected.

  We show experimental results for two datasets of this type. **DS1** was generated with $n = 20$, $m = 50$, and $[d_{min}, d_{max}] = [8, 15]$. The value of k was set to 11 for the experiments with these graphs. **DS2** was generated with $n = 20, m = 50$ and $[d_{min}, d_{max}] = [3, 5]$, and the experiments were run with $k = 5$.

- **DS2, DS4** - Geometric (2D and 3D) Datasets: A set of n sensors and m locations are created as random points in the unit square (**DS2**) or unit cube (**DS4**) . A sensor covers all locations within distance r. Locations with degree less than $d_{\min}$ $(= 4)$ are deleted. Once again we use $n = 20$ sensors and $m = 50$ locations, and the experiments were run with $k = 5$. The value of r were set to 0.4 and 0.6 for **DS2** and **DS4** respectively.

- **DS5** - Colored Datasets: This dataset is designed to capture multi-modal, heterogenous sensor deployments. Given a connectivity graph and c colors, each location is given a color (*modality*) uniformly selected in range $[1, c]$. Each sensor can monitor a subset of colors. This color set is chosen uniformly from all non-empty subsets of

$1, 2, \cdots, c$. The connectivity graph is then filtered so that sensors are only connected to locations with colors in their color set. Locations with degree less than $d_{\min}$ are deleted.

We conducted experiments on colored version of all of **DS1, DS2, DS3**, and **DS4**. We report the results for the colored variant on the **DS2** (geometric 2D dataset) with c set to 3.