, , 1–15 ()

Efficient Point Coverage in Wireless Sensor Networks*

JIE WANG wang@cs.uml.edu
Department of Computer Science, University of Massachusetts, Lowell, MA 01854, USA.

NING ZHONG zning@cs.uml.edu
Department of Computer Science, University of Massachusetts, Lowell, MA 01854, USA.

Abstract. We study minimum-cost sensor placement on a bounded 3D sensing field, R, which comprises a number of discrete points that may or may not be grid points. Suppose we have ℓ types of sensors available with different sensing ranges and different costs. We want to find, given an integer $\sigma \geq 1$, a selection of sensors and a subset of points to place these sensors such that every point in R is covered by at least σ sensors and the total cost of the sensors is minimum. This problem is known to be NP-hard. Let k_i denote the maximum number of points that can be covered by a sensor of the ith type. We present in this paper a polynomial-time approximation algorithm for this problem with a proven approximation ratio $\gamma = \sum_{i=1}^{\ell} k_i - \sigma + 1$. In applications where the distance of any two points has a fixed positive lower bound, each k_i is a constant, and so we have a polynomial-time approximation algorithms with a constant guarantee. While γ may be large, we note that it is only a worst-case upper bound. In practice the actual approximation ratio is small, even on randomly generated points that do not have a fixed positive minimum distance between them. We provide a number of numerical results for comparing approximation solutions and optimal solutions, and show that the actual approximation ratios in these examples are all less than 3, even though γ is substantially larger.

Keywords: sensor placement, approximation algorithms, linear programming

1. Introduction

Coverage is a fundamental issue in wireless sensor networks. It deals with problems of deploying wireless sensors and constructing sensor networks to cover regions of interest to detect, track, and report threats of mechanical, chemical, and biological agents, or monitor environmental and structural changes. The regions of interest could be areas of natural land forms and physical contours, and critical man-made structures such as airports, bridges, city blocks, high-rise buildings, oil tanker engine rooms, and oil platforms, to name just a few. Actual sensor deployments have so far used up to 10,000 sensor nodes in a single application [6].

Research in sensor network coverage has mainly followed four directions:

1. Design communication protocols (e.g., [12, 13, 5]).

2. Investigate coverage measures and develop analytical expressions of coverage properties based on these measures (e.g., [7, 12, 13, 8, 9, 11]).

* This research was supported in part by NSF under grant CCF-04080261 and by NSF of China under grant 60273062.

3. Achieve maximum lifetime of sensor networks (e.g., [3, 2]).
4. Find a minimum-cost sensor placement on sensing fields for surveillance (e.g., [4, 10, 14]).

We study in this paper the problem of minimum-cost sensor placement on collections of points that may or may not be grid points. Let R be a bounded 3D (or 2D) region comprising n discrete points. These points are to be covered by sensors and sensors may be placed on any of these points. Moreover, we assume that each point can only be placed at most one sensor. For convenience, we call a point a *site* if it is selected to place a sensor. Assume that we have several types of sensors available with different costs and different sensing ranges. We want to find a selection of sensors and sites to place these sensors such that each point in the sensing field is covered by at least σ sensors and the total cost of sensors is minimum, where $\sigma \geq 1$ is an integer. This problem is known to be NP-hard [4].

Chakrabarty *et al* [4] were the first to study this problem on grid sensing fields. By a grid sensing field it means a collection of grid points in a given grid. They formulated an ILP (Integer Linear Programming) model with $\Theta(\ell n^2)$ variables and $\Theta(\ell n^2)$ constraints for solving the problem, where n is the number of grid points in the sensing field and ℓ the number of different types of sensors. Their ILP model is a linearization of an easy model that contains quadratic constraints. Taking a different approach, we obtain a new ILP model with only ℓn variables and $2n$ constraints. (Sahni and Xu also presented the same formulation in a recent paper [14], but they did not provide efficient solutions.) Morever, this new model can be applied to any set of discrete points. Namely, points do not have to be grid points and the point sets may have irregular boundaries. The boundary of a given point set is the line (or lines) connecting neighboring boundary points in the set. This paper is based on the new ILP model.

The ILP problem is intractable when the given sensing field is large. A simple method of handling a large grid is, as suggested in [4], dividing it into a number of manageable small sub-grids and combine the optimal solution to each small sub-grid as the approximation solution to the original grid. This is a simple divide-and-conquer (d-n-c) scheme. Given a set of available sensors, let $N_{p,m}$ denote the optimal number of sensors required for covering a cubic (or square) grid sensing field, where m is the dimension of the sensing field and p the number of grid points in each dimension. Let the corresponding cost of sensor deployment be $C_{p,m}$. Chakrabarty *et al* [4] showed that

$$\begin{aligned} N_{2^k p,m} &\leq 2^{km} N_{p,m}, \\ C_{2^k p,m} &\leq 2^{km} C_{p,m}. \end{aligned}$$

The d-n-c scheme can be applied to cuboid and rectangular grid sensing fields. While one can apply d-n-c to a point set that contains irregular boundaries or non-grid points, dividing it properly to yield a good approximation solution to the original set could be difficult, especially when there is no clear indication of clusters.

We present in this paper a new approximation algorithm to the ILP problem with a proven approximation ratio. Our algorithm using the standard technique of

solving ILP models: We first loosen the integer requirement to allow variables in the ILP model to take real values, turning it into a pure LP (Linear Programming) model. After solving the LP problem (which has fast algorithms), we convert the LP solution to an ILP solution in $O(n \log n)$ time.

The approximation in our algorithm depends on LP solutions. We observe that, through a large number of experiments, the approximation ratio rarely reaches the (worst-case) theoretical upper bound. Indeed, we have not been able to produce a single case where the actual approximation ratio is even near this worst-case upper bound. There is a good explanation for this; the reader is referred to the discussion at the end of Section 3.1. In practice the actual approximation ratio tends to be small, even though the theoretical upper bound is much larger.

In applications where any two points in a sensing field has a fixed positive lower bound, the approximation ratio is a constant. This is a practical assumption in certain applications. For examples, in grid sensing fields or in sensing fields of highly structural objects such as bridges and oil platforms, only certain locations with a fixed positive minimum distance may be points of interest to monitor or to place sensors. In applications where several points are physically close, we may simply consider a cluster of close points within a fixed distance a single point, where the center of the cluster is the location of this point. Sensor coverage obtained according to this simplification covers the centers of clusters, but may not cover all the points in a cluster. To compensate this loss of coverage we could reduce the effective radius of sensors in our algorithms so that the sensor placement we obtain that covers centers of clusters will have an actual effect that also covers the points in each cluster.

We show that, through experiments on all square grid sensing fields where optimal solutions can be obtained on a PC using the latest *lpsolve* package [1], the approximation solution in each case is less than 3 times of the optimal solution, although the theoretical upper bound of the approximation ratio is substantially larger. Even if there is no fixed minimum positive distance between points, we show that, through experiments on sets of points that are independently and uniformly generated at random, including sets that are dense, the actual approximation ratio in each case is still less than 3, although the theoretical upper bound ratios are much higher.

In addition, we demonstrate a simple point set containing non-grid points, on which our approximation algorithm is better than d-n-c. We note that the d-n-c scheme is sometimes better than our approximation algorithm, especially on cubic and square grid sensing fields. Thus, network designers may want to apply both methods on a given point set and select the best sensor deployment plan.

The rest of the paper is organized as follows. In Section 2 we describe the ILP model used in this paper. In Section 3 we present our approximation algorithm, prove its correctness, time complexity, and approximation ratio. We then demonstrate a simple point set with non-grid points on which our approximation algorithm is better than d-n-c. In Section 4 we present a number of numerical results, comparing runtime of the ILP model used in this paper and the ILP model used in [4], and comparing approximation solutions and the optimal solutions.

2. The minimization problem and an ILP model

Let R be a bounded 3D (or 2D) region consisting of n discrete points, which may or may not be grid points. For convenience, we simply view R as a set of points. Let $t_1, \ldots, t_\ell$ be ℓ types of sensors with ranges (i.e., radius) $r_1, \ldots, r_\ell$, where $r_1 < \cdots < r_\ell$. We assume that a sensor can only be placed on a point (also called a site) and each site can only be occupied by at most one sensor. The sensor placement problem is to find, given an integer $\sigma \geq 1$, a selection of sensors and a subset of sites to place these sensors such that every point in R is covered by at least σ sensors and the total cost of the sensors is minimum.

We label the n points as $1, 2, \ldots, n$, and denote by $d(i, j)$ the Euclidean distance between point i and point j. Let

$$\begin{aligned} E_v &= \{(i,j) \mid 0 \leq d(i,j) \leq r_v\},\ v = 1, \ldots, \ell. \\ E_v[i] &= \{j \mid (i,j) \in E_v\},\ i = 1, \ldots, n. \end{aligned}$$

Let C_v denote the cost of a type-t_v sensor. Let x_i^v, $v = 1, \ldots, \ell$, be 0-1 integer variables such that

$$x_i^v = \begin{cases} 1, & \text{if a type-}t_v\text{ sensor is placed at grid point } i \\ 0, & \text{otherwise} \end{cases}$$

The sensor placement problem can be formulated as the following ILP problem:

$$\text{Minimize} \quad \sum_{i=1}^{n} \sum_{v=1}^{\ell} C_v x_i^v \tag{1}$$

$$\text{Subject to, for all } i\text{:} \quad \sum_{v=1}^{\ell} \sum_{j \in E_v[i]} x_j^v \geq \sigma \tag{2}$$

$$\sum_{v=1}^{\ell} x_i^v \leq 1 \tag{3}$$

where Constraint 2 ensures that each point is covered by at least σ sensors, and Constraint 3 ensures that each point can only be used as a site to place at most one sensor.

3. Approximation algorithm

We first loosen the integer constraints in the ILP model by allowing integer variables x_i^v to take real values in $[0, 1]$. This gives rise to an LP model, which can be solved using fast algorithms. We then convert the optimal LP solution to an integer solution for the ILP problem. For simplicity we will first present our approximation algorithm on two types of sensor (i.e., $\ell = 2$). We then generalize the algorithm to the general case.

3.1. Coverage using two types of sensors

Denote by A and B the two types of sensors. Let $\{x_i^{A,*}, x_i^{B,*} \in [0,1] \mid i = 1, \ldots n\}$ be an optimal solution to the LP model with

$$OPT_{LP} = \sum_{i=1}^{n} C_A x_i^{A,*} + C_B x_i^{B,*}.$$

To convert this solution to an integer solution, we construct a graph $G = (V, E)$ with $V = \{1, \ldots, n\}$, where a pair of vertices are connected with an edge if and only if a sensor placed at one end point also covers the other end point. There are two types of connections. We use edge labels to mark these connections. That is, for each $i, j \in V$ with $i \neq j$, if $d(i,j) \leq r_A$, connect i and j and label the edge with A. If $r_A < d(i,j) \leq r_B$, connect i and j and label the edge with B.

Let k_A denote the maximum number of points a type-A sensor can cover on a given sensing field. Similarly, let k_B denote the maximum number of points a type-B sensor can cover. Namely,

$$k_A = \max_{i \in V}\{|E_A[i]|\},\ k_B = \max_{i \in V}\{|E_B[i]|\}.$$

Without loss of generality, we assume that $\sigma \leq |E_A[i]| + |E_B[i]|$ for each point i. (Otherwise, point i cannot be covered by σ sensors.)

The idea of our algorithm is this: Start from point i with the largest degree in R, and select the σ largest values $x_{ij}^{A,*}$ and $x_{ij}^{B,*}$ that can over i. Set these σ variables to 1. Remove point i and any other point from R that is also covered by these σ sensors. Continue this process until all points are removed.

Algorithm $\mathcal{A}$:

1. Let $S = \emptyset$ and L be a sorted list of nodes in V in non-increasing order according to their degrees.

2. Select $i_0 \in L$ such that vertex i_0 has the largest degree. Let

   $$H_A = \{x_j^{A,*} \mid j \in E_A[i_0]\},\ H_B = \{x_j^{B,*} \mid j \in E_B[i_0]\},\ H = H_A \cup H_B.$$

   Sort H according to the values of the variables. Let $h_{j_1}, \ldots, h_{j_\sigma}$ be the σ variables in H with the largest values (not in any particular order), where $h_{j_u} = x_{j_u}^{A,*}$ or $x_{j_u}^{B,*}$, $u = 1, \ldots, \sigma$, such that $j_u \neq j_{u'}$ if $u \neq u'$.

3. Include $h_{j_1}, \ldots, h_{j_\sigma}$ in S. Let W be the intersection of the subsets of points covered by corresponding sensors placed at points j_u for $u = 1, \ldots, \sigma$. Remove W from L. That is, set

   $$S = S \cup \{h_{j_1}, \ldots, h_{j_\sigma}\},\ L = L - W.$$

   (Note that W can be constructed as follows: Start with $W = E_A[j_1]$ if $h_{j_1} \in H_A$ or $W = E_B[j_1]$ otherwise. For $u = 2, \ldots, \sigma$, if $h_{j_u} \in H_A$, set $W = W \cap E_A[j_u]$; if $h_{j_u} \in H_B$, set $W = W \cap E_B[j_u]$.)

4. Repeat Step 2 until $L = \emptyset$.
5. Set the value of each variable in S to 1, and the value of each variable in $V - S$ to 0.

THEOREM 1 *Algorithm $\mathcal{A}$ provides an integer solution to the ILP problem in $O(n \log n)$ time.*

Proof: It is obvious that Algorithm $\mathcal{A}$ provides an integer solution (see Step 5). Note that a previously selected variable at Step 2 can be selected again at a later time. Since the variables selected at Step 2 have different indexes, variables that are given the value of 1 at Step 5 all have different indexes. This implies that each point can only be occupied by at most one sensor. The set W constructed at Step 3 in each iteration of the algorithm is nonempty, for it contains at least one point $i_0 \in L$, and W consists of points that can be covered by exactly σ sensors of the current selection. Removing W from L in each iteration, the algorithm will stop within n iterations. Since additional sensors may be selected at a later step that may cover some of the points included in a previously constructed W, we know that each point will be covered by σ or more sensors.

Graph G can be constructed in $O((k_A + k_B)n)$ time and the descending list L can be constructed in $O((\log(k_A + k_B)n \log n))$ time. At Step 2, i_0 can be found in constant time, H can be constructed in $O(k_A + k_B)$ time, and the σ variables with the largest values in H can be selected in $O(\sigma(k_A \log k_A + k_B \log k_B))$ time. At Step 3, the set W can be constructed in $O(\sigma(k_A + k_B))$ time. Removing W from L can be carried out in $O(\sigma) \leq O(k_A + k_B)$ time using a proper data structure for implementing L. We note that Steps 2 and 3 can only be iterated at most n times, and k_A and k_B are independent of n. Thus, Algorithm $\mathcal{A}$ runs in $O(n \log n)$ time. ■

Let OPT_{ILP} denote the optimal solution of the original ILP problem and Δ the solution of Algorithm $\mathcal{A}$. Let $x_i^{A,\Delta}$ and $x_i^{B,\Delta}$ denote the values assigned to variables x_i^A and x_i^B, respectively, at Step 5 in Algorithm $\mathcal{A}$. Let $\beta = k_A + k_B - \sigma + 1$.

THEOREM 2 $\Delta \leq \beta \cdot OPT_{ILP}$.

Proof: We first note that $OPT_{LP} \leq OPT_{ILP}$. We then show that for any variable h put in S, we must have

$$h \geq 1/\beta. \tag{4}$$

Assume that h is put in S with respect to point i_0 selected at Step 2. We note that $\sigma - 1$ other variables from H_A and H_B are also included in S at the same time as h is included. Let $h_1, \ldots, h_\sigma$ denote these variables in the descending order. From Constraint 2 we have

$$\sum_{j \in E_A[i_0]} x_j^{A,*} + \sum_{j \in E_B[i_0]} x_j^{B,*} \geq \sigma.$$

We note that there are at most k_A variables $x^{A,*}$ and k_B variables $x^{B,*}$ in this inequality. Thus, $h_1 \geq \sigma/(k_A + k_B)$. Since

$$\sum_{j \in E_A[i_0]} x_j^{A,*} + \sum_{j \in E_B[i_0]} x_j^{B,*} - h_1 \geq \sigma - 1,$$

we have

$$h_2 \geq \frac{\sigma - 1}{k_A + k_B - 1}.$$

A straightforward induction leads us to the following inequality:

$$h_i \geq \frac{\sigma - i + 1}{k_A + k_B - i + 1},\ i = 1, \ldots, \sigma.$$

This implies that Inequality 4 is true. Hence,

$$\begin{aligned} OPT_{LP} &= \sum_i C_A x_i^{A,*} + C_B x_i^{B,*} \\ &\geq \sum_{x_i^{A,*} \in S} C_A x_i^{A,*} + \sum_{x_i^{B,*} \in S} C_B x_i^{B,*} \\ &\geq \frac{1}{\beta}\left(\sum_i C_A x_i^{A,\Delta} + \sum_i C_B x_i^{B,\Delta}\right) \\ &= \Delta/\beta. \end{aligned}$$

It follows from $OPT_{ILP} \geq OPT_{LP}$ that $\Delta \leq \beta \cdot OPT_{ILP}$. ■

The approximation ratio β we show here is a worst-case upper bound. The actual approximation ratio depends on the optimal solution $x_j^{A,*}$ and $x_j^{B,*}$. The theoretical upper bound β may be reached only if in Constraint 2 in the LP model, for each point i selected in Algorithm $\mathcal{A}$, and for each $j \in E_A[i]$ and $j \in E_B[i]$, we have $x_j^{A,*} = x_j^{B,*} = \sigma/(|E_A[i]| + |E_B[i]|)$. This situation is rare. In practice, the actual approximation ratio tends to be small, although the theoretical upper bound is significantly larger. This is because the LP solutions often provide sufficiently good values of $x_j^{A,*}$ and $x_j^{B,*}$ to select from for the approximation algorithm; namely, these values are seldom all equal; some of them are either equal to 1 or close to 1, while some others are equal to 0 or close to 0. We will present numerical experiments in Section 4. Indeed, we have not seen a single case where the actual approximation ratio is even near the worst-case theoretical upper bound.

3.2. Coverage using more than two types of sensors

We generate Algorithm $\mathcal{A}$ to allow sensors of more than two types; i.e., $\ell > 2$. As in Section 3.1 we construct a graph $G = (V, E)$ with $V = \{1, \ldots, n\}$ such that for each pair of vertices i and j and for each $v = 1, \ldots, \ell$, if $r_{v-1} < d(i, j) \leq r_v$, connect i and j and label the edge with a label v, where $r_0 = 0$.

Let k_v denote the maximum number of points the type-t_v sensor can cover on the given point set. Assume that $\sigma \leq \sum_{v=1}^{\ell} |E_v[i]|$ for each point i. Similar to Algorithm $\mathcal{A}$ we have the following approximation algorithm.

Algorithm $\mathcal{B}$:

1. Let $S = \emptyset$ and L be a sorted list of nodes in V in non-increasing order according to their degrees.

2. Select $i_0 \in L$ such that vertex i_0 has the largest degree. For $v = 1, \ldots, \ell$, let

$$H_v = \{x_j^{v,*} \mid j \in E_v[i_0]\},\ H = \bigcup_{v=1}^{\ell} H_v.$$

   Sort H according to the values of the variables. Let $h_{j_1}, \ldots, h_{j_\sigma}$ be the σ variables in H with the largest values (not in any particular order), with σ_v variables from H_v, where for each u with $1 \leq u \leq \sigma$: $h_{j_u} = x_{j_u}^{v,*}$ for some v with $1 \leq v \leq \sigma$, such that $j_u \neq j_{u'}$ if $u \neq u'$.

3. Let $U = \{v \in [1, \ell] \mid (\exists u \in [1, \sigma])[h_{j_u} \in H_v]\}$. Set

$$S = S \cup \{h_{j_1}, \ldots, h_{j_\sigma}\},\ L = V - \bigcap_{u=1}^{\sigma} \bigcap_{v \in U} \left[\bigcap_{h_{j_u} \in H_v} E_v[j_u] \right].$$

4. Repeat Step 2 until $L = \emptyset$.

5. Set the value of each variable in S to 1, and the value of each variable not in S to 0.

Let $x_i^{v,*} \in [0, 1]$ be an optimal LP solution. Let $OPT_{LP} = \sum_{i=1}^{n} \sum_{v=1}^{\ell} C_v x_i^{v,*}$. Let Δ_σ denote the solution of Algorithm $\mathcal{B}$. Let $\gamma = \sum_{v=1}^{\ell} k_v - \sigma + 1$. Similar to Theorems 1 and 2 it is straightforward to show the following result, and we omit its proof.

THEOREM 3 *Algorithm $\mathcal{B}$ provides an integer solution to the ILP problem in $O(n \log n)$ time, and $\Delta_\sigma \leq \gamma \cdot OPT_{ILP}$.*

3.3. Approximation and d-n-c

Finding a good d-n-c plot is not easy, even on simple sets of points. To this end, we present a simple point set that contains non-grid points on which our approximation algorithm is better than d-n-c on regular plots.

The notion of grid points depends on the ranges of given sensors. Let $r_1 < r_2 < \ldots < r_\ell$ be the ranges of ℓ types of sensors. Let $r = gcd(r_1, r_2, \ldots, r_\ell)$. Let R be a given point set. We form a grid so that each edge on the smallest cube (or square) has length r, where the corner points are grid points. We place this grid to R in a

natural way to make R fall on grid points as much as possible. The points in W that do not fall on grid points are the non-grid points.

Our example consists of a set of 48 grid and non-grid points (see Figure 1(a)) and two types of sensors A and B, where $r_A = 100$ and $r_B = 200$ under the same distance unit; $C_A = 100$ and $C_B = 200$ under the same currency unit. For convenience, we omit the distance unit (which may be feet, meters, etc.) and the currency unit (which may be American Dollars, Euros, etc.).

Note that there always exist d-n-c plots that provide the optimal solution to the original set. Finding such a plot, however, could be as hard as finding the optimal solution itself. Points in our example are evenly distributed and so there is no clear advantage of one d-n-c plot over the other. Thus, what we will demonstrate is that our approximation algorithm is better than the regular d-n-c plots.

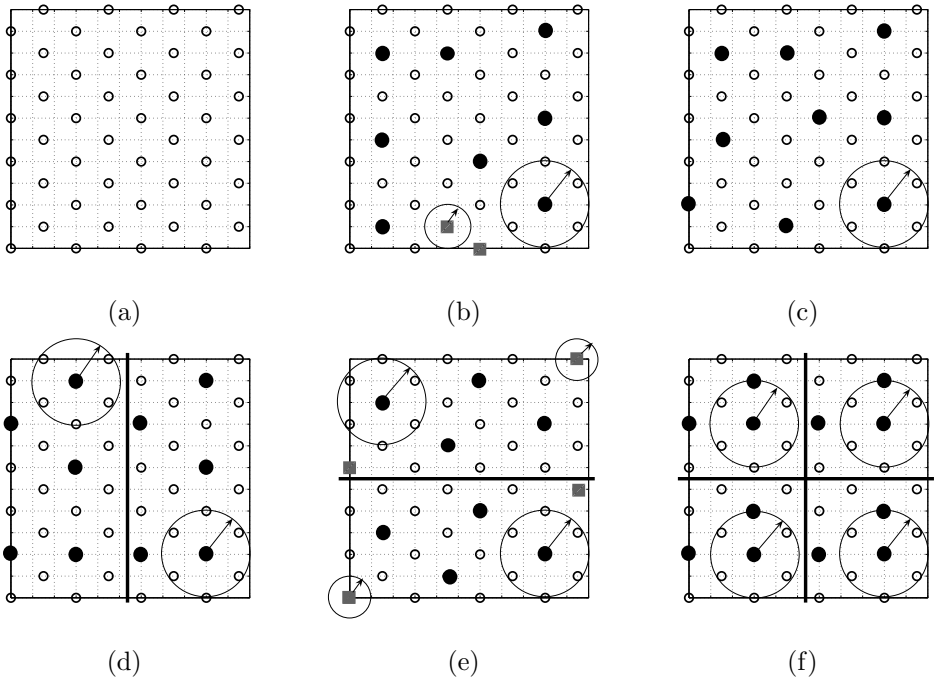


Figure 1. (a) A sensing field with 48 points shown as small circles. There are 24 grid points and 24 non-grid points. The length of each edge in the smallest square is 100. The non-grid points are in the middle of the corresponding columns. (b) Optimal sensor deployment plan produced by the ILP model, where the grey square denotes a type-A sensor and the black solid circle denotes a type-B sensor. (c) Sensor deployment plan produced by Algorithm $\mathcal{A}$. (d)–(f): Three d-n-c plots and the corresponding sensor deployment plans, where the grey square denotes a type-A sensor and the black solid circle denotes a type-B sensor.

Figure 1(a) is a 1:100 scaled sensing field consisting of 48 points, shown as small circles, where 24 points are grid points and the rest non-grid points. Figure 1(b)

illustrates the optimal sensor deployment plan produced by the ILP model. Figure 1(c) illustrates the sensor deployment plan produced by Algorithm $\mathcal{A}$, which happens to be optimal as well. Figure 1(d) to (f) illustrate three normal d-n-c plots and the corresponding sensor deployment plans. Table 1 provides a comparison of sensor cost of each sensor deployment plan.

Table 1. Sensor cost comparison.

methods	*number of sensors*			*cost*
	type-A	type-B	total	
ILP	2	8	10	1800
Algorithm $\mathcal{A}$	0	9	9	1800
d-n-c plot a	0	10	10	2000
d-n-c plot b	4	8	12	2000
d-n-c plot c	0	12	12	2400

4. Implementations and experiments

We compare the runtime of the ILP model used in this paper (referred to as the new ILP) and the ILP model used in [4] (referred to as the old ILP), and compare the optimal solutions produced by the ILP model and the approximation solutions produced by our approximation algorithm. We implemented the two ILP models using *lpsolve* 5.5 [1]. We wrote a C program to generate input files and a C program to implement Algorithm $\mathcal{A}$. We performed all of our experiments on a Dell Optiplex GX 260 PC, equipped with a 2.26 GHz Pentium-4 CPU and 1 GB RAM.

4.1. Grid sensing fields

We first provide numerical results on square grid sensing fields for the purpose to compare with the numerical experiments presented in [4]. For each square grid, its point set consists of exactly all the grid points.

In Table 2, the first column is the grid size. The second column is σ. The third and the fourth columns are the runtime in seconds for solving the new and the old ILP models, respectively. The fifth column is the optimal solution. We use the following two types of sensors A and B as in [4]: $r_A = 100$m, $r_B = 200$m, $C_A = 150$ USD, and $C_B = 200$ USD. We omit the measurement unit and the currency. When a sensing field is bigger than 9×9 (81 grid points in total), executions of the old ILP model are suspended with an "out of memory" error message, partly due to the large input size of the old IPL model. For a 10×10 grid, for instance, the input size of the old model is 2.69 MB, while the input size of the new model is only 16.7 KB. For a 20×20 (400 grid points in total), the runtime of the new ILP model for $\sigma = 1$ surges to 14 hours.

Since LP problems have polynomial-time algorithms (available also in *lpsolve* 5.5), our approximation algorithms can produce results efficiently on very large grids. For

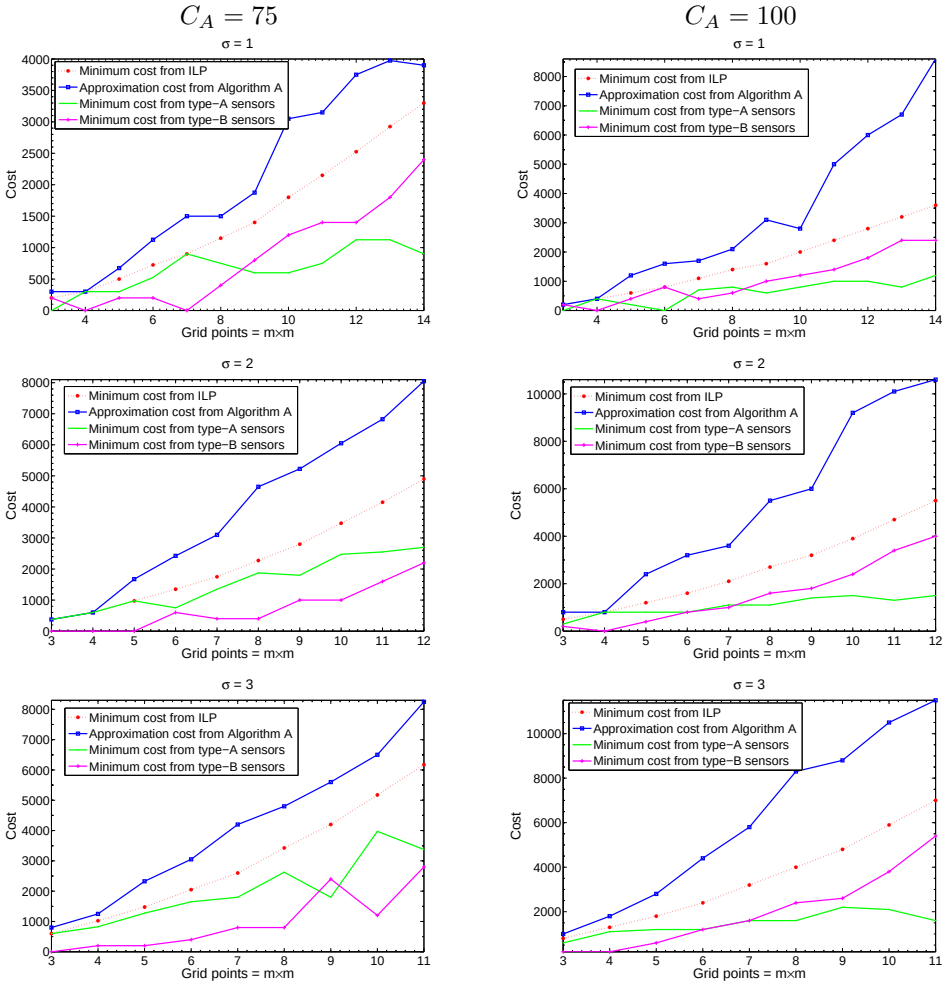


Figure 2. Sensor cost comparison of IPL and Algorithm-$\mathcal{A}$ solutions. Also compared are the number of type-A sensors and the number of type-B sensors occurred in the optimal solutions in the form of their costs.

Table 2. Runtime (in seconds) comparison of the new and the old ILP models, where "o.m." stands for "out of memory."

grid size	σ	*runtime of new ILP*	*runtime of old ILP*	*optimal solution*		
				# of A	# of B	cost
9×9	1	1.593	151.297	0	9	1800
9×9	2	3.828	279.875	3	15	3450
9×9	3	13.812	273.906	1	25	5150
10×10	1	2.484	o.m.	1	10	2150
10×10	2	15.516	o.m.	1	20	4150
10×10	3	53.969	o.m.	1	30	6150
11×11	1	1.547	o.m.	0	12	2400
11×11	2	22.047	o.m.	0	24	4800
11×11	3	274.73	o.m.	0	36	7200
12×12	1	6.735	o.m.	1	14	2950
12×12	2	323.016	o.m.	1	28	5750
12×12	3	1704.187	o.m.	1	42	8550
13×13	1	27.315	o.m.	0	17	3400
13×13	2	1742.235	o.m.	4	30	6600
13×13	3	80670.469	o.m.	2	48	9900

example, Algorithm $\mathcal{A}$ produces a solution in less than 3 seconds on the 20×20 grid sensing field with $\sigma = 3$, while the execution of the new IPL model is still not reaching an optimal solution after 168 hours of continuous running.

In the following experiments, we fix $r_A = 100$, $r_B = 200$, and $C_B = 200$. We let C_A take different values of 50, 75, 100, and 125. Shown in Figure 2 are some of the numerical results for $C_A = 75$ and $C_B = 100$, respectively.

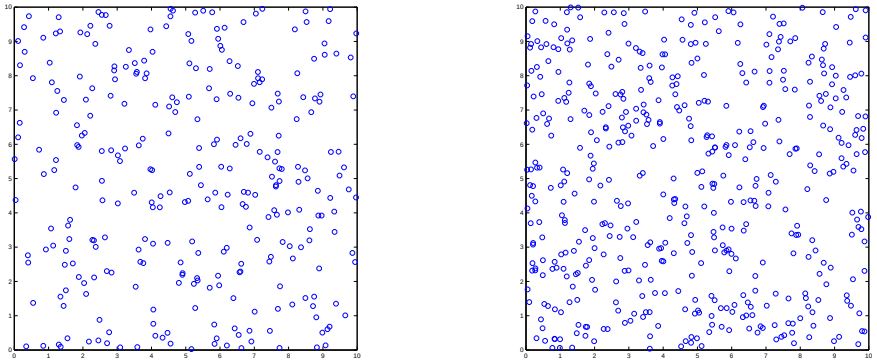
Let $\rho = \frac{r_A}{C_A} \cdot \frac{C_B}{r_B}$. If $\rho > 1$, it means that a type-A sensor can cover more points per dollar than a type-B sensor and so a type-A sensor is more likely to be selected in optimal deployment plans. This is confirmed by the three graphs in the left-hand column in Figure 2, where $\rho = \frac{4}{3}$. To see this we just need to convert the cost curves of type-A sensors and type-B sensors in Figure 2 in terms of the number of sensors. Likewise, if $\rho < 1$, then type-B sensors are more likely to be selected. This is confirmed by Table 2, where $\rho = \frac{2}{3}$. If $\rho = 1$ then both types of sensors would have the same likelihood to be selected. This is confirmed by the three graphs in the right-hand column in Figure 2. We have performed a large number of other experiments (not presented here due to space limitation); all confirm this observation.

4.2. Sensing fields with randomly generated points

In Section 4.1 we have seen that in grid sensing fields, the actual approximation ratio provided by our approximation algorithm is small, even though the theoretical upper bound is significantly larger. In each of our experiments, the actual

approximation ratio is less than 3. In this section we show that even when points are randomly generated that do not have a fixed minimum distance between two points, including very dense sets of points, the actual approximation ratio produced by our algorithm is still less than 3. In particular, we generate points in a fixed area independently and uniformly at random, from a small number of points to a large number of points.

We present here a set of numerical results on randomly generated points in a 10×10 area with $\sigma = 1$ and two types of sensors A and B, where $r_A = 0.5$, $r_B = 1$, $C_A = 100$, and $C_B = 200$. The numbers of points are $50N$, where $N = 1, 2, \ldots, 10$. Figure 3 shows, respectively, 300 points and 500 points in the 10×10 area that are generated independently and uniformly at random. The latter is an example of a dense set of points.



(a) (b)

Figure 3. (a) A sensing field with 300 randomly generated points in a 10×10 area. (b) A sensing field with 500 randomly generated points in a 10×10 area.

Table 3 shows the optimal solution, the approximation solution, k_A, k_B, actual approximation ratio, and the worst-case approximation ratio β of Theorem 2 in each of these 10 cases. In each case the time used to obtain its approximation solution is less than 3 seconds, while the time used to obtain its optimal solution, although still manageable, is much longer. The case of 450 points and the case of 500 points have similar optimal solutions, but the time required to obtain the optimal solution in the case of 500 points is substantially longer than that of 450 points. Generating more than 500 points will not result in much difference in optimal solutions, for the extra points will likely fall in the existing coverage. In all these cases the actual approximation ratio is less than 2.17, while the theoretical approximation ratio β is substantially larger in each case.

Figure 4 compares the minimum costs, actual approximation costs, and the theoretical upper bound of the approximation costs for each of the 10 cases. The (worst-case) theoretical upper bound of the approximation cost is calculated by

Table 3. Comparison of optimal cost, approximation cost, actual approximation ratio, and the theoretical approximation ratio β.

number of points	*Optimal solution*	*Appr. solution*	k_A	k_B	*actual appr. ratio*	*theoretical appr. ratio* β
50	3700	3900	3	4	1.05	7
100	5500	5500	4	7	1	11
150	6000	6700	5	9	1.12	14
200	6600	9700	6	12	1.47	18
250	7100	10300	7	12	1.45	19
300	7000	11000	9	14	1.57	23
350	7300	12300	9	15	1.68	24
400	7200	13900	11	17	1.93	28
450	7300	15600	12	19	2.14	31
500	7500	16300	12	21	2.17	33

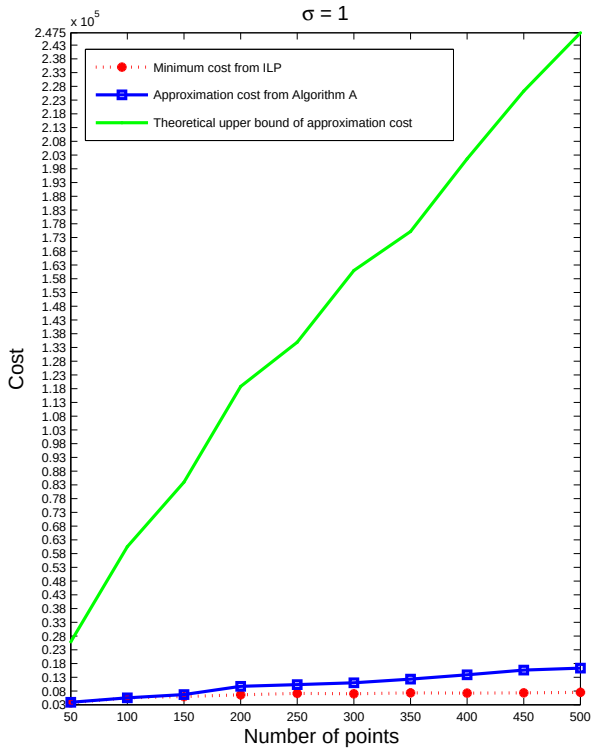


Figure 4. Comparisons of optimal costs, actual approximation costs, and the theoretical upper bound of approximation costs. Points are generated independently and uniformly at random in a 10×10 area.

$\beta \times OPT$, where OPT is the optimal solution. The actual approximation costs are close to the minimum costs, although the theoretical upper bounds are far away.

We also run experiments with increased sensor radii to generate denser networks. For example, let $r_A = 1$ and $r_B = 2$. For the same sets of points, k_A can be as high as 26 and k_B can be as high as 60. The actual approximation ratio in each case is still within the range of 3, while the theoretical upper bound can be as high as 86.

Acknowledgments

We thank Dr. Benyuan Liu for a number of valuable discussions. We thank the anonymous referees for their constructive comments.

References

1. M. Berkelaar, P. Notebaert, K. Eikland. LP Solve: Linear Programming Code, version 5.5. Eindhoven Univ. of Technology, The Netherlands. 2005.
2. M. Cardei, D.-Z. Du. Improving wireless sensor network lifetime through power aware organization, *ACM Wireless Networks*, 11(3):333–340, 2005.
3. M. Cardei, D. MacCallum, X. Cheng, M. Min, X. Jia, D. Li, D.-Z. Du. Wireless sensor networks with energy efficient organization, *Journal of Interconnection Networks*, 3(3–4):213–229, 2002.
4. K. Chakrabarty, S. S. Iyengar, H. Qi, and E. Cho. Grid coverage for surveillance and target location in distributed sensor networks. *IEEE Trans. on Comput.*, 51(12):1448–1453, 2002.
5. T. Clouqueur, V. Phipatanasuphorn, P. Ramanathan, and K. K. Saluja. Sensor deployment strategy for target detection. In *Proc. of 1st ACM International Workshop on Wireless Sensor Networks and Applications*, 2002, pages 42–48.
6. D. Culler. Sensor networks–the next tier of the Internet. Keynote speeach in the *Workshop on Sensor Networks–What Is Real and What Lies Ahead*, Boston University, November 18, 2005.
7. D. Gage. Command control for many-robot systems. In *Proc. of the 9th AUVS Technical Symposium*, 1992. Reprinted in *Unmanned Systems Magazine*, 10(4):28–34, 1992.
8. C. Gui and P. Mohapatra. Power conservation and quality of surveillance in target tracking sensor networks. In *Proc. of the 10th ACM Mobicom*, 2004, pages 129–143.
9. C.F. Hsin and M. Liu. Network coverage using low duty-cycled sensors: random and coordinated sleep algorithms. In *Proc. of the 3rd intl. symposium on Information processing in sensor networks*, 2004, pages 433–442.
10. K. Kar and S. Banerjee. Node placement for connected coverage in sensor networks. In *Proc. WiOpt 2003: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, 2003.
11. B. Liu and D. Towsley. A study on the coverage of large-scale sensor networks. In *Proc. of the 1st IEEE Intl. Conference on Mobile Ad Hoc and Sensor Systems*, 2004, pages 475–483.
12. S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. B. Srivastava. Coverage problems in wireless ad-hoc sensor networks. In *Proc. IEEE Infocom*, 2001, pages 1380–1387.
13. S. Meguerdichian, F. Koushanfar, G. Qu, and M. Potkonjak. Exposure in wireless ad-hoc sensor networks. In *Proc. of the 7th Intl. Conference on Mobile Computing and Networking*, 2001, pages 139–150.
14. S. Sahni and X. Xu. Algorithms for wireless sensor networks. *Intl. Jr. on Distr. Sensor Networks*, 1(1):35–56, 2005.