EuroCG 2014, Ein-Gedi, Israel, March 3–5, 2014

Hybrid Algorithms for Scheduling Sensors for Guarding Polygonal Domains

Esther M. Arkin* Alon Efrat† Joseph S. B. Mitchell ‡ Eli Packer§

Abstract

The art gallery problem models one aspect of optimally placing sensors to do visibility coverage in geometric domains. We define and solve a new version of this problem in which each guard/sensor can be functional only for a limited period of time (e.g. due to limited battery life), and the task is to schedule sets of sensors within the domain to maximize the total time that the domain is covered. We present an optimal (but rather slow) algorithm and an approximating heuristic for the problem. We show how these algorithms could be combined to achieve optimal results more efficiently. We implemented both solutions and experimented with many different input sets. Our experiments verify that our hybrid technique significantly decreases the running time of the exact algorithm, while maintaining optimality.

1 Introduction

The art gallery problem is a well studied computational and combinatorial geometry problem, with numerous variations introduced over the years. The essence of the problem is to "guard" or "cover" a geometric a domain while minimizing resources (e.g., the number of guards). It has been shown that the problem is not easy to solve: most natural variants are known to be NP-hard and difficult to approximate. For the problem of minimizing guards within a simple polygon, there is an exact algorithm [3], which is not polynomial-time, and a pseudo-polynomial $O(\log opt)$-approximation algorithm, where opt is the optimal number of guards [2]. In addition, experiments have shown [5, 9] that optimal and nearly-optimal solutions can be computed in practice using heuristics and combinatorial optimization techniques.

*Dept. Applied Mathematics and Statistics, Stony Brook University, esther.arkin@stonybrook.edu, partially supported by NSF (CCF-1018388).

†Department of Computer Science, University of Arizona, alon@cs.arizona.edu

‡Dept. of Applied Mathematics and Statistics, Stony Brook University, jbsm@ams.stonybrook.edu, partially supported by NSF (CCF-1018388).

§IBM Research Center, Haifa, Israel elip@il.ibm.com

Here, we consider a model with the following inputs: (1) a set S of identical (point) sensors, each equipped with a limited (known) lifetime battery; (2) a set W of *witness points* that need to be guarded by the sensors; and, (3) a polygonal domain (polygon with holes), P, which constitutes the environment in which the sensors and the witness points live. We assume that sensors can see in all directions to any distance. We say that a sensor $s \in S$ *guards* a witness point $w \in W$ if the line segment sw lies within P. The task is to schedule the sensors in order to maximize the total time that the witness points remain guarded by the sensors. Our task is to compute a collection of subsets of S, together with an interval of time each subset is activated, while respecting the constraint that no sensor is activated for more time than its battery allows.

Our Contribution. The problem of sensor scheduling has been intensively studied (see, e.g., [7, 4, 8, 1, 6]). However, we are not aware of prior work that addresses sensors whose coverage is explicitly based on visibility constraints within a geometric domain. We developed an optimal (but slow) solution to the problem based on linear and integer programming techniques. We also devised a simple greedy (polynomial-time) heuristic to obtain a feasible solution. We then combined these into a hybrid approach that utilizes the greedy heuristic to reduce the computational cost of the exact method. We experimentally show that this hyrbid approach acheives optimality far faster than the straightforward exact algorithm does.

2 Our Techniques

To optimally solve the scheduling problem, we use linear and integer programming to iteratively converge to the optimal solution. The idea is to iteratively add guarding sets to improve the total duration of coverage. Once an optimal solution is obtained, we identify it, and the process terminates. Within this framework, having a good starting solution may decrease the number of iterations, thereby leading to significantly faster convergence to optimality. Our hybrid solution is based on a greedy heuristic to find good starting guarding sets, followed by the iterative exact algorithm. We denote the iterative exact method by

ITER and the greedy heuristic method by GREEDY. In Section 3 we present experimental evidence of the advantages of the hybrid solution.

2.1 Preliminaries

The input consists of polygon P (with or without holes), and sets of sensors S and witness points W inside P. Each sensor $s \in S$ has a limited *lifetime* during which it could be active. A set $C \subseteq S$ of (active) sensors is called a *cover* if every witness point is seen by at least one $c \in C$. We normalize the lifetime of each sensor to be one time unit. The solution is a collection of m covers, $C_i \subseteq S$, each with an associated length of time, t_i, that it is active. Note that the sets $\{C_i : 1 \leq i \leq m\}$ are not necessarily disjoint (that is, a sensor could be active in multiple covers). Since the lifetime of each sensor is limited to one unit, we require that for each $s \in S$, $\sum_{i:s \in C_i} t_i \leq 1$. The overall schedule activates the sensors in each cover C_i, for an amount of time t_i, with the covers C_i in an arbitrary order. The measure of our result (the objective function that we wish to maximize) is simply the total duration time of all the covers: $\sum_{1 \leq i \leq m} t_i$.

2.2 The ITER Algorithm

Optimal solution for a given set of covers. Given a collection of covering sets $\mathcal{C}$, we find an optimal duration assignment by means of a simple linear programming formulation. Let $n = |S|$ be the number of sensors. Starting with a collection, $\mathcal{C}$, of covering sets, we iteratively enlarge it, each time adding one cover that either maintains or improves the total duration of the collection, until we reach optimum.

Let $M_{n \times m}$ be a matrix whose columns correspond to the covers $\vec{C_1} \ldots \vec{C_m} \in \mathcal{C}$. (Note that $\mathcal{C}$ is only a subset of the exponentially large set of all possible covers.) The jth entry of $\vec{C_i}$ is 1 if sensor j belongs to the cover set C_i and is 0 otherwise. Let x_i be the time associated with cover C_i. Our goal is to maximize the total duration of all covers in $\mathcal{C}$, subject to the constraint that each sensor is active for a total time of at most one time unit. Thus, we obtain the following linear program:

(1) Find $\vec{x} = (x_1 \ldots x_m) = \arg\max \vec{x} \cdot \vec{1}$ **subject to**
$\mathbf{M} \cdot \vec{x} \leq \vec{1}$
$\vec{x} \geq 0$

Improving the collection of covers. The dual of the above (primal) linear program is as follows.

(2) Find $\vec{y} = (y_1 \ldots y_n) = \arg\min \vec{y} \cdot \vec{1}$ **subject to**
$\mathbf{M}^t \cdot \vec{y} \geq \vec{1}$
$\vec{y} \geq 0$

Note that in the dual problem each row corresponds to a cover. In order to improve the collection of covers, we want to add a cover to the collection $\mathcal{C}$ so that the total duration (the objective of the primal) goes up. By LP duality theory, this is equivalent to finding a cover C' to add to $\mathcal{C}$ such that $C' \cdot y < 1$ (the dual constraint is violated by the current solution y); then, adding the constraint corresponding to C' to (2) causes the solution y to change and the optimal objective of the dual to go up or to stay the same (adding the constraint cannot cause the minimization objective to decrease). Note that if we cannot find such a cover C', then it means that the current vector y satisfies all covers (in the sense that $C' \cdot y \geq 1$ for all possible covers C') and thus constitutes an overall optimal solution to the full problem (which considers all exponentially many possible covers). Thus, we want to find a cover C' for which $C' \cdot y < 1$ (for the current vector y that solves (2)), or conclude that no such cover exists. The problem of computing such a cover is readily formulated as an integer programming problem:

(3) Find $\vec{C} = (c_1 \ldots c_n) = \arg\min \sum_{i=1}^{n} y_i c_i$ **subject to**
$c_i \in \{0, 1\}$
$\forall j, \sum_{i=1}^{n} v_{ij} c_i \geq 1$

Here, the constraint matrix V specifies the visibility relationship: $v_{ij} = 1$ iff sensor i sees witness point j. The constraint $\sum_{i=1}^{n} v_{ij} c_i \geq 1$ ensures that each witness point is seen (i.e., that $\vec{C}$ corresponds to a valid cover). Note that the y_i's are constants here; they came from the solution to (2). If we find a solution $\vec{C} = (c_1 \ldots c_n)$ for which $\sum_{i=1}^{n} y_i c_i < 1$, we add C to the collection $\mathcal{C}$, yielding a new row in (2) (column in (1)), solve the program and continue. If, on the other hand, $\sum_{i=1}^{n} y_i c_i \geq 1$ holds for all possible choices of $\vec{C}$, we conclude that an optimal solution has been found, and the algorithm halts. We conclude with a pseudocode to summarize the overall algorithm.

Sensor Scheduling
1. Consider the primal LP (1) and the dual LP (2). Solve (2), obtaining y.
2. Solve the integer programming problem (3), using y in the objective; let c denote the solution
3. If the cost associated with c is less than 1, add the corresponding cover to $\mathcal{C}$, and goto Step 1
4. Obtain the solution of the primal (1) from the dual (2)

2.3 The GREEDY Heuristic

As discussed in the previous section, ITER starts with any collection of guarding sets (covers); the collection need not be close to optimal. Given that the iterations are computationally costly (asymptotically exponential), we are motivated to minimize the number of iterations in order to decrease the total processing time significantly. Thus, our goal is to find a good starting collection of covers that, on the one hand, will not be too large and, on the other hand, will be a good starting point from which optimality will be obtained relatively fast.

We chose a greedy heuristic similar to the one used by Amit et al. [9]. The idea is to build sets of guarding sets (covers) incrementally, assuming that the duration associated with each cover will be one, so that each sensor uses up its energy in a single iteration and thus appears in at most one guarding set. The construction of each guarding set proceeds as follows. We first build the visibility graph of the sensors and the witness points. Then, we start by choosing the sensor that sees the most witness points. Then we mark the seen witness points and look for another sensor that sees the most unseen witness points, etc. We continue in this fashion until all witness points are seen; this results in one cover set. The next covering set is constructed similarly, ignoring the sensors that have been chosen before (since their lifetime is assumed now to have been depleted). We continue constructing guarding sets until we fail to be able to cover the witness points with the remaining sensors. Upon failure, the collection of guarding sets we constructed in this greedy process will be used now as the starting point for ITER.

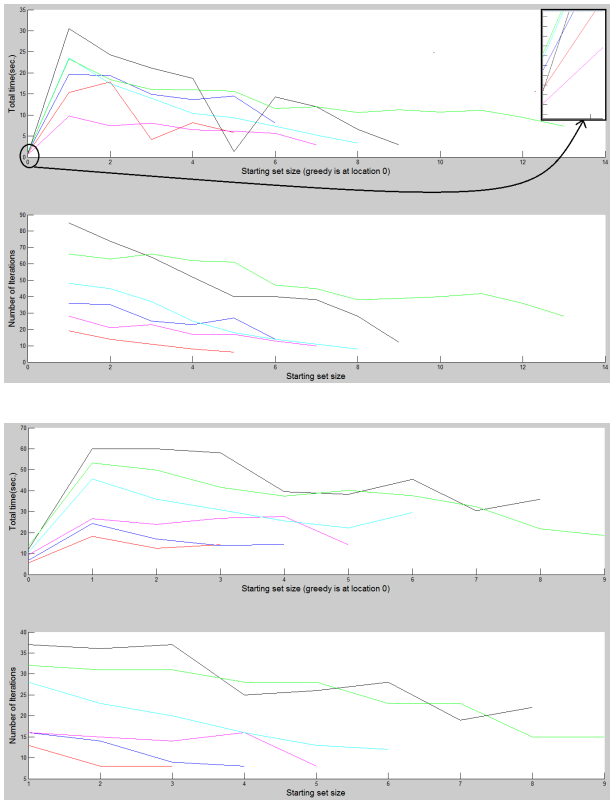


Figure 1: Results of running our tool on the data shown at Figure 3(b) and (c) (top and bottom, respectively). We ran each data several times with 70 random guard candidates and 20 witness points. Each produced a different number of guarding sets. Each color represents one activation of the tool, showing the running time and the number of iterations needed for ITER to complete.

3 Experiments

We implemented the techniques we describe in the previous section and experimented with many varieties of polygons, each with different sets of sensors and witness points chosen randomly inside the polygons. Our goal was to identify several charectersitics of the technqiues and, in particular, to evaluate the effectiveness of the hybrid solution. We do so by comparing the results of ITER with different subsets of covers generated by GREEDY. Figure 3 depicts three polygons used in the experiemnts.

An interesting observation concerns the results of GREEDY. It produced optimal solutions (identical to ITER) in almost all experiments. Even in the cases for which it was not optimal, it was less than one unit away from optimality. Figure 3(a) shows a special case in which

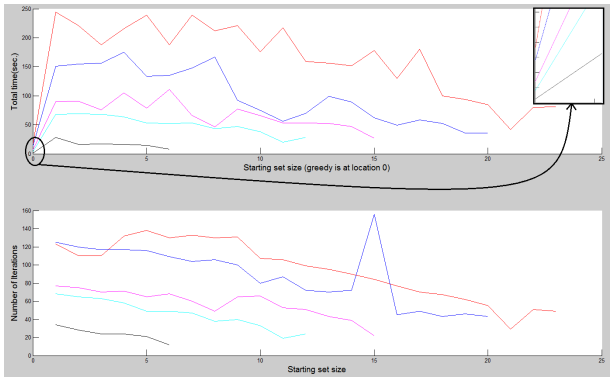
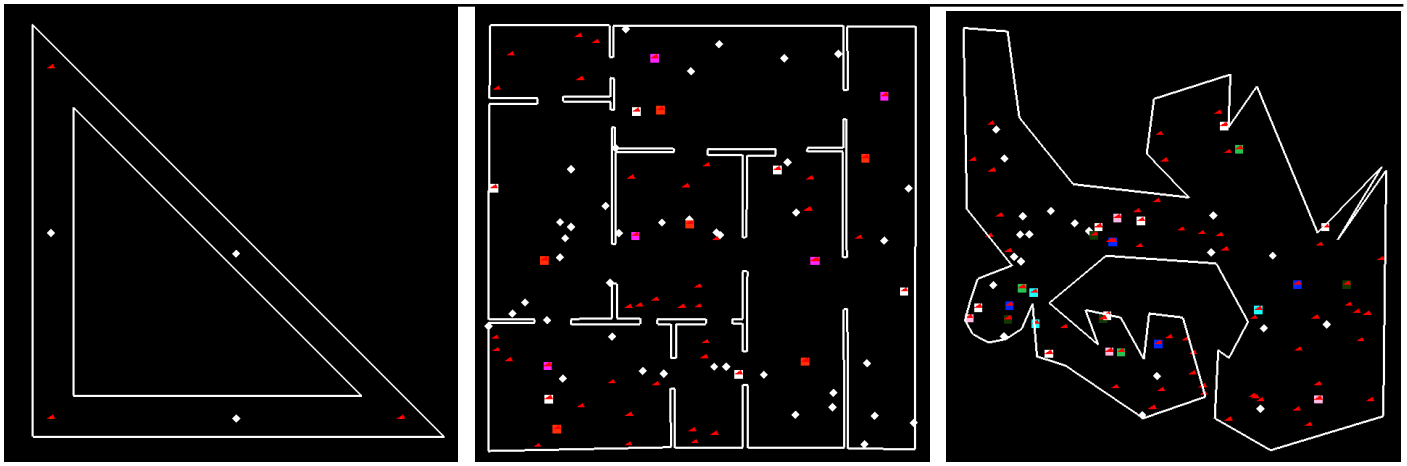


Figure 2: Results of running our tool on the data shown at Figure 3(b). We ran with a different number of sensors (50 to 250, in increments of 50). In the top subfigure the graphs from top to bottom correspond to decreasing numbers of sensors.



(a) (b) (c)

Figure 3: Results of our experiments. The red triangles are the sensors. The white diamonds are the witness points. The squares represent the guarding sets—each set has a unique color to distinguish among the sets. In all cases, except the one on the left, the sensors and witness points were chosen randomly inside the polygons.

GREEDY gives a solution that is 50% away from optimality: GREEDY selects two arbitrary sensors to cover the three witness points for one one unit of time, while the optimal solution is to activate each pair of sensors for half a unit of time, resulting in a total duration of 1.5 time units. However, we could not find any other case in which GREEDY was not very close to optimal, when not identical to it. Another interesting observation is that in most cases ITER was iterating with many fractional temporary solutions, converging to integral ones.

To evaluate our hybrid solution, we used the results of GREEDY as initial settings. We tried many subsets of the GREEDY results. Figures 1 and 2 show the results. We learned that it takes many iterations for ITER to converge to optimality. Increasing the initial guarding set size in principal decreased the number of iterations and the total time. Although the decrease was very noisy and not monotone, the general decrease is visible. It was also not surprising that the computation time of GREEDY was significantly smaller than that of ITER. Overall, it is evident that we achieved a significant speed up by using our hybrid solution—taking the complete GREEDY solution as the input for ITER improved the performance significantly when comparing it with starting with some arbitrary cover (in our case one cover from the GREEDY result). The speed-up factor ranged between 2 and 10.

References

[1] A. Deshpande, S. Khuller, A. Malekian, and M. Toossi. Energy efficient monitoring in sensor networks. In *LATIN 2008: Theoretical Informatics*, pages 436–448. Springer, 2008.

[2] A. Deshpande, T. Kim, E. D. Demaine, and S. E. Sarma. A pseudopolynomial time (log)-approximation algorithm for art gallery problems. In *WADS*, pages 163–174, 2007.

[3] A. Efrat and S. Har-Peled. Guarding galleries and terrains. In *IFIP TCS*, pages 181–192, 2002.

[4] M. Gibson and K. Varadarajan. Decomposing coverings and the planar sensor cover problem. In *Foundations of Computer Science, 2009. FOCS'09. 50th Annual IEEE Symposium on*, pages 159–168. IEEE, 2009.

[5] A. Kröller, T. Baumgartner, S. P. Fekete, , and C. Schmidt. Exact solutions and bounds for general art gallery problems. *ACM Journal of Experimental Algorithmics*, 17, 2012.

[6] J. Pach, D. Pálvölgyi, and G. Toth. Survey on the decomposition of multiple coverings. *To appear*, 2012.

[7] J. Pach and G. Tóth. Decomposition of multiple coverings into many parts. *Computational Geometry*, 42(2):127–133, 2009.

[8] K. Varadarajan. Weighted geometric set cover via quasi-uniform sampling. In *Proceedings of the 42nd ACM symposium on Theory of computing*, pages 641–648. ACM, 2010.

[9] Y.Amit, J. Mitchell, and E.Packer. Locating guards for visibility coverage of polygons. *Int. J. Comput. Geometry Appl.*, 20:601–630, 2010.