Guarding Monotone Polygons with Half-Guards

Matt Gibson* Erik Krohn† Matthew Rayford‡

Abstract

We consider a variant of the art gallery problem where all guards are limited to seeing to the right inside of a monotone polygon. We provide a polynomial-time 4-approximation algorithm for a version of the problem where we wish to point-guard the vertices of the polygon. We then extend this algorithm and provide a $O(1)$-approximation to point-guard the boundary of the polygon and ultimately the entire polygon.

1 Introduction

In the geometric set cover problem, we are given some set of points P and a set S where each $s \in S$ can cover some subset of P. The subset of P is generally induced by some geometric object. For example, P might be a set of points in the plane, and s consists of the points contained within some disk in the plane. For most variants, the problem is NP-hard and can easily be reduced to an instance of the combinatorial set cover problem which has a polynomial-time $O(\log n)$-approximation algorithm—the best possible approximation under standard complexity assumptions [5, 18, 10, 12, 17]. The main question is to then determine which of the geometric set cover problem variants we can obtain polynomial-time approximation algorithms with approximation ratio $o(\log n)$, as any such algorithm must exploit the geometry of the problem to achieve the result. This area has been studied extensively, see for example [4, 21, 3], and much progress has been made utilizing algorithms that are based on solving the standard linear programming relaxation.

Unfortunately, these techniques do not work for set cover variants based on visibility, such as the well-known *art gallery problem*. An instance of the art gallery problem takes as input a simple polygon P. The polygon P is defined by a set of points $V = \{v_1, v_2, \ldots, v_n\}$. There are edges connecting $\{v_i, v_{i+1}\}$ where $i = 1, 2, \ldots, n-1$ and an edge connecting $\{v_n, v_1\}$. If these edges do not intersect other than at the points in V, then P is called a simple polygon. The edges of a simple polygon give us two disjoint regions: inside the polygon and outside the polygon. For any two points $p, q \in P$, we say that p sees q if the line segment $\overline{pq}$ does not go outside of P. The art gallery problem seeks to find a set of points $G \subseteq P$ such that every point $p \in P$ is seen by some point in G. We call this set G a guarding set. In the point-guarding problem, guards can be placed anywhere inside of P. In the vertex guarding problem, guards are only allowed to be placed at vertices in V. The optimization problem is thus defined as finding the smallest such G in each case.

These problems are motivated by applications such as line-of-sight transmission networks in terrains, signal communications and broadcasting, cellular telephony systems and other telecommunication technologies as well as placement of motion detectors and security cameras.

1.1 Previous Work

The question of whether guarding simple polygons is NP-hard was independently confirmed by Aggarwal [2] and Lee and Lin [16]. They showed that the problem is NP-hard for both vertex guarding and point-guarding.

Along with being NP-hard, Brodén et al. [6] and Eidenbenz [9] independently proved that point-guarding simple polygons is APX-hard. This means that there exists a constant $\epsilon > 0$ such that no polynomial-time algorithm can guarantee an approximation ratio of $(1+\epsilon)$ unless P = NP. Ghosh provides a $O(\log n)$-approximation for the problem of vertex guarding an n-vertex simple polygon in [11]. This result can be improved for simple polygons using randomization, giving an algorithm with expected running time $O(nOPT^2 \log^4 n)$ that produces a vertex guard cover with approximation factor $O(\log OPT)$ with high probability, where OPT is the smallest vertex guard cover for the polygon [8]. Whether a polynomial time constant factor approximation algorithm can be obtained for vertex guarding a simple polygon is a longstanding and well-known open problem. Deshpande et al. [7] present a pseudopolynomial randomized algorithm for finding a point-guard cover with approximation factor $O(\log OPT)$. King and Kirkpatrick provide a $O(\log \log OPT)$-approximation algorithm for the problem of guarding a simple polygon with guards on the perimeter in [13]. The point-guarding problem seems to be much more difficult and little is known about it [7].

Additional Polygon Structure. Due to the inherent difficulty in fully understanding the art gallery problem for simple polygons, there has been some work done guarding polygons with some additional structure. A simple polygon P is x-monotone (or simply monotone) if any vertical line intersects the boundary of P at most

*University of Texas at San Antonio, gibson@cs.utsa.edu
†University of Wisconsin - Oshkosh, krohne@uwosh.edu
‡University of Wisconsin - Oshkosh, rayfom16@uwosh.edu

twice. Let l and r denote the leftmost and rightmost vertices of P, respectively. Consider the "top half" of the boundary of P by walking along the boundary clockwise from l to r. We call this the *ceiling* of P. Similarly, we obtain the *floor* of P by walking clockwise along the boundary from r to l. Notice that both the ceiling and the floor are x-monotone polygonal chains—that is a vertical line intersects it in at most one point. Krohn and Nilsson [15] give a polynomial-time constant factor approximation algorithm for point-guarding monotone polygons. They also proved point-guarding and vertex guarding a monotone polygon is NP-hard [14, 15].

α-Floodlights. Motivated by the fact that many cameras and other sensors generally are not able to sense in 360°, previous works have considered the problem when guards have a fixed sensing angle α for some $0 < \alpha \leq 360$. This problem is often referred to as the *α-floodlight problem*. 180°-floodlights are sometimes referred to as *half-guards*. Some of the work on this problem has involved proving necessary and sufficient bounds on the number of α-floodlights required to guard (or illuminate) an n vertex simple polygon P, where floodlights are anchored at vertices in P and no vertex is assigned more than one floodlight, see for example [19, 20]. It is known that computing a minimum cardinality set of α-floodlights to illuminate a simple polygon P is APX-hard for both the point-guard and vertex guard variants [1].

1.2 Our Contribution

In this paper, we consider guarding monotone polygons with half-guards that can see in one direction, namely to the right. Let $p.x$ denote the x-coordinate of a point p. We modify visibility in that the definition of *sees* is changed to: a point p sees a point q if the line segment $\overline{pq}$ does not go outside of P and $p.x \leq q.x$.

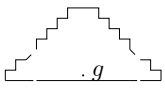

Figure 1

Our main result is to give a polynomial-time constant factor approximation algorithm for point-guarding monotone polygons with half-guards that see to the right. Krohn and Nilsson [15] obtained a similar result using full guards, but the algorithms are quite different and many new observations are needed to obtain the algorithm given in this paper. Indeed, note that there are monotone polygons P that can be covered with one full guard that require $\Omega(n)$ guards considered in this paper (see for example, Figure 1).

The remainder of the paper is sectioned as follows: in Section 2, we provide a 4-approximation for point-guarding a monotone polygon using half-guards where we wish to guard only the vertices of the polygon. In Section 3, we extend the algorithm given in Section 2 to provide a 20-approximation for guarding the entire boundary of the polygon. In Section 4, we extend the algorithm given in Section 3 to provide a 40-approximation for guarding the entire polygon.

2 Guarding the Vertices

In this section, we give a polynomial-time 4-approximation algorithm for guarding the vertices of a monotone polygon P with guards that see to the right. We do this by first giving a 2-approximation algorithm for guarding the vertices of the ceiling. We then have the algorithm for the entire polygon by symmetrically applying the ceiling algorithm to the vertices of the floor, giving a 4-approximation for guarding all vertices of P.

Before we describe the algorithm, we provide some preliminary definitions. The rightmost vertex that a point p sees on the ceiling is denoted $R_c(p)$. A vertical line that goes through a point p is denoted l_p. Given two points p, q in P such that $p.x < q.x$, we use (p, q) to denote the points r such that $p.x < r.x < q.x$. Similarly, we use $(p, q]$ to denote points r such that $p.x < r.x \leq q.x$, etc.

2.1 Ceiling Guard Algorithm

We first give a high level overview of our algorithm for guarding the vertices of the ceiling. Any feasible solution must place a guard at the leftmost vertex of the ceiling (or this vertex will not be covered). We begin by placing a guard here, and we iteratively place guards from left to right. When placing a new guard, we let S denote the guards we have already placed, and we let p denote the leftmost vertex on the ceiling that is not seen by a guard in S. The next guard we place, g, will lie somewhere on the line l_p. We initially place g at the intersection of l_p and the floor, and we slide g vertically along l_p until some condition holds. Let $C(S)$ denote the set of ceiling vertices seen by S, and let $C(g)$ denote the set of ceiling vertices seen by g. Note that as g slides up l_p, ceiling vertices may join and leave $C(g)$ as the vertices on the ceiling that g sees may change. Our algorithm locks in a final position for the guard g by sliding it vertically along l_p until moving it any higher will cause g to no longer see some vertex in $C(g) \setminus C(S)$ (the ceiling vertices seen by g that are not seen by any previously placed guard). See, for example, Figure 2. In this figure, initially g does not see v, but as we slide g up the line l_p, v becomes a new vertex in $C(g) \setminus C(S)$. If we slide g up any higher than as depicted in the figure, then g would no longer see v, and therefore we lock in the position of g. We then add g to S, and we repeat this procedure until all vertices on the ceiling are guarded. The formal ceiling guarding algorithm is shown in Algorithm 1.

This algorithm clearly returns a set of guards that sees every vertex on the ceiling. All steps, except the sliding step, can be trivially done in polynomial time. Since the polygon is simple, any vertex that is seen by a point on l_p must by seen by a contiguous line segment of l_p. We

Algorithm 1 Ceiling Guard

```
1: procedure CEILING GUARD(monotone polygon P)
2:     S ← {s} such that s is placed at the leftmost point l.
3:     while there is an unseen ceiling vertex do
4:         Let p be the leftmost ceiling vertex that is currently unseen by any guards in S. Initially place a guard g where l_p intersects the floor. Slide g up until it stops seeing some vertex v ∈ C(g) \ C(S) on the ceiling. Place g at the point on l_p just before it stopped seeing v.
5:         S ← S ∪ {g}.
6:     end while
7:     return S
8: end procedure
```

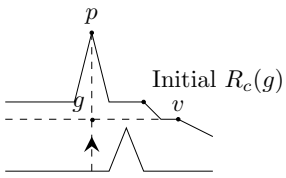

Figure 2: Moving g vertically on l_p until it stops seeing some ceiling vertex v.

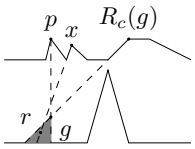

Figure 3: A point r is g-ceiling-dominated.

compute these line segments for each vertex that is seen from l_p. This takes $O(n^2)$ time. In the sliding step, we only need to consider the top and bottom points of these line segments. There are at most $2n$ of these points on l_p to consider during the sliding step. Therefore, Algorithm 1 runs in polynomial time. It remains to prove the approximation ratio.

2.2 Proof of Approximation

In this subsection, we will show that Algorithm 1 will place no more than 2 times the number of guards in the optimal solution. An optimal solution $\mathcal{O}$ is a minimum cardinality guard set such that for any vertex v on the ceiling of P, there exists some $g \in \mathcal{O}$ that sees v. The argument will be a charging argument; every guard we place will be charged to a guard in $\mathcal{O}$ in a manner such that each guard in $\mathcal{O}$ will be charged at most twice.

We now provide a key lemma that will be used to show that we do not charge a guard of $\mathcal{O}$ more than twice. Consider some guard g chosen by the algorithm, and let S_g denote the set of guards consisting of g and every guard that we chose prior to g. For any point r in P, we say that r is *g-ceiling-dominated* if every ceiling vertex to the right of g seen by r is also seen by some $g' \in S_g$.

Lemma 1 *Consider a guard g placed in step 5 of the algorithm. A point r that is to the left of g and below the ray $\overrightarrow{R_c(g)g}$ is g-ceiling-dominated.*

Proof. Let x denote a ceiling vertex that is seen by r to the right of p (inclusive). The proof will consider two cases depending on if the line segment $\overline{rx}$ intersects l_p below or above g. In both scenarios, we prove that x must be seen by some guard in S_g. The lemma immediately follows. We will again let S denote the guards placed prior to g (i.e., $S = S_g \setminus \{g\}$).

First suppose that $\overline{rx}$ intersects l_p at g or below g. While sliding g up l_p, it would have passed through the intersection point of $\overline{rx}$ and l_p, and therefore g saw x at this point in time. If x is not seen by some guard in S, then the final placement of g must see x as well. If the final placement of g is such that g does not see x, then it must be that x was already seen by some guard in S. Therefore it must be that x is seen by some guard in S_g.

Now let x be such that $\overline{rx}$ intersects l_p strictly above the final placement of g. For this to be the case, it must be that x is in $[p, R_c(g)]$ since r is left of g and is below $\overrightarrow{R_c(g)g}$. We will show that g must also see x. If g does not see x then either the floor must "pierce" $\overline{gx}$ from below or the ceiling must pierce $\overline{gx}$ from above. The floor cannot block g from x because it would also block g from $R_c(g)$, and the ceiling cannot block g from x because otherwise it would also block r from x. Therefore g sees x. See Figure 3. □

We now describe our method of charging the guards chosen by our algorithm to the guards in $\mathcal{O}$. When we place a guard g, we will charge g to some guard in $\mathcal{O}$ to the left of g. We prove by induction that when we place our guard g, we can charge g to a guard in $\mathcal{O}$ that has previously been charged at most once.

Base case: Our algorithm places a guard at the leftmost point and there must also be an optimal guard at this point. If this were not the case, then the optimal solution would not have guarded the leftmost point. We charge our guard to this optimal guard. Our base case then considers the first guard our algorithm places in the while loop. Consider the placement of this guard g. Let p be the first ceiling vertex not seen by the initial guard; the initial optimal guard also does not see p. Therefore, the optimal solution must have an uncharged guard o on l_p or to the left of l_p, and we can charge g to o.

Inductive Step: We assume the inductive hypothesis holds true for the first $k-1$ iterations of the while loop and we are on the k^{th} iteration. We consider the placement of guard g and the vertical line l_p that it is on. Let f denote the guard placed in iteration $k-1$, and let l_f denote the vertical line it is on, see Figure 4. We consider two cases depending on whether there is a guard in $\mathcal{O}$ that is in $(f, g]$.

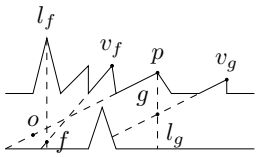

Figure 4: An optimal guard o to the left of l_f that can see p.

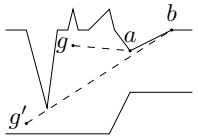

Figure 5: Guard g does not see b since a is blocking it; $\{a, b\}$ is not entirely seen.

Case 1: There is an $o \in \mathcal{O}$ in $(f, g]$. All previously charged guards $o' \in \mathcal{O}$ satisfy $o'.x \leq f.x$, and therefore o has not had a guard charged to it. We charge g to o.

Case 2: There is no optimal guard in $(f, g]$. In this case, we show that an optimal guard o' is f-ceiling-dominated when f was placed and o' was not f'-ceiling-dominated for any guard f' placed by the algorithm prior to placing f. We charge g to o'. An optimal guard o' can only be charged by this procedure in the iteration *immediately after* it becomes dominated, and therefore it can only be charged once by this case.

Since there is no optimal guard in $(f, g]$, p must be seen by an optimal guard o such that $o.x \leq f.x$. We will show that the line segment $\overline{op}$ must cross the line l_f strictly above f. If it crosses l_f through or below the final placement of f, then f must have seen p at some point while sliding vertically. It follows that either f sees p or some previous guard sees p, a contradiction. Therefore it must be that $\overline{op}$ crosses l_f strictly above f.

When we placed f, it slid vertically until it would have lost sight of some ceiling vertex v_f that was not seen by any previous guard. We will first show that v_f must be to the left of p. Since f does not see p, some part of P must block f from seeing p. The ceiling cannot block f from p because it would also block o from seeing p (since $\overline{op}$ crosses above f). If v_f were to the right of p, then the floor would not be able to block f from p either because it would also block f from seeing v_f. Therefore it must be that v_f is to the left of p.

Now let $o' \in \mathcal{O}$ denote an optimal guard that sees v_f. We will show that o' is f-ceiling-dominated by Lemma 1. Since we are in Case 2, it must be that $o'.x \leq f.x$. It remains to show that o' is below the ray $\overrightarrow{R_c(f)f}$. We will do this by showing that o' is below the ray $\overrightarrow{v_f f}$, and therefore must also be below $\overrightarrow{R_c(f)f}$ if $v_f \neq R_c(f)$. It follows that o' is below $\overrightarrow{v_f f}$ due to the manner in which the location of f was chosen. The point f stopped sliding when it would have lost sight of v_f, and in particular, it must be that the *ceiling* would prevent f from seeing v_f because f is only sliding vertically. Therefore any point in P that is to the left of f and is above the ray $\overrightarrow{v_f f}$ must also be blocked from seeing v_f. Since o' sees v_f, it must be that o' is below the ray $\overrightarrow{v_f f}$, satisfying the conditions of Lemma 1, and therefore is f-ceiling-dominated.

We charge g to o'. Since o' sees v_f (v_f was not guarded by our algorithm until we placed f), it must be that o' was not dominated prior to the placement of f. Thus o' can be charged a guard by this procedure only once.

This completes our charging scheme, which charges each guard g picked by our algorithm to an optimal guard in $\mathcal{O}$. We have shown that each guard in $\mathcal{O}$ can be charged at most once in Case 1 and at most once in Case 2, and therefore our algorithm returns a set of guards of size at most $2|\mathcal{O}|$. By running this algorithm on the ceiling and symmetrically applying the algorithm on the floor, we have the following theorem.

Theorem 2 *There is a polynomial-time 4-approximation algorithm for point-guarding the vertices of a monotone polygon with half-guards.*

3 Guarding the Boundary

In the previous section, we provided an algorithm to guard the vertices of the polygon with at most 4OPT guards. However, the algorithm is not guaranteed to guard the entire boundary, see Figures 5 and 6 for example. We will now provide a modification of Algorithm 1 to ensure that the entire boundary is seen. Similar to the last section, we begin by providing a polynomial-time 10-approximation algorithm that will cover the entire ceiling. This algorithm can be symmetrically applied to the floor to then give a polynomial-time 20-approximation algorithm that covers the entire boundary of P.

Suppose we have a guard set S that covers all of the vertices of the ceiling, and consider some edge $\{a, b\}$ on the ceiling such that a is to the left of b. If one guard $g \in S$ sees both a and b, then it is easy to see that g sees the entire edge $\{a, b\}$. Therefore, if some edge is not completely covered by S, then it must be that every guard that sees a does not see b (and vice versa). At a high level, our algorithm for guarding the entire ceiling begins by covering the vertices of the ceiling similarly to Algorithm 1. If at some point in time during this process we have that our current guard set sees both vertices of an edge but does not cover the entire edge, then we place additional guards to ensure that the entire edge is indeed covered.

For ease of description, we maintain two different sets of guards: S and S'. S is the set of guards chosen to cover vertices (similar to Algorithm 1), and S' is the set of guards chosen to fill in a missing gap on some edge. To prove the approximation ratio, we charge each guard in S' to one of the guards in S. We prove that each guard of S will have at most four guards of S' charged to it. Since each guard of $\mathcal{O}$ has at most two guards of S charged to it, we then have that each guard in $\mathcal{O}$ has at most 10 guards of $S \cup S'$ charged to it, giving us the approximation ratio.

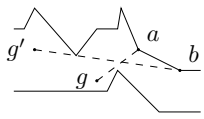

Figure 6: Guard g does not see b since the floor is blocking it; $\{a, b\}$ is not completely seen.

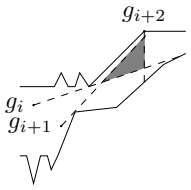

Figure 7: An example of a middle pocket.

Suppose we have just added a guard g to S so as to cover some vertices of the ceiling. Let E denote the set of edges such that for every edge in the set, the endpoints are seen by some guard in S but part of the edge is unseen (see step 6 of Algorithm 2). We prove that each edge $e = \{a, b\} \in E$ will fall into one of the four following cases, and each case is handled differently.
Case 1: g sees a and some guard in S sees b. a is blocking g from b. We note in this case that no other ceiling point can block g from b, see Figure 5.
Case 2: g sees a and some guard in S sees b. g does not see b because the floor is blocking g from b, see Figure 6.
Case 3: g sees b and some guard in S sees a. g does not see a because a is to the left of g.
Case 4: g sees b and some guard in S sees a. g does not see a because the ceiling is blocking g from a.

Due to lack of space, we omit the proofs that these four cases are exhaustive, and the algorithm places guards into S' in a way so that every edge in E is completely seen by S'. Moreover, we prove we can charge each guard of S' to a guard of S such that any guard in S is charged at most one guard for each of the four cases.

Now we have that each edge will be covered as soon as S sees both endpoints of the edge. At the end of the algorithm we have that every vertex on the ceiling is seen by a guard in S, we will have that $S \cup S'$ covers the entire ceiling. Each guard in S is charged at most one guard in S' per case, and therefore $|S'| \leq 4|S|$. We already had that $|S| \leq 2|\mathcal{O}|$, and thus $|S'| \leq 8|\mathcal{O}|$. Our final guarding set then satisfies $|S \cup S'| \leq 10|\mathcal{O}|$. By applying the algorithm on the ceiling and the floor, we have a 20-approximation algorithm for guarding the entire boundary of P.

Theorem 3 *There is a polynomial-time 20-approximation algorithm for point-guarding the boundary of a monotone polygon with half-guards.*

4 Guarding the Entire Polygon

Algorithm 2 ensures that the entire boundary of the polygon is seen. It is possible that parts of the interior of the polygon are unseen, see Figure 7. After Algorithm 2 is run, we run the final algorithm to ensure that the entire polygon is guarded. In this algorithm, we let S denote all guards returned by Algorithm 2. For any point p on the boundary of P, we let p^- denote a point on the boundary to the left of p that is "infinitesimally close" to p. A *middle pocket* is defined as an unseen part of the polygon that is not touching the boundary of the polygon. See Figure 7. The final algorithm will ensure that all middle pockets are guarded. The algorithm processes S from left to right and considers two consecutive guards. The algorithm places a guard at a strategic location ensuring

Algorithm 2 Modified Ceiling Guard

1: $S \leftarrow \{s\}$ such that s is placed at leftmost point l.
2: Let S' denote the initially empty set of "extra" guards we add to cover edges.
3: **while** there exists an unseen point on the ceiling of P from our guards in S **do**
4: Let p be the leftmost ceiling vertex that is currently unseen by any guards in S. Place a guard g where l_p intersects the floor. Slide g up until it stops seeing some vertex $v \in C(g)$ on the ceiling. Place g at the point just before it stopped seeing v. $S \leftarrow S \cup \{g\}$.
5: Let o be the vertex to the left of p on the ceiling.
6: Let E be the set of ceiling edges such that for every edge $e = \{a, b\} \in E$, g sees exactly one vertex on the edge, S only sees the other vertex on the edge, and S does not see the entire edge e.
7: **if** g sees some vertex a such that a is the leftmost vertex of some edge in E **then**
8: Let a_r be the rightmost a vertex that g sees, such that a is the leftmost vertex of some edge $e \in E$. Let b_r be vertex immediately to the right of a_r on the ceiling.
9: **if** a blocks g from b_r **then** ▷ Case 1
10: Let $g_{b_r} \in S$ be the leftmost guard that sees b_r. Draw a line l from g_{b_r} to b_r and place a guard g' at the point where l intersects l_p. Remove all edges in E that g' sees. $S' \leftarrow S' \cup \{g'\}$.
11: **end if**
12: **if** the floor blocks g from b_r **then** ▷ Case 2
13: Remove all edges in E that g' sees, and place a guard g' at a_r. $S' \leftarrow S' \cup \{g'\}$.
14: **end if**
15: **end if**
16: **if** a part of $\overline{ab}$ is not seen by S **then** ▷ Case 3
17: Remove $\{a, b\}$ from E, and place a guard g' at a. $S' \leftarrow S' \cup \{g'\}$.
18: **end if**
19: **while** E is not empty **do** ▷ Case 4
20: Remove $e = \{a, b\}$ from E and place a guard g' at a. $S' \leftarrow S' \cup \{g'\}$.
21: **end while**
22: **end while**
23: **return** $S \cup S'$.

that any part of the polygon that is unseen between the consecutive guards is now seen. The following lemma is proved in [15].

Lemma 4 *Consider a middle pocket p of a partial guard set S in a monotone polygon. Let r be the leftmost point in p. Not all guards of S can be to the left of r.*

Note that Lemma 4 is proved in [15] for guards that see in all directions, but it also trivially applies to our scenario since it deals with a region of P that lies entirely to the right of a set of guards.

Lemma 5 *Any middle pockets between 2 consecutive guards can be guarded with 1 guard.*

After the final algorithm terminates, the entire boundary is seen and all middle pockets are guarded; thus the entire polygon is seen. Note that each extra guard that the final algorithm places can be charged to S_i, and therefore each guard output by Algorithm 2 will be charged at most one guard placed by the final algorithm. We have the following theorem.

Theorem 6 *There is a 40-approximation algorithm for point-guarding a monotone polygon with half-guards.*

References

[1] Ahmed Abdelkader, Ahmed Saeed, Khaled A. Harras, and Amr Mohamed. The inapproximability of illuminating polygons by α-floodlights. In *CCCG*, pages 287–295, 2015.

[2] Alok Aggarwal. *The art gallery theorem: its variations, applications and algorithmic aspects*. PhD thesis, 1984.

[3] Greg Aloupis, Jean Cardinal, Sébastien Collette, Stefan Langerman, David Orden, and Pedro Ramos. Decomposition of multiple coverings into more parts. In *SODA*, pages 302–310, 2009.

[4] Boris Aronov, Esther Ezra, and Micha Sharir. Small-size epsilon-nets for axis-parallel rectangles and boxes. *SIAM J. Comp.*, 39(7):3248–3282, July 2010.

[5] M. Bellare, S. Goldwasser, C. Lund, and A. Russell. Efficient probabilistically checkable proofs and applications to approximations. In *STOC*, pages 294–304, 1993.

[6] Björn Brodén, Mikael Hammar, and Bengt J. Nilsson. Guarding lines and 2-link polygons is APX-hard. In *CCCG*, pages 45–48, 2001.

[7] Ajay Deshpande, Taejung Kim, Erik D. Demaine, and Sanjay E. Sarma. A pseudopolynomial time O(log n)-approximation algorithm for art gallery problems. In *WADS*, pages 163–174, 2007.

[8] Alon Efrat and Sariel Har-Peled. Guarding galleries and terrains. *Information Processing Letters*, 100(6):238–245, 2006.

[9] Stephan Eidenbenz. Inapproximability results for guarding polygons without holes. In *ISAAC*, pages 427–436, 1998.

[10] Uriel Feige, Magnús M. Halldórsson, Guy Kortsarz, and Aravind Srinivasan. Approximating the domatic number. *SIAM Journal on Computing*, 32(1):172–195, 2003.

[11] Subir Kumar Ghosh. On recognizing and characterizing visibility graphs of simple polygons. *Discrete & Computational Geometry*, 17(2):143–162, 1997.

[12] David S. Johnson. Approximation algorithms for combinatorial problems. STOC, pages 38–49. ACM, 1973.

[13] James King and David G. Kirkpatrick. Improved approximation for guarding simple galleries from the perimeter. *Discrete & Computational Geometry*, 46(2):252–269, 2011.

[14] Erik Krohn and Bengt J. Nilsson. The complexity of guarding monotone polygons. In *CCCG*, pages 167–172, 2012.

[15] Erik Krohn and Bengt J. Nilsson. Approximate guarding of monotone and rectilinear polygons. *Algorithmica*, 66(3):564–594, 2013.

[16] D. T. Lee and A. K. Lin. Computational complexity of art gallery problems. *IEEE Trans. Inform. Theory*, 32(2):276–282, March 1986.

[17] Carsten Lund and Mihalis Yannakakis. On the hardness of approximating minimization problems. *J. ACM*, 41(5):960–981, September 1994.

[18] Ran Raz and Shmuel Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. STOC, pages 475–484. ACM, 1997.

[19] Bettina Speckmann and Csaba D. Tóth. Allocating vertex π-guards in simple polygons via pseudo-triangulations. *Discrete & Computational Geometry*, 33(2):345–364, 2005.

[20] Csaba D. Tóth. Art galleries with guards of uniform range of vision. *Computational Geometry*, 21(3):185 – 192, 2002.

[21] Kasturi R. Varadarajan. Epsilon nets and union complexity. In *Symposium on Computational Geometry*, pages 11–16, 2009.