

# Improved Approximation for Guarding Simple Galleries from the Perimeter

James King · David Kirkpatrick

Received: 7 February 2010 / Revised: 22 February 2011 / Accepted: 8 April 2011 /  
Published online: 6 May 2011  
© Springer Science+Business Media, LLC 2011

**Abstract** We provide an  $O(\log \log \text{OPT})$ -approximation algorithm for the problem of guarding a simple polygon with guards on the perimeter. We first design a polynomial-time algorithm for building  $\varepsilon$ -nets of size  $O(\frac{1}{\varepsilon} \log \log \frac{1}{\varepsilon})$  for the instances of HITTING SET associated with our guarding problem. We then apply the technique of Brönnimann and Goodrich to build an approximation algorithm from this  $\varepsilon$ -net finder. Along with a simple polygon  $P$ , our algorithm takes as input a finite set of potential guard locations that must include the polygon's vertices. If a finite set of potential guard locations is not specified, e.g., when guards may be placed anywhere on the perimeter, we use a known discretization technique at the cost of making the algorithm's running time potentially linear in the ratio between the longest and shortest distances between vertices. Our algorithm is the first to improve upon  $O(\log \text{OPT})$ -approximation algorithms that use generic net finders for set systems of finite VC-dimension.

**Keywords** Art gallery problem · Polygon guarding · Approximation algorithm · Epsilon-nets · Perimeter guards · Vertex guards

---

Some of these results appeared in preliminary form as D. Kirkpatrick. *Guarding galleries with no nooks*. In *Proceedings of the 12th Canadian Conference on Computational Geometry (CCCG'00)*, pp. 43–46, 2000.

---

J. King (✉)  
School of Computer Science, McGill University, Montreal, Canada  
e-mail: [jking@cs.mcgill.ca](mailto:jking@cs.mcgill.ca)

D. Kirkpatrick  
Department of Computer Science, University of British Columbia, Vancouver, Canada  
e-mail: [kirk@cs.ubc.ca](mailto:kirk@cs.ubc.ca)

## 1 Introduction

### 1.1 The Art Gallery Problem

In computational geometry, art gallery problems are motivated by the question, “How many security cameras are required to guard an art gallery?” The art gallery is modeled as a connected polygon  $P$ . A camera, which we will henceforth call a *guard*, is modeled as a point in the polygon, and we say that a guard  $g$  *sees* a point  $q$  in the polygon if the line segment  $\overline{gq}$  is contained in  $P$ . We call a set  $G$  of points a *guarding set* if every point in  $P$  is seen by some  $g \in G$ . Let  $V(P)$  denote the vertex set of  $P$  and let  $\partial P$  denote the boundary of  $P$ . We assume that  $P$  is closed and non-degenerate so that  $V(P) \subset \partial P \subset P$ .

We consider the minimization problem that asks, given an input polygon  $P$  with  $n$  vertices, for a minimum guarding set for  $P$ . Variants of this problem typically differ based on what points in  $P$  must be guarded and where guards can be placed, as well as whether  $P$  is simple or contains holes. Typically, we want to guard either  $P$  or  $\partial P$ , and our set of potential guards is typically  $V(P)$  (vertex guards),  $\partial P$  (perimeter guards), or  $P$  (point guards). For results on art gallery problems not related to minimization problems we direct the reader to O’Rourke’s book [18], which is available for free online.

The problem was proved to be NP-complete first for polygons with holes by O’Rourke and Supowit [19]. For guarding simple polygons it was proved to be NP-complete for vertex guards by Lee and Lin [17]; their proof was generalized to work for point guards by Aggarwal [1]. This raises the question of approximability. There are two major hardness results. First, for guarding simple polygons, Eidenbenz [9] proved that the problem is APX-complete, meaning that we cannot do better than a constant-factor approximation algorithm unless  $P = NP$ . Subsequently, for guarding polygons with holes, Eidenbenz et al. [10] proved that the minimization problem is as hard to approximate as SET COVER in general if there is no restriction on the number of holes. It therefore follows from results about the inapproximability of SET COVER by Feige [11] and Raz and Safra [20] that, for polygons with holes, it is NP-hard to find a  $o(\log n)$ -approximation. These hardness results hold whether we are dealing with vertex guards, perimeter guards, or point guards.

Ghosh [13] provided an  $O(\log n)$ -approximation algorithm for guarding polygons with or without holes with vertex guards. His algorithm decomposes the input polygon into a polynomial number of cells such that each point in a given cell is seen by the same set of vertices. This discretization allows the guarding problem to be treated as an instance of SET COVER and solved using general techniques. This will be discussed further in Sect. 1.2. In fact, applying methods for SET COVER developed after Ghosh’s algorithm, it is easy to obtain an approximation factor of  $O(\log \text{OPT})$  for vertex guarding simple polygons or  $O(\log h \log \text{OPT})$  for vertex guarding a polygon with  $h$  holes.

When considering point guards or perimeter guards, discretization is far more complicated since two distinct points will not typically be seen by the same set of potential guards even if they are very close to each other. Deshpande et al. [7] obtain an approximation factor of  $O(\log \text{OPT})$  for point guards or perimeter guards by developing a sophisticated discretization method that runs in pseudopolynomial time.

It is a pseudopolynomial-time algorithm in that its running time may be linear in the ratio between the longest and shortest distances between two vertices. Efrat and Har-Peled [8] provided a randomized algorithm with the same approximation ratio that runs in fully polynomial expected time; their discretization technique involves only considering guards that lie on the points of a very fine grid.

Our contribution is an algorithm for guarding simple polygons, using either vertex guards or perimeter guards. Our algorithm has a guaranteed approximation factor of  $O(\log \log \text{OPT})$  and the running time is polynomial in  $n$  and the number of potential guard locations. This is the best approximation factor obtained for vertex guards and perimeter guards. If no finite set of guard locations is given, we use the discretization technique of Deshpande et al. and our algorithm is polynomial in  $n$  and  $\Delta$ , where  $\Delta$  is the ratio between the longest and shortest distances between vertices.

## 1.2 Guarding Problems as Instances of HITTING SET

### 1.2.1 Set Cover and Hitting Set

SET COVER is a well-studied NP-complete optimization problem. Given a universe  $\mathcal{U}$  of elements and a collection  $\mathcal{S}$  of subsets of  $\mathcal{U}$ , SET COVER asks for a minimum subset  $\mathcal{C}$  of  $\mathcal{S}$  such that  $\bigcup_{S \in \mathcal{C}} S = \mathcal{U}$ . In other words, we want to cover all of the elements in  $\mathcal{U}$  with the minimum number of sets from  $\mathcal{S}$ . In general, SET COVER is not only difficult to solve exactly (see, e.g., [12]) but is also difficult to approximate—no polynomial-time approximation algorithm can have a  $o(\log n)$  approximation factor unless  $P = NP$  [20]. Conversely, a simple greedy heuristic (repeatedly picking the set that covers the most uncovered elements) [6] for SET COVER attains an  $O(\log n)$  approximation factor. Another problem, HITTING SET, asks for a minimum subset  $\mathcal{H}$  of  $\mathcal{U}$  such that  $S \cap \mathcal{H} \neq \emptyset$  for any  $S \in \mathcal{S}$ . Any instance of HITTING SET can easily be formulated as an instance of SET COVER and vice versa.

### 1.2.2 Set Systems of Guarding Problems

Guarding problems can naturally be expressed as instances of SET COVER or HITTING SET. We wish to model an instance of a guarding problem as an instance of HITTING SET. The desired set system  $(\mathcal{U}, \mathcal{S})$  is constructed as follows.  $\mathcal{U}$  contains the potential guard locations. For each point  $p$  that needs to be guarded,  $S_p$  is the set of potential guards that see  $p$ , and  $\mathcal{S} = \{S_p \mid p \in P\}$ .

### 1.2.3 $\varepsilon$ -Nets

Informally, if we wish to relax the HITTING SET problem, we can ask for a subset of  $\mathcal{U}$  that hits all *heavy* sets in  $\mathcal{S}$ . This is the idea behind  $\varepsilon$ -nets. For a set system  $(\mathcal{U}, \mathcal{S})$  and an additive weight function  $w$ , an  $\varepsilon$ -net is a subset of  $\mathcal{U}$  that hits every set in  $\mathcal{S}$  having weight at least  $\varepsilon \cdot w(\mathcal{U})$ . An oft-exploited fact is that small  $\varepsilon$ -nets can be constructed if a set system has bounded *VC-dimension*.

**Definition 1** (VC-Dimension [22]) For a set system  $(\mathcal{U}, \mathcal{S})$ , let  $Y$  be a maximum cardinality subset of  $\mathcal{U}$  such that  $\mathcal{S} \cap Y = 2^Y$ . The *VC-dimension* of  $(\mathcal{U}, \mathcal{S})$  is equal to  $|Y|$ .

It is known that set systems of VC-dimension  $d$  admit  $\varepsilon$ -nets of size  $O(\frac{d}{\varepsilon} \log \frac{1}{\varepsilon})$  [3] and that this is asymptotically optimal without further restrictions [16]. It is also known that set systems associated with the guarding of simple polygons with point guards have constant VC-dimension [14, 21], and this bound also applies *a fortiori* to perimeter guards and vertex guards. Thus when guarding simple polygons we can construct  $\varepsilon$ -nets of size  $O(\frac{1}{\varepsilon} \log \frac{1}{\varepsilon})$  using general techniques. In a polygon with  $h$  holes the VC-dimension is  $O(\log h)$  [21] and therefore  $\varepsilon$ -nets of size  $O(\frac{1}{\varepsilon} \log \frac{1}{\varepsilon} \log h)$  can be constructed.

Using techniques specific to vertex guarding or perimeter guarding a simple polygon, we are able to break through the general  $\Theta(\frac{d}{\varepsilon} \log \frac{1}{\varepsilon})$  lower bound to build smaller  $\varepsilon$ -nets. This result is stated in the following theorem.

**Theorem 1** *For the problem of guarding a simple polygon with vertex guards or perimeter guards, we can build  $\varepsilon$ -nets of size  $O(\frac{1}{\varepsilon} \log \log \frac{1}{\varepsilon})$  in polynomial time.*

*Proof* In Sect. 3 we introduce the basic ideas that allow the construction of  $\varepsilon$ -nets of size  $O(1/\varepsilon^2)$ . In Sect. 4 we give a more complicated, hierarchical technique that lets us construct  $\varepsilon$ -nets of size  $O(\frac{1}{\varepsilon} \log \log \frac{1}{\varepsilon})$ .  $\square$

A similar result for a different problem was recently obtained by Aronov et al. [2], who proved the existence of  $\varepsilon$ -nets of size  $O(\frac{1}{\varepsilon} \log \log \frac{1}{\varepsilon})$  when  $\mathcal{S}$  is either a set of axis-parallel rectangles in  $\mathbb{R}^2$  or axis-parallel boxes in  $\mathbb{R}^3$ .

#### 1.2.4 Approximating HITTING SET with $\varepsilon$ -Nets

Brönnimann and Goodrich [5] introduced an algorithm for using a *net finder* (an algorithm for finding  $\varepsilon$ -nets) to find approximately optimal solutions for the HITTING SET problem. Their algorithm gives an initial weighting to the elements in  $\mathcal{U}$ . The net finder is then used to find an  $\varepsilon$ -net for  $\varepsilon = 1/2c'$ , with  $c'$  fixed at a constant between 1 and  $2 \cdot \text{OPT}$ . If there is a set in  $\mathcal{S}$  not hit by the  $\varepsilon$ -net, the algorithm picks such a set and doubles the weight of every element in it. It then repeats, finding a new  $\varepsilon$ -net given the new weighting. This continues until the algorithm finds an  $\varepsilon$ -net that hits every set in  $\mathcal{S}$ . If the net finder constructs  $\varepsilon$ -nets of size  $f(1/\varepsilon)$ , their main algorithm finds a hitting set of size  $f(4 \cdot \text{OPT})$ .

Previous approximation algorithms achieving guaranteed approximation factors of  $\Theta(\log \text{OPT})$  [7, 8] have used this technique, along with generic  $\varepsilon$ -net finders returning  $\varepsilon$ -nets of size  $O(\frac{1}{\varepsilon} \log \frac{1}{\varepsilon})$  for set systems of constant VC-dimension. Instead, we use our net finder from Theorem 1 to obtain the following corollary, whose proof is given in Sect. 2.

**Corollary 1** *Let  $P$  be a simple polygon with  $n$  vertices and let  $G$  be a finite set of potential guard locations such that  $V(P) \subseteq G \subset \partial P$ . Let  $T \subseteq P$  be the set of points we want to guard. There is an algorithm, running in time polynomial in  $n$  and  $|G|$ , that outputs a guarding set for  $T$  of size  $O(\text{OPT} \cdot \log \log \text{OPT})$ , where  $\text{OPT}$  is the size of the minimum subset of  $G$  that guards  $T$ .*

### 1.2.5 Discretization

The number of iterations performed by the main algorithm (see Sect. 2.1.1) is bounded by a function of the number of guard locations. Therefore if the set  $G$  of potential guard locations is infinite, e.g., if  $G = \partial P$ , we must perform discretization to find a finite set  $G' \subset G$  of potential guard locations to consider. We assume that the original set  $G$  consists of a polynomial number of closed intervals of  $\partial P$ .

We use the discretization technique of Deshpande et al. [7], designed for the more general point guarding problem. For our restricted application, this algorithm breaks  $G$  into a number of closed line segments having the property that, if a point  $x$  is seen by a point  $g$  on a segment,  $x$  must be seen by at least one of the endpoints of the segment containing  $g$ . The number of segments required is polynomial in  $n$  and  $\Delta$ , where  $\Delta$  is the ratio between the longest and shortest distances between vertices. We use the set of segment endpoints as the finite guarding set  $G'$ . It is not difficult to see that, if  $G$  contains a hitting set of size  $k$ ,  $G'$  contains a hitting set of size at most  $2k$ .

## 2 The Main Algorithm

Our main algorithm is an application of that presented by Brönnimann and Goodrich [5]. Their algorithm provides a generic way to turn a *net finder*, i.e., an algorithm for finding  $\varepsilon$ -nets for an instance of HITTING SET, into an approximation algorithm. Along with a net finder we also need a *verifier*, which either states correctly that a set  $H$  is a hitting set, or returns a set from  $S$  that is not hit by  $H$ .

For now we assume the existence of an appropriate net finder and verifier, each running in polynomial time. We describe the simple verifier in Sect. 2.1.2; we describe the net finder, our main contribution, in Sects. 3 and 4.

### 2.1 Main Algorithm

$G$  is the finite set of potential guard locations and  $T$  is the (possibly infinite) set of points that must be guarded. We first assign a weight function  $w$  to the set  $G$ . When the algorithm starts each element of  $G$  has weight 1. The main idea of the algorithm is to repeatedly find an  $\varepsilon$ -net  $H$  and, if  $H$  is not a hitting set (i.e., if it does not see everything in  $T$ ), to choose a point  $p \in T$  that is not seen by  $H$  and double the weight of any guard that sees  $p$ .

#### 2.1.1 Bounding the Number of Iterations

For now assume we know the value of  $\text{OPT}$  and we set  $\varepsilon = \frac{1}{2 \cdot \text{OPT}}$ . We give an upper bound for the number of doubling iterations the algorithm can perform. Each iteration increases the total weight of  $G$  by no more than a multiplicative factor of  $(1 + \varepsilon)$  (since the guards whose weight we double have at most an  $\varepsilon$  proportion of the total weight). Therefore after  $k$  iterations the weight has increased to at most

$$|G| \cdot (1 + \varepsilon)^k \leq |G| \cdot \exp\left(\frac{k}{2 \cdot \text{OPT}}\right) \leq |G| \cdot 2^{\left(\frac{3k}{4 \cdot \text{OPT}}\right)}.$$

Let  $\mathcal{H} \subseteq G$  be an optimal hitting set (i.e., guarding set) of size  $\text{OPT}$ . For an element  $h \in \mathcal{H}$  define  $z_h$  as the number of times the weight of  $h$  has been doubled. Since  $\mathcal{H}$  is a hitting set, in each iteration some guard in  $\mathcal{H}$  has its weight doubled, so we have

$$\sum_{h \in \mathcal{H}} z_h \geq k$$

and

$$\begin{aligned} w(\mathcal{H}) &= \sum_{h \in \mathcal{H}} 2^{z_h} \\ &\geq \text{OPT} \cdot 2^{\left(\frac{k}{\text{OPT}}\right)} \quad (\text{since } 2^x \text{ is a convex function}). \end{aligned}$$

We now have

$$\text{OPT} \cdot 2^{\left(\frac{k}{\text{OPT}}\right)} \leq w(\mathcal{H}) \leq w(G) \leq |G| \cdot 2^{\left(\frac{3k}{4\text{OPT}}\right)},$$

which gives us

$$k \leq 4 \cdot \text{OPT} \cdot \log\left(\frac{|G|}{\text{OPT}}\right).$$

This bound also tells us that the total weight  $w(G)$  never exceeds  $\frac{|G|^4}{\text{OPT}^3}$ .

We must now address the fact that the value of  $\text{OPT}$  is unknown. We maintain a variable  $c'$  which is our guess at the value of  $\text{OPT}$ , starting with  $c' = 1$ . If the algorithm runs for more than  $4 \cdot c' \cdot \log\left(\frac{|G|}{c'}\right)$  iterations without obtaining a guarding set, this implies that there is no guarding set of size  $c'$  so we double our guess. When our algorithm eventually obtains a hitting set, we have  $\text{OPT} \leq c' \leq 2 \cdot \text{OPT}$ . The hitting set obtained is a  $\left(\frac{1}{2c'}\right)$ -net built by our net finder. Therefore, using the method from Sect. 4 to build an  $\varepsilon$ -net of size  $O\left(\frac{1}{\varepsilon} \log \log \frac{1}{\varepsilon}\right)$ , we obtain a guarding set of size  $O(\text{OPT} \cdot \log \log \text{OPT})$ .

### 2.1.2 Verification

The main algorithm requires a verification oracle that, given a set  $H$  of guards, either states correctly that  $H$  guards  $T$  or returns a point  $p \in T$  that is not seen by  $H$ . We can use the techniques of Bose et al. [4] to find the visibility polygon of any guard in  $H$  efficiently. It will always be the case that  $|H| < n$ . Finding the union of visibility polygons of guards in  $H$  can be done in polynomial time, as can comparing this union with  $T$ .

## 3 Building Quadratic Nets

In this section we show how to build an  $\varepsilon$ -net using  $O(1/\varepsilon^2)$  guards. This result is not directly useful to us but we use this section to perform the geometric leg work, and hopefully provide some intuition, without worrying about the hierarchical decomposition to be described in Sect. 4. It should be clear that these  $\varepsilon$ -nets can be constructed in polynomial time.

### 3.1 Subdividing the Perimeter

For the construction both of the  $\varepsilon$ -nets in this section and those in the next section we will subdivide the perimeter into a number of *fragments*. Fragment endpoints will always lie on potential guard locations, but the weight of a guard location may be split between multiple fragments and a fragment may consist of a single guard location.

The key difference between the construction of the  $\varepsilon$ -nets in this section and those in the next section is the method of fragmentation. In this section, the perimeter will simply be divided into  $m = 4/\varepsilon$  fragments each having weight  $\frac{\varepsilon}{4}w(G)$ . For our purposes,  $1/\varepsilon$  will always be an integer so  $m$  will always be an integer.

### 3.2 Placing Extremal Guards

For two fragments  $A_i$  and  $A_j$  we will place guards at *extreme points of visibility*. Those are the first and last points on  $A_i$  seen from  $A_j$  and the first and last points on  $A_j$  seen from  $A_i$ . For a contiguous fragment we define the first (resp. last) point of the segment according to the natural clockwise ordering on the perimeter. We use  $G(A_i, A_j)$  to denote the set of up to four extremal guards placed between  $A_i$  and  $A_j$  (see Fig. 1).

These extreme points of visibility might not lie on potential guard locations. In fact, it is entirely possible that two fragments  $A_i$  and  $A_j$  see each other even if no guard location of  $A_i$  sees  $A_j$  and vice versa. If an extreme point of visibility is not a potential guard location, we will simply not place a guard there. Our proofs, in particular the proof of Lemma 2, will only require guards on extreme points of visibility that either lie on vertices or on fragment endpoints.

### 3.3 All Pairs Extremal Guarding

Our aim in this section is to build an  $\varepsilon$ -net by placing extremal guards for every pair  $(A_i, A_j)$  of fragments. We denote this set of guards with

$$S_{\text{AP}} = \bigcup_{i \neq j} G(A_i, A_j).$$

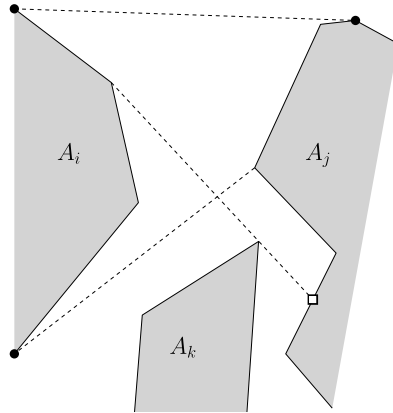
Note that  $|S_{\text{AP}}| \leq 4 \binom{m}{2} = O(1/\varepsilon^2)$ . Also note that every fragment endpoint is included in  $S_{\text{AP}}$ , since an endpoint is an extreme point of visibility of its two incident fragments.

**Lemma 1** *Any point not guarded by  $S_{\text{AP}}$  sees at most four fragments.*

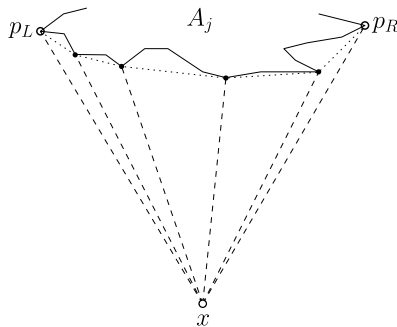
**Corollary 2**  *$S_{\text{AP}}$  is an  $\varepsilon$ -net of size  $O(1/\varepsilon^2)$ .*

For the proof of Lemma 1 we need to present additional properties of the fragments that can be seen by a point. For a point  $x$ , the fragments seen by  $x$  are ordered clockwise in the order they appear on the boundary of  $P$ . We need to consider lines of sight from  $x$ , and what happens when a transition is made from seeing one fragment  $A_i$  to seeing the next fragment  $A_j$ . There are three possibilities:

**Fig. 1** Placing guards at extreme points of visibility. The vertices marked with black disks belong to the set  $G(A_i, A_j)$ . The non-vertex point marked with a square also belongs to  $G(A_i, A_j)$  if and only if it is a fragment endpoint



**Fig. 2** In the proof of Lemma 2,  $x$  sees every point on the geodesic from  $p_L$  to  $p_R$ . This can be seen by a gift wrapping argument: starting with a rubber band stretched from  $p_L$  to  $x$ , keep one end at  $p_L$  and slide the other end in a straight line from  $x$  to  $p_R$ . The rubber band now lies along the geodesic, and no point on it can have ever been obscured from  $x$



1.  $j = i + 1$  and  $x$  sees the guard at the common endpoint of  $A_i$  and  $A_j$
2.  $A_j$  occludes  $A_i$ , in which case we say that  $x$  has a *left tangent* to  $A_j$  (see Fig. 3)
3.  $A_i$  was occluding  $A_j$ , in which case we say that  $x$  has a *right tangent* to  $A_i$  (see Fig. 4).

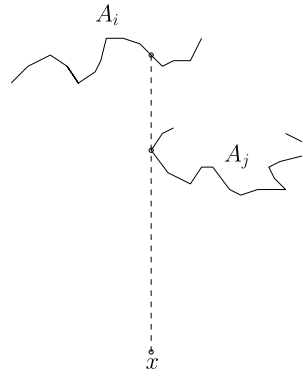
We say a fragment  $A$  *owns* a point  $x$  if  $x$  sees  $A$  in a sector of size at least  $\pi$  (see Fig. 5). We assume any point  $x$  is owned by at most one fragment; if  $x$  is a fragment endpoint it will itself be a guard, and otherwise if  $x$  is owned by two fragments then only those two fragments can see it.

**Lemma 2** *Let  $A_i, A_j, A_k$  be fragments that are seen by  $x$  consecutively in clockwise order. If  $x$  has a left tangent to  $A_j$ , and the combined angle of  $A_j$  and  $A_k$  at  $x$  is no more than  $\pi$ , then  $x$  sees a guard in  $G(A_j, A_k)$ . Symmetrically, if  $x$  has a right tangent to  $A_j$ , and the combined angle of  $A_i$  and  $A_j$  at  $x$  is no more than  $\pi$ , then  $x$  sees a guard in  $G(A_i, A_j)$ .*

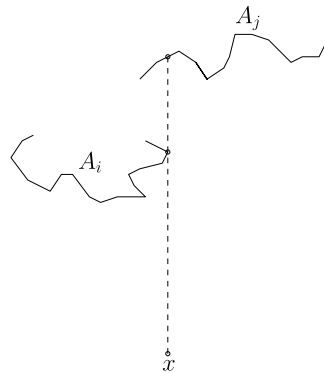
*Proof* We can assume w.l.o.g. that  $x$  has a left tangent to  $A_j$  since the proof of the other case is symmetric. There are now two cases we have to deal with, depending on whether  $x$  has a right tangent to  $A_j$  (case 1) or a left tangent to  $A_k$  (case 2). Define  $p_L$  and  $p_R$ , respectively, as the first and last points on  $A_j$  seen by  $x$ . Observe (see Fig. 2) that  $x$  must see every vertex on the geodesic (shortest path inside  $P$ ) between



**Fig. 3** The point  $x$  has a left tangent to  $A_j$



**Fig. 4** The point  $x$  has a right tangent to  $A_i$



$p_L$  and  $p_R$ . Let  $q$  be the first point on  $A_j$  seen from  $A_k$ . In both cases 1 and 2 (see Figs. 6 and 7),  $q$  must be a vertex of the geodesic between  $p_L$  and  $p_R$ . This can be shown by contradiction; if  $q$  lies between consecutive vertices of this geodesic then those two consecutive vertices must also be seen from  $A_k$ , and one of them comes before  $q$ .

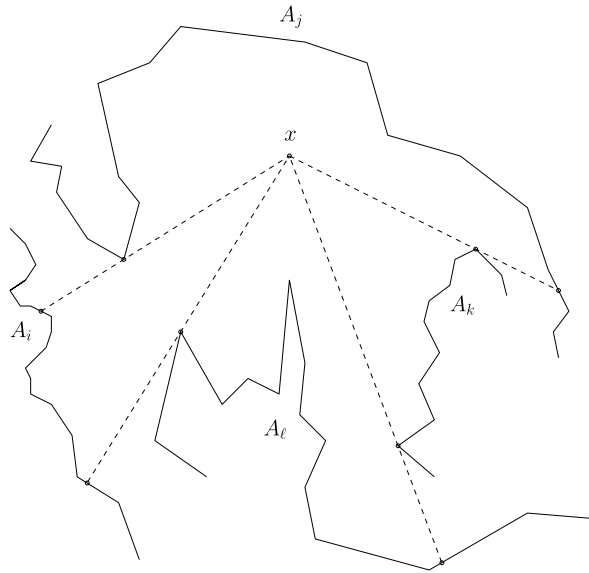
The restriction that the combined angle of  $A_j$  and  $A_k$  at  $x$  is no more than  $\pi$  is necessary to ensure that the geodesic of interest from  $A_k$  to  $A_j$  does not ‘pass behind’  $x$  to see a point on  $A_j$  before  $p_L$ .

It should be emphasized that, since there is a left tangent to  $A_j$ ,  $p_L$  will always be a vertex. Also, if  $p_R$  is not a vertex it will not be the first point on  $A_j$  seen from  $A_k$ .  $\square$

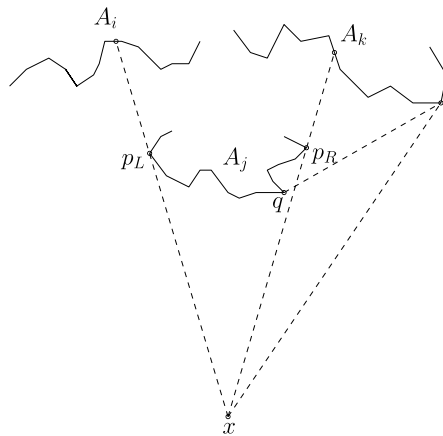
The proof of Lemma 1 is now fairly straightforward.

*Proof of Lemma 1* Let  $x$  be a point that sees at least five fragments. Assume  $x$  is not a fragment endpoint, otherwise it is itself a guard in  $S_{AP}$ . If we have a directed graph whose underlying undirected graph is a cycle, then either we have a directed cycle or we have a vertex with in-degree 2. By the same principle, either some fragment seen by  $x$  has no tangent from  $x$ , or every fragment seen by  $x$  has a left tangent from  $x$  (or every one has a right tangent, which can be handled symmetrically).

**Fig. 5** The point  $x$  has no tangent to  $A_i$ , a left tangent to  $A_j$ , both a left and right tangent to  $A_k$ , and a right tangent to  $A_\ell$ .  $A_j$  owns  $x$



**Fig. 6** Case 1 in the proof of Lemma 2. The point  $x$  has a left tangent and a right tangent to  $A_j$



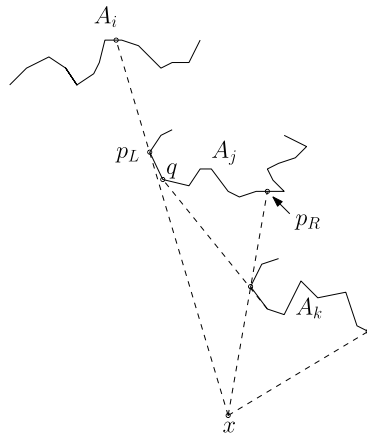
If a fragment seen by  $x$  has no tangent from  $x$ , call such a fragment  $A_0$  and let  $A_{-2}, A_{-1}, A_0, A_1, A_2$  be fragments seen by  $x$  in clockwise order. If the combined angle at  $x$  of  $A_{-2}$  and  $A_{-1}$  is more than  $\pi$ , the combined angle of  $A_1$  and  $A_2$  is less than  $\pi$ . So we can apply Lemma 2 with one of the two pairs of fragments to show that  $x$  is seen by a guard.

If every fragment seen by  $x$  has a left tangent from  $x$ , then we can apply Lemma 2 using two consecutive fragments with a combined angle at  $x$  of less than  $\pi$ .  $\square$

Before we move on we will prove one more helpful lemma.

**Lemma 3** *The number of fragments seen by an unguarded point  $x$  that do not have a tangent from  $x$  is at most 1.*

**Fig. 7** Case 2 in the proof of Lemma 2. The point  $x$  has a left tangent to  $A_j$  and a left tangent to  $A_k$



*Proof* Assume the contrary and let  $A_0$  and  $A_i$  be two such fragments (assume  $A_i \neq A_1$ , otherwise a contradiction could be reached immediately). If one such fragment owns  $x$ , assume it is  $A_0$  and call the next two fragments seen by  $x$  in the clockwise direction  $A_1$  and  $A_2$ , respectively. By Lemma 2,  $x$  is seen by a guard in  $G(A_1, A_2)$  so we reach a contradiction. If no such fragment owns  $x$  then assume w.l.o.g. that, over the fragments seen by  $x$  between  $A_0$  and  $A_i$  going clockwise, the combined angle at  $x$  is less than  $\pi$  (if this is not true it must be true going counter-clockwise). Again,  $x$  is seen by a guard in  $G(A_1, A_2)$  so we reach a contradiction. □

### 4 Hierarchical Fragmentation

In the last section we showed how a quadratic number of guards (i.e.,  $O(1/\varepsilon^2)$ ) could be placed to ensure that any unguarded point sees at most four fragments. In this section we discuss how hierarchical fragmentation can be used to reduce the number of guards required to  $O(\frac{1}{\varepsilon} \log \log \frac{1}{\varepsilon})$ . We will use  $S_{HF}$  to denote the guarding set constructed in this section. It should be clear that these  $\varepsilon$ -nets can be constructed in polynomial time.

We can consider the hierarchy as represented by a tree. At the root there is a single fragment representing the entire perimeter of the polygon. This root fragment is broken up into a certain number of child fragments. Fragmentation continues recursively until a specified depth  $t$  is reached. We will set  $t = \lceil \log \log \frac{1}{\varepsilon} \rceil$ . The *fragmentation factor* (equivalently, the branching factor of the corresponding tree) is not constant, but rather depends on both  $t$  and the level in the hierarchy. The fragmentation factor generally decreases as the level of the tree increases. Specifically, if  $b_i$  is the fragmentation factor at the  $i$ th step, we have

$$b_i = \begin{cases} 2^{2^{t-1}+1} \cdot 4t \cdot 2^{1-t} \cdot \alpha, & i = 1, \\ 2^{2^{t-i}+1}, & 1 < i \leq t, \end{cases}$$

where  $\alpha \leq 1$  is a term introduced only to deal with an issue arising from ceilings and

double exponentials, namely the fact that  $2^{\lceil \log \log 1/\varepsilon \rceil}$  is not in  $O(1/\varepsilon)$ .  $\alpha$  is specified in (3) later in this section.

If  $f_i$  is the total number of fragments after the  $i$ th fragmentation step, this gives us

$$f_i = \begin{cases} 1, & i = 0, \\ 4t \cdot 2^{2^t - 2^{t-i} - t + i + 1} \cdot \alpha, & 0 < i \leq t, \\ 4t \cdot 2^{2^t} \cdot \alpha, & i = t, \end{cases}$$

since

$$\begin{aligned} f_i &= \prod_{j=1}^i b_j \\ &= 4t \cdot 2^{1-t} \cdot \alpha \cdot \prod_{j=1}^i 2^{2^{t-j} + 1} \\ &= 4t \cdot 2^{1-t + \sum_{j=1}^i (2^{t-j} + 1)} \cdot \alpha \\ &= 4t \cdot 2^{2^t - 2^{t-i} - t + i + 1} \cdot \alpha. \end{aligned}$$

Our algorithm will place guards at all pairs of *sibling fragments*, i.e., fragments having the same parent fragment. Whereas before we placed guards at the points in  $G(A_i, A_j)$  for any pair of fragments  $(A_i, A_j)$ , we now only do so if  $A_i$  and  $A_j$  are siblings. For the purposes of this guard placement, the complement of the parent fragment, i.e., the subset of  $G$  outside the parent fragment, will be considered a dummy child fragment. That is, it will be considered a child fragment when placing guards, but not when counting the number of child fragments seen from some point  $x$  as in the statement of Corollary 3 or in the proof of Lemma 4. To denote the complement of a fragment  $A$  we use  $\bar{A}$ . Considering  $\bar{A}$  to be a child of  $A$  when placing guards allows us to consider the children of  $A$  as if they were fragments with guards placed for all pairs. For example, we can obtain the following corollary from Lemmas 1 and 3.

**Corollary 3** *For an unguarded point  $x$  and a fragment  $A$ , the number of child fragments of  $A$  seen by  $x$  is at most 3, and at most one of these child fragments does not have a tangent from  $x$ .*

The total number of guards placed will be

$$|S_{HF}| \leq 4 \sum_{i=1}^t \binom{b_i + 1}{2} f_{i-1} \leq 4 \sum_{i=1}^t b_i^2 f_{i-1}.$$

If  $t \geq 6$  we have  $b_i \leq 2^{2^{t-i} + 1}$  for all values of  $i$ . This gives us

$$\begin{aligned} |S_{HF}| &\leq 4\alpha \sum_{i=1}^t 2^{2(2^{t-i} + 1)} \cdot 4t \cdot 2^{2^t - 2^{t-i} + 1 - t + i} \\ &= 16t\alpha \sum_{i=1}^t 2^{2^t - t + i + 2} \end{aligned}$$

$$\begin{aligned}
 &= 16t\alpha \cdot 2^{2^t-t+3} (2^t - 1) \\
 &< 16t\alpha \cdot 2^{2^t+3} \\
 &= 128t\alpha \cdot 2^{2^t}.
 \end{aligned}$$

Recall that  $t = \lceil \log \log \frac{1}{\varepsilon} \rceil$ . We need to define  $\alpha$  in a way that ensures  $b_1$  is an integer. In addition, as required by our upcoming proof of Lemma 4 and Corollary 4, the following two equations must hold:

$$|S_{HF}| = O\left(\frac{1}{\varepsilon} \log \log \frac{1}{\varepsilon}\right), \tag{1}$$

$$\frac{f_t}{4t} \geq \frac{1}{\varepsilon}. \tag{2}$$

To satisfy these three criteria, it suffices to set

$$\alpha = \frac{\lceil 2^{2^{t-1}+1} \cdot 4t \cdot 2^{-t} \cdot 2^{\log(1/\varepsilon)-2^t} \rceil}{2^{2^{t-1}+1} \cdot 4t \cdot 2^{-t}} = \frac{\lceil 4t \cdot 2^{\log(1/\varepsilon)+1-t-2^{t-1}} \rceil}{4t \cdot 2^{2^{t-1}+1-t}}. \tag{3}$$

We must now provide a generalization of Lemma 1 that works with our hierarchical fragmentation.

**Lemma 4** *Any point not guarded by  $S_{HF}$  sees at most  $4i$  fragments at level  $i$ .*

Applying this with  $i = t$  and using (1) and (2), we get

**Corollary 4**  *$S_{HF}$  is an  $\varepsilon$ -net of size  $O(\frac{1}{\varepsilon} \log \log \frac{1}{\varepsilon})$ .*

*Proof of Lemma 4* Let  $x$  be a point that does not see any guard in  $S_{HF}$ . From the tree associated with the hierarchical fragmentation, we consider the subtree of fragments that see  $x$ . We define a *branching fragment* as a fragment with multiple children seen by  $x$  and we claim that at any level there are at most two branching fragments. We call a non-root fragment *fruitful* if it or one of its descendants is a branching fragment. Corollary 3 tells us that any fragment has at most three children seen by  $x$ . At level 1 there are at most four fragments seen by  $x$ , so it follows that the number of fragments seen by  $x$  at level  $i$  is at most  $4i$ . We must now prove our claim that there are at most two branching fragments at any level.

First we note that a branching fragment either has no tangent from  $x$  or owns  $x$ . To see this, consider a fragment  $A$  that has a tangent from  $x$  and does not own  $x$ . Assume w.l.o.g. that  $x$  has a left tangent to  $A$  and call the point of tangency  $p_L$ .  $x$  must then also have a left tangent to the child fragment  $A_0$  of  $A$  that contains  $p_L$ .  $A_0$  must be the leftmost child fragment of  $A$  seen by  $x$ . If  $x$  sees another child fragment  $A_1$  of  $A$  to the right of  $A_0$ , then by Lemma 2 it is seen by a guard in  $G(A_0, A_1)$ .

For our casewise analysis, we now define six types of fragments with regard to  $x$ ; these types categorize all non-root fragments. The type of a fragment  $A$  depends on

**Table 1** Fragment types with regard to an unguarded point  $x$

Fragment type	Seen by $x$	Owned by $x$	Number of tangents from $x$
I	No	No	0
II	Yes	No	1 or 2
III	Yes	No	0
IV	Yes	Yes	0
V	Yes	Yes	2
VI	Yes	Yes	1

**Table 2** For a fragment of a given type, the possible child fragments seen by  $x$

Fragment type	Child fragments seen by $x$
I	None.
II	Exactly one of type (II). None of any other type.
III	At most three of types (II–III), including at most one of type (III). None of any other type.
IV	At most one of types (III–VI). At most three total.
V	At most three of types (II and V), including at most one of type (V). None of any other type.
VI	Case 1: Exactly one of type (VI). None of any other type. Case 2: At most three of types (II, III and V), including at most two of types (III and V). None of any other type.

whether it is seen by  $x$ , whether it is owned by  $x$ , and how many tangents from  $x$  it has. Fragment types are defined in Table 1.

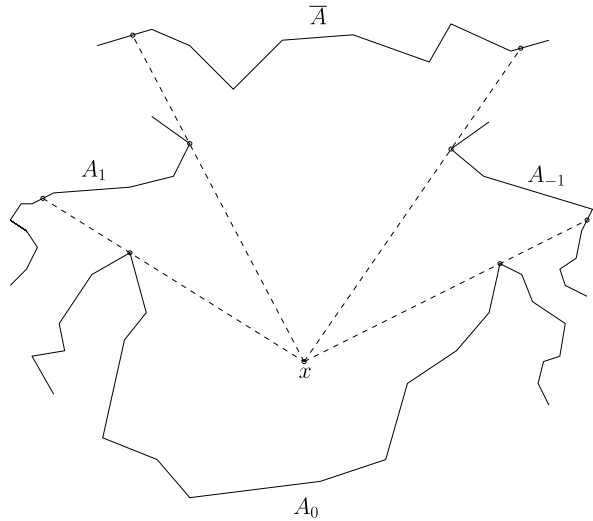
For a fragment seen by  $x$ , we consider the possible child fragments that are also seen by  $x$ . These possibilities, whose proofs we reserve for later, are summarized in Table 2. Assuming for now that the claims in Table 2 are true, we can state several facts:

- Only fragments of type (III–VI) can be fruitful.
- Only fragments of type (VI) can have more than one fruitful child, and they can have at most two fruitful children.
- The parent of a fragment of type (VI) is either the root fragment or a fragment of type (VI) with no other children.
- If the root has a child fragment of type (VI), the root cannot have a child of type (III).

Together, these facts imply that any level has at most two fruitful fragments. We can now state the following:

- Level 1 has at most four child fragments that see  $x$ , at most two of which are fruitful.
- A fruitful fragment has at most three child fragments that see  $x$ , at most one of which is fruitful.
- A non-fruitful fragment has at most one child fragment that sees  $x$ .

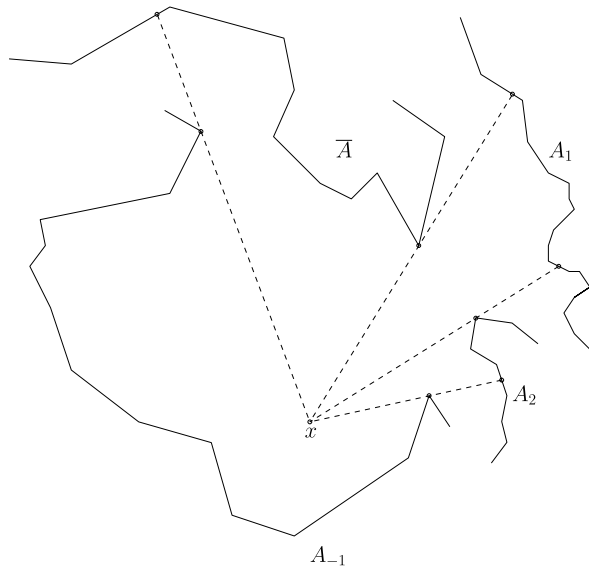
**Fig. 8** The only way a fragment of type (v) can have three children seen by  $x$



Therefore any level has at most two fruitful fragments and the number of fragments at level  $i$  that see  $x$  is at most  $4i$ . It remains to prove the claims in Table 2. We do so now, row by row.

- (I) Trivial.
- (II) Follows from Lemma 2.
- (III) By Corollary 3,  $x$  can see at most three child fragments of  $A$ , at most one of which is of type (III). Any other fragments seen by  $x$  must be of type (II).
- (IV) If a child of  $A$  owns  $x$  it must be the only child of  $A$  that sees  $x$ , and this child is also of type (IV). Otherwise,  $A$  would have a child fragment  $A_i$  that is seen by  $x$ , does not own  $x$ , and is adjacent to  $\bar{A}$ .  $x$  would then be seen by a guard in  $G(A_i, \bar{A})$ . Any other fragments seen by  $x$  must be of type (II).
- (V) See Fig. 8. Because  $\bar{A}$  is, in a sense, a ‘dummy’ child of type (III),  $A$  cannot have a real child of type (III) by the proof of Lemma 3. Further, if  $A$  has a child  $A_0$  that owns  $x$ , this child must also be of type (v). Otherwise assume w.l.o.g. that  $A_1$ , immediately clockwise from  $A_0$ , has a left tangent from  $x$ . Then, using  $A_2$  to denote the fragment clockwise from  $A_1$  ( $A_2$  might be  $\bar{A}$ ),  $x$  is seen by  $G(A_1, A_2)$ .
- (VI) See Fig. 9. Case 1 is the trivial case in which  $x$  sees only one child fragment. We now consider case 2 in which  $x$  sees more than one child fragment. Assume w.l.o.g. that  $\bar{A}$  has a right tangent. If  $A_{-1}$  is the child of  $A$  seen by  $x$  immediately counterclockwise from  $\bar{A}$  then  $\bar{A}$  and  $A_{-1}$  together must span an angle of over  $\pi$  at  $x$ , otherwise  $x$  is seen by  $G(A_{-1}, \bar{A})$ . This implies that no child other than  $A_{-1}$  can own  $x$ .  
First we consider the sub-case in which  $A_{-1}$  owns  $x$ . If  $A_1$  is the child of  $A$  seen by  $x$  immediately clockwise from  $\bar{A}$  then  $A_1$  cannot have a tangent from  $x$  otherwise  $x$  would be seen by  $G(\bar{A}, A_1)$ . If  $x$  can see  $A_2$ , a child of  $A$  between  $A_1$  and  $A_{-1}$ , then  $x$  must have two tangents to  $A_{-1}$  otherwise it would be seen by  $G(A_1, A_2)$ .

**Fig. 9** The only way a fragment of type (VI) can have three children seen by  $x$



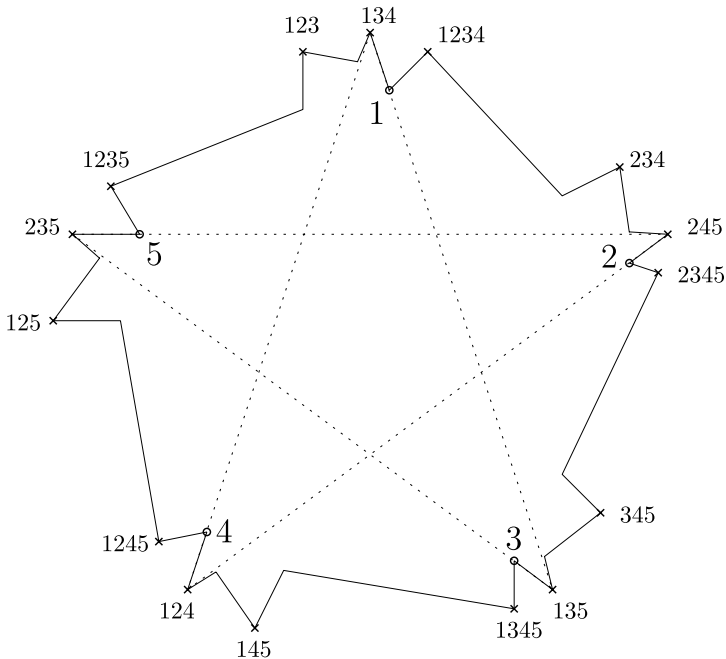
Now we consider the sub-case in which no child of  $A$  owns  $x$ . By Corollary 3, along with  $\bar{A}$  and  $A_{-1}$ ,  $x$  can only be seen by two other children of  $A$ ; again call them  $A_1$  and  $A_2$ . Since  $A_1$  and  $A_2$  are rotationally consecutive around  $x$ ,  $x$  must have a tangent to at least one of them (or be seen by a guard at their common endpoint).

Therefore, if  $A$  has more than one child seen by  $x$ , at most two are of types (III) and (v), and if there is a third it must be of type (II). □

### 5 Open Problems

- We have obtained a  $o(\log \text{OPT})$ -approximation factor for vertex guards and perimeter guards. Can the same be done for point guards?
- Can we do better than  $O(\log \log \text{OPT})$  for perimeter guards? In particular, can we find a constant-factor approximation algorithm to match the hardness of approximation result of Eidenbenz [9]?
- For simple polygons, the set systems associated with point guards have maximum VC-dimension at least 6 and at most 23 [21]; it is believed that the true value is closer to the lower end of this range, perhaps even 6 [14]. The upper bound of 23 holds *a fortiori* for set systems associated with perimeter guards but the lower bound of 6 does not. A lower bound of 4 follows from a trivial modification to an example for monotone chains [15]; we can increase this bound to 5 without too much difficulty (see Fig. 10). Can set systems associated with perimeter guards actually have VC-dimension as high as 6? And can the upper bound of 23 be improved? It seems that improving the upper bound would be easier for perimeter guards than for point guards.





**Fig. 10** A polygon with a set  $S$  of five points on the perimeter. The points in  $S = \{1, 2, 3, 4, 5\}$  are marked with *circles* and labeled with large numbers. Each point in  $S$  sees all of  $S$ , and each guard seeing a subset of  $S$  of size 3 or 4 is marked with a *cross* and labeled with small numbers indicating which points in  $S$  it sees. Guards seeing the 16 subsets of  $S$  of size 0, 1, or 2 are not shown. Adding these is a simple matter of adding nooks with very small angles of visibility, thus we can construct a polygon with 5 points on the perimeter shattered by  $2^5$  perimeter guards. Such a polygon can also be obtained via a fairly straightforward modification of the example of Kalai and Matoušek for point guards [14]

## References

1. Aggarwal, A.: The art gallery theorem: its variations, applications and algorithmic aspects. Ph.D. thesis, The Johns Hopkins University, Baltimore, MD (1984)
2. Aronov, B., Ezra, E., Sharir, M.: Small-size  $\varepsilon$ -nets for axis-parallel rectangles and boxes. *SIAM J. Comput.* **39**(7), 3248–3282 (2010)
3. Blumer, A., Ehrenfeucht, A., Haussler, D., Warmuth, M.K.: Learnability and the Vapnik-Chervonenkis dimension. *J. ACM* **36**(4), 929–965 (1989)
4. Bose, P., Lubiw, A., Munro, J.I.: Efficient visibility queries in simple polygons. *Comput. Geom. Theory Appl.* **23**(3), 313–335 (2002)
5. Brönnimann, H., Goodrich, M.T.: Almost optimal set covers in finite VC-dimension. *Discrete Comput. Geom.* **14**(1), 463–479 (1995)
6. Chvátal, V.: A greedy heuristic for the set-covering problem. *Math. Oper. Res.* **4**(3), 233–235 (1979)
7. Deshpande, A., Kim, T., Demaine, E.D., Sarma, S.E.: A pseudopolynomial time  $O(\log n)$ -approximation algorithm for art gallery problems. In: Proc. 10th Workshop on Algorithms and Data Structures, Halifax, Canada, pp. 163–174 (2007)
8. Efrat, A., Har-Peled, S.: Guarding galleries and terrains. *Inf. Process. Lett.* **100**(6), 238–245 (2006)
9. Eidenbenz, S.: Inapproximability results for guarding polygons without holes. In: Proc. 9th Int. Symp. Algorithms and Computation. Lecture Notes in Computer Science, vol. 1533, pp. 427–436 (1998)
10. Eidenbenz, S., Stamm, C., Widmayer, P.: Inapproximability results for guarding polygons and terrains. *Algorithmica* **31**(1), 79–113 (2001)
11. Feige, U.: A threshold of  $\ln n$  for approximating set cover. *J. ACM* **45**(4), 634–652 (1998)

12. Garey, M., Johnson, D.: *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, San Francisco (1979)
13. Ghosh, S.: Approximation algorithms for art gallery problems. In: Proc. Canadian Information Processing Society Congress, pp. 429–436 (1987)
14. Kalai, G., Matoušek, J.: Guarding galleries where every point sees a large area. *Isr. J. Math.* **101**(1), 125–139 (1997)
15. King, J.: VC-dimension of visibility on terrains. In: Proc. 20th Canadian Conference on Computational Geometry, Montreal, Canada, pp. 27–30 (2008)
16. Komlós, J., Pach, J., Woeginger, G.: Almost tight bounds for  $\epsilon$ -nets. *Discrete Comput. Geom.* **7**(1), 163–173 (1992)
17. Lee, D., Lin, A.: Computational complexity of art gallery problems. *IEEE Trans. Inf. Theory* **32**, 276–282 (1986)
18. O'Rourke, J.: *Art Gallery Theorems and Algorithms*. Oxford University Press, London (1987)
19. O'Rourke, J., Supowit, K.J.: Some NP-hard polygon decomposition problems. *IEEE Trans. Inf. Theory* **29**(2), 181–189 (1983)
20. Raz, R., Safra, S.: A sub-constant error-probability low-degree-test and a sub-constant error-probability PCP characterization of NP. In: Proc. 29th ACM Symp. Theory of Computing. El Paso, TX, pp. 475–484 (1997)
21. Valtr, P.: Guarding galleries where no point sees a small area. *Isr. J. Math.* **104**(1), 1–16 (1998)
22. Vapnik, V.N., Chervonenkis, A.Y.: On the uniform convergence of relative frequencies of events to their probabilities. *Theory Probab. Appl.* **16**(2), 264–280 (1971)