Approximate Guarding of Monotone and Rectilinear Polygons

Erik Krohn[*] Bengt J. Nilsson[†]

Abstract

We show that vertex guarding a monotone polygon is NP-hard and construct a constant factor approximation algorithm for interior guarding monotone polygons. Using this algorithm we obtain an approximation algorithm for interior guarding rectilinear polygons that has an approximation factor independent of the number of vertices of the polygon. If the size of the smallest interior guard cover is OPT for a rectilinear polygon, our algorithm produces a guard set of size $O(OPT^2)$. Computational geometry Art gallery problems Monotone polygons Rectilinear polygons Approximation algorithms

1 Introduction

The art gallery problem is perhaps the best known problem in computational geometry. It asks for the minimum number of guards to guard a space having obstacles. Originally, the obstacles were considered to be walls mutually connected to form a closed Jordan curve, hence, a simple polygon. Tight bounds for the number of guards necessary and sufficient were found by Chvátal [7] and Fisk [17]. Subsequently, other obstacle spaces, both more general and more restricted than simple polygons have also been considered for guarding problems, most notably, polygons with holes and simple rectilinear polygons [21, 32].

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

We distinguish between two types of guarding problems in simple polygons. Vertex guarding considers only guards positioned at vertices of the polygon, whereas interior guarding allows the guards to be placed anywhere in the interior of the polygon.

The computational complexity question of guarding simple polygons was settled by Aggarwal [1] and Lee and Lin [26] independently when they showed that the problem is NP-hard for both vertex guards and interior guards. Further results have shown that already for very restricted subclasses of polygons the problem is still NP-hard [2, 30].

Chen *et al.* [5] claim that vertex guarding a monotone polygon is NP-hard, however the details of their proof are omitted and still to be verified. We present a new proof that vertex guarding a monotone polygon is NP-hard.

The approximation complexity of guarding polygons has been studied by Eidenbenz and others. Eidenbenz [14] shows that polygons with holes cannot be efficiently guarded by fewer than $\Omega(\log n)$ times the optimal number of interior or vertex guards, unless P=NP, where n is the number of vertices of the polygon. Brodén *et al.* and Eidenbenz [2, 13] independently prove that interior guarding simple polygons is APX-hard.

[*]Department of Computer Science, University of Wisconsin — Oshkosh, Oshkosh, WI, 54901, USA. email: krohne@uwosh.edu
[†]Department of Computer Science, Malmö University, SE-205 06 Malmö, Sweden. email: bengt.nilsson.TS@mah.se

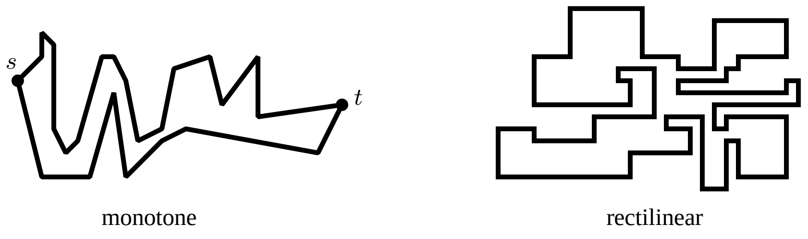


Figure 1: Illustrating the polygon classes.

Any polygon (with or without holes) can be efficiently vertex guarded with logarithmic approximation factor in n, the number of vertices of the polygon. The algorithm is a simple reduction to SET COVER and goes as follows [19]: compute the arrangement produced by the visibility polygons of the vertices. Next, let each vertex v correspond to a set in the set cover instance consisting of elements corresponding to the faces of the arrangement that lie in the visibility polygon of v. The greedy algorithm for SET COVER will then produce a guard cover having logarithmic approximation factor.

The above result can be improved for simple polygons using randomization, giving an algorithm with expected running time $O(n\,OPT_v^2 \log^4 n)$ that produces a vertex guard cover with approximation factor $O(\log OPT_v)$ with high probability, where OPT_v is the smallest vertex guard cover for the polygon [12].

Taking the same approach one step further, Deshpande *et al.* [11] present a pseudo-polynomial randomized algorithm for finding a guard cover (without any restriction on placement) with approximation factor $O(\log OPT)$.

We prove polynomial time deterministic approximation algorithms for interior guarding of monotone and rectilinear polygons. As we have already mentioned, vertex guarding of monotone polygons is NP-hard, and furthermore, optimally guarding rectilinear polygons is also NP-hard [23]. This provides the basis for our interest in approximation algorithms for these problems.

The art gallery problem concerns itself with covering polygons using star shaped pieces, the visibility polygons of the guards. Covering polygons with other type of objects, e.g., convex polygons, etc., remains NP-hard in general; [8, 9, 16, 20, 29, 32, 33, 34].

The next section contains some useful definitions. Section 3 contains our NP-hardness proof for monotone polygons and in Sections 4 and 5 we describe the approximation algorithms for guarding monotone and rectilinear polygons respectively.

2 Definitions

A polygon $\mathbf{P}$ is *l-monotone* if there is a line of monotonicity l such that any line orthogonal to l has a simply connected intersection with $\mathbf{P}$. When we talk about monotone polygons, we will henceforth assume that they are x-monotone, i.e., the x-axis is the line of monotonicity for the polygons we consider; see Figure 1.

The boundary of a monotone polygon $\mathbf{P}$ can be subdivided into two chains, the *upper chain* U and the *lower chain* D. Let s and t be the leftmost and rightmost vertices of $\mathbf{P}$ respectively. The chain U consists of the boundary path followed from s to t in clockwise direction, whereas D is the boundary path followed from s to t in counterclockwise direction.

A polygon $\mathbf{P}$ is *rectilinear* if the boundary of $\mathbf{P}$ consists of axis parallel line segments. Hence, at each vertex, the interior angle between the two connecting boundary edges is either 90 or 270 degrees; see Figure 1.

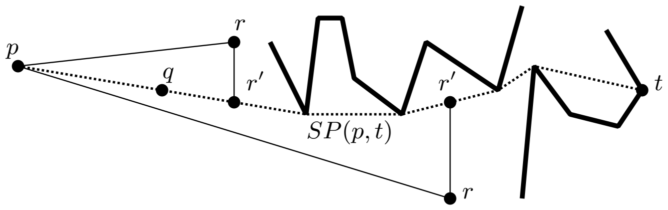


Figure 2: Illustrating the proof of Lemma 2.1.

Let $\mathbf{VP}(p)$ denote the visibility polygon of $\mathbf{P}$ from the point p, i.e, the set of points in $\mathbf{P}$ that can be connected with a line segment to p without intersecting the outside of $\mathbf{P}$.

Consider a partial set of guard points $g_1, \ldots, g_m$ in $\mathbf{P}$ and the union of their visibility polygons $\bigcup_{i=1}^{m} \mathbf{VP}(g_i)$, the set $\mathbf{P} \setminus \bigcup_{i=1}^{m} \mathbf{VP}(g_i)$ is the region of $\mathbf{P}$ not seen by the points $g_1, \ldots, g_m$. This region consists of a set of simply connected polygonal regions called *pockets* bounded by either the polygon boundary or the edges of the visibility polygons.

The following definitions are useful for monotone polygons. Since the x-axis is the line of monotonicity it makes sense to say that an object A in the polygon is to the left or to the right of some other object B if there is vertical line that separates the two objects. We will occasionally use $A \geq B$ ($A \leq B$) to denote that A is to the right (to the left) of B.

Let q be a point in $\mathbf{VP}(p)$. We denote by $\mathbf{VP}_R(p,q)$ the part of $\mathbf{VP}(p)$ that lies to the right of q. Similarly, $\mathbf{VP}_L(p,q)$ is the part of $\mathbf{VP}(p)$ to the left of q. Hence, $\mathbf{VP}(p) = \mathbf{VP}_L(p,q) \cup \mathbf{VP}_R(p,q)$ for all points $q \in \mathbf{P}$. We also denote $\mathbf{VP}_R(p) = \mathbf{VP}_R(p,p)$ and $\mathbf{VP}_L(p) = \mathbf{VP}_L(p,p)$.

In the sequel, we will also let $SP(p,q)$ denote the shortest (Euclidean) path between points p and q inside $\mathbf{P}$.

LEMMA 2.1 *If q is a point on $SP(p,t)$ inside a monotone polygon $\mathbf{P}$, then $\mathbf{VP}_R(p,q) \subseteq \mathbf{VP}_R(q)$.*

PROOF: Let r be a point to the right of q in $\mathbf{P}$ that is visible from p. To prove that r is seen from q consider the vertical line through r and its intersection point r' with $SP(p,t)$. The three points p, r, and r' define a polygon in $\mathbf{P}$ having three convex vertices and possibly some reflex vertices on the path $SP(p,r')$. Since r sees both p and r', r sees all of the path $SP(p,r')$ and hence also the point q; see Figure 2. □ □

3 NP-Hardness of Vertex Guarding Monotone Polygons

In this section, we will show that vertex guarding a monotone polygon is NP-hard. The reduction is from *Monotone 3SAT* (M3SAT) [18, page 259 (problem L02)]. An M3SAT instance $(\mathcal{X}, \mathcal{C})$ consists of a pair of sets, a set of Boolean variables, $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ and a set of clauses, $\mathcal{C} = \{c_1, c_2, ..., c_m\}$. Each clause contains three literals, $c_i = x_j \vee x_k \vee x_l$, a *positive clause*, or $c_i = \bar{x}_j \vee \bar{x}_k \vee \bar{x}_l$, a *negative clause*, for $1 \leq j,k,l \leq n$. An M3SAT instance is satisfiable, if a satisfying truth assignment for $\mathcal{C}$ exists such that all clauses c_i are true.

An ordinary 3SAT instance can easily be transformed to an M3SAT instance by taking each non-monotone clause and replacing it by three monotone ones as follows.

$$\begin{array}{lcl} c_i = x_j \vee x_k \vee \bar{x}_l & \longrightarrow & (\bar{z}_{i1} \vee \bar{z}_{i2} \vee \bar{x}_l) \wedge (z_{i1} \vee x_j \vee x_k) \wedge (z_{i2} \vee x_j \vee x_k) \\ c_i = \bar{x}_j \vee \bar{x}_k \vee x_l & \longrightarrow & (z_{i1} \vee z_{i2} \vee x_l) \wedge (\bar{z}_{i1} \vee \bar{x}_j \vee \bar{x}_k) \wedge (\bar{z}_{i2} \vee \bar{x}_j \vee \bar{x}_k) \end{array}$$

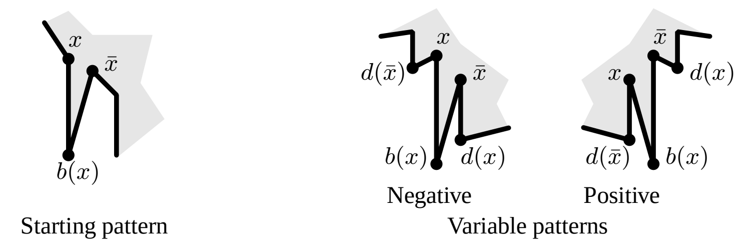


Figure 3: The different types of variable patterns.

where z_{i1} and z_{i2} are new variables used only in these three clauses.

It is easy to verify that a truth assignment makes clause c_i true if and only if the truth assignment makes all the three monotone replacement clauses true as well.

By appropriately duplicating clauses, we can assume that the instance has m clauses where m is odd and the instance has $(m+1)/2$ positive clauses and $(m-1)/2$ negative clauses. Also, let $K = n\,(m+1)$.

We show that any M3SAT instance is polynomially transformable to an instance of vertex guarding a monotone polygon. We construct a monotone polygon $\mathbf{P}$ from the M3SAT instance such that $\mathbf{P}$ is guardable by K or fewer guards if and only if the M3SAT instance is satisfiable. We first present some basic gadgets to show how the polygon is constructed. We then connect these gadgets together to create a polygon.

Starting Pattern: The lower boundary of the polygon is divided into two parts, the left and the right sides. The first gadgets on the left side are the *starting patterns*. The starting patterns are shown to the left in Figure 3. In each pattern, the bottom of the downward spike $b(x)$ is the distinguished vertex of the pattern. This area is only seen by vertices x and $\bar{x}$ and must be guarded by one of these two vertices. This pattern appears along the left side of the lower boundary of the monotone polygon a total of n times, one corresponding to each variable.

Variable Pattern: On the left and the right side of the lower boundary we have *variable patterns* that verify the assigned truth value of each variable. This pattern is shown to the right in Figure 3. Once again, the bottom of the spike at $b(x)$ must be guarded by either x or $\bar{x}$. The pattern has additional distinguished vertices that we call *ledges* $d(x)$ and $d(\bar{x})$ that must both be seen and this is what forces the choice of guard placement at either x or $\bar{x}$.

Figure 4 shows how the starting patterns are connected to variable patterns. If we choose x_j in the starting pattern, we are forced to continuing to choose x_j in each of the subsequent variable patterns. If we at some variable pattern would choose $\bar{x}_j$ instead of x_j, the ledge $d(\bar{x}_j)$ is not seen. Similarly, if we in the starting pattern choose $\bar{x}_j$, we are, by the same argument, forced to continuing to choose $\bar{x}_j$ in each of subsequent variable patterns.

Clauses: For each clause c in the boolean formula, there is a sequence of variable patterns $x_1, \ldots, x_n$ along either the left or the right side of the lower boundary and a clause pattern along the upper boundary of the polygon. On the left side of the lower boundary, the variable pattern sequence corresponds to negative clauses, on the right side, to positive clauses.

The clause pattern on the upper boundary consists of three vertices in an upward spike such that the top vertex of the spike is only seen by the variable patterns corresponding to the literals in the clause; see

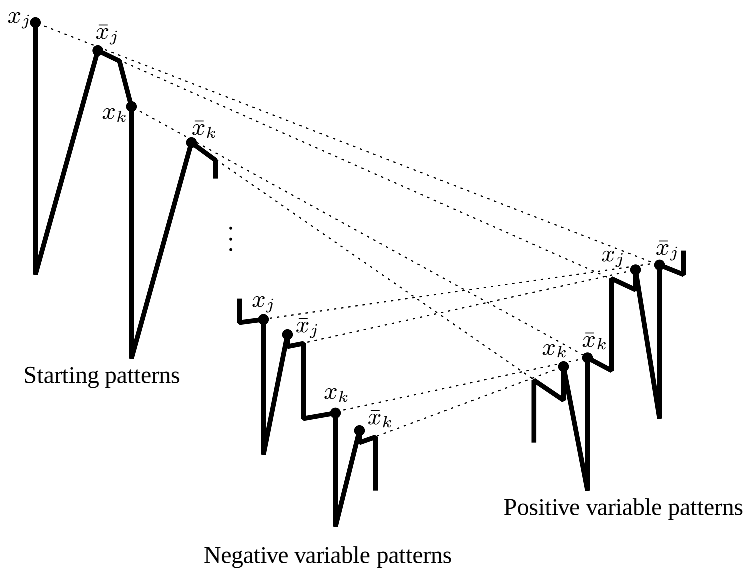


Figure 4: Variable patterns transferring logical values.

Figure 5. We denote the top vertex of the spike by c to correspond to the clause.

We choose our truth value for each variable in the starting variable patterns. The truth values are then mirrored in turn between variable patterns on the right side, corresponding to positive clauses, and variable patterns on the left side, corresponding to negative clauses, of the lower boundary. Truth values do not change in the mirroring process since a variable x_j in clause c_i only sees the ledge $d(x_j)$ in the next variable pattern and none of the other ledges. Similarly $\bar{x}_j$ only sees ledge $d(\bar{x}_j)$ in the next variable pattern; see Figure 4.

In the example of Figure 5 the M3SAT clause corresponds to $c = \bar{x}_1 \vee \bar{x}_3 \vee \bar{x}_5$. Hence, a vertex guard placement that corresponds to a truth assignment that makes c true, will have at least one guard on $\bar{x}_1$, $\bar{x}_3$ or $\bar{x}_5$ and can therefore see vertex c without additional guards.

We still have variables x_2 and x_4 in the clause, however none of them or their negations see the vertex c. They are there simply to transfer their truth values in case these variables are needed in later clauses.

The monotone polygon we construct consists of $4n + (6n + 4)m + 2$ vertices. Each starting variable pattern having four vertices, each variable pattern six vertices, the clause spike consists of three vertices plus one blocking vertex at the start of each clause sequence on the lower boundary and the two leftmost and rightmost points of the polygon.

Consider an M3SAT instance $(x_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_3 \vee \bar{x}_5) \wedge (x_3 \vee x_4 \vee x_5)$. Figure 6 shows how this instance is transformed into a monotone polygon and a placement of guards corresponding to the satisfying truth assignment $x_1 = x_2 = x_4 = x_5 = \mathit{false}$, $x_3 = \mathit{true}$.

Exactly $K = n(m + 1)$ guards are required to guard the polygon since there are K bottom vertices $b(x_j)$ at downward spikes and no vertex in the polygon can see more than one such $b(x_j)$ vertex.

If the M3SAT instance is satisfiable, then we place guards at vertices in accordance to whether the variable is true or false in each of the sequences of variable patterns. Each clause vertex is seen since one of the literals in the associated clause is true and the corresponding vertex has a guard.

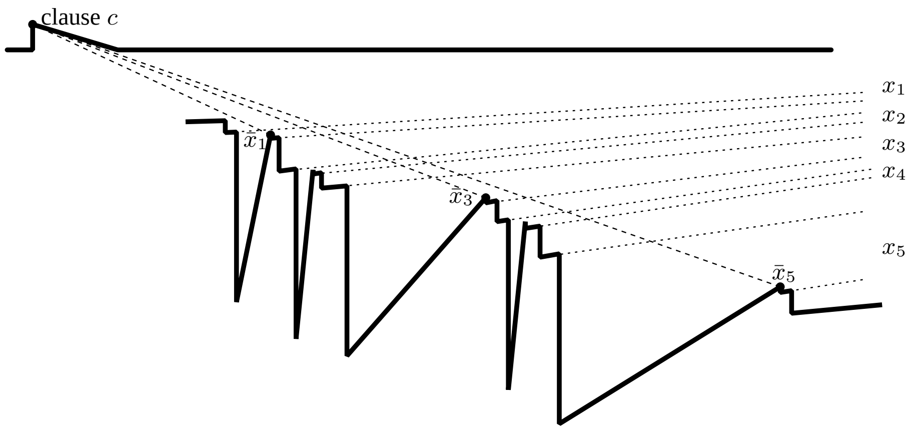


Figure 5: A variable pattern sequence with its clause spike.

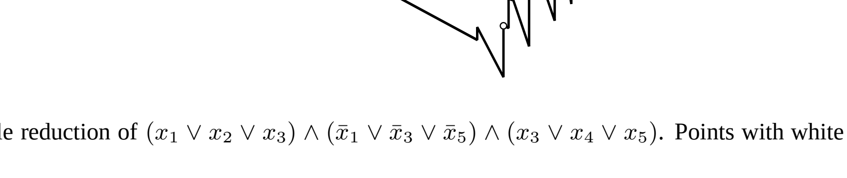


Figure 6: Example reduction of $(x_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_3 \vee \bar{x}_5) \wedge (x_3 \vee x_4 \vee x_5)$. Points with white centers mark the guards.

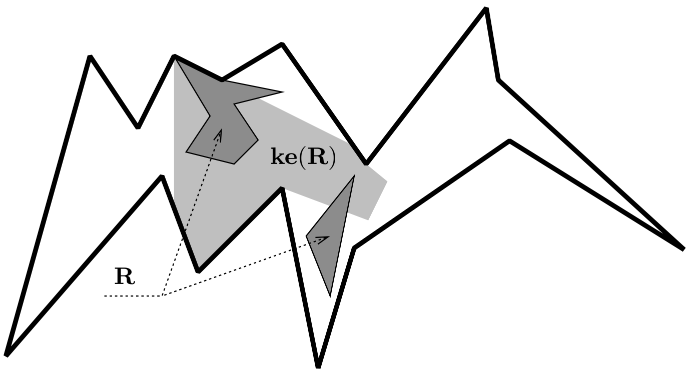


Figure 7: Illustrating the concept of a kernel expansion. The darker shaded areas are the components of $\mathbf{R}$ and the lighter shaded area is the kernel expansion of $\mathbf{R}$.

Suppose we have a vertex guard cover of size exactly K. Since each bottom spike $b(x_j)$ is guarded there is a guard at one of x_j, $\bar{x}_j$, or $b(x_j)$ itself. They together make up K guards so there can be no other guards. Since each clause vertex c_i is also seen, we can establish which of the guards see this vertex and deduce a satisfying truth assignment from this guard placement.

We have proved the following theorem.

THEOREM 1 *Finding the smallest vertex guard cover for a monotone polygon is NP-hard.*

Note that our proof does not immediately generalize to interior guards. In the next section, we show how to approximate the minimum number of interior guards in a monotone polygon.

4 Interior Guarding Monotone Polygons

4.1 The Guarding Algorithm

Our algorithm for guarding a monotone polygon $\mathbf{P}$ incrementally guards $\mathbf{P}$ starting from the left, moving right. Hence, we are interested in the structure of the pockets that occur when guarding is done in this way. We define the main region that we will be interested in, the *spear*, that will guide the placement of our guards.

Let $\mathbf{R}$ be a, possibly disconnected, set of points in $\mathbf{P}$ and let q be a point in $\mathbf{P}$. We denote by $\mathbf{R}_L(q)$ the points of $\mathbf{R}$ to the left of q. Let $v_{\mathbf{R}}$ be a leftmost point of $\mathbf{R}$, hence, if q is to the left of $v_{\mathbf{R}}$, then $\mathbf{R}_L(q) = \emptyset$.

DEFINITION 4.1 *The kernel expansion of $\mathbf{R}$, denoted $\mathbf{ke}(\mathbf{R})$, is formally defined as the set of points*

$$\mathbf{ke}(\mathbf{R}) \overset{def}{=} \{q \geq v_{\mathbf{R}} \mid \mathbf{R}_L(q) \subseteq \mathbf{VP}(q)\},$$

i.e., all the points q in $\mathbf{P}$ to the right of $v_{\mathbf{R}}$ that see everything in $\mathbf{R}_L(q)$; see Figure 7. Figure 7 shows the kernel expansion of a region consisting of two connected components.

In Section 4.3.1 we describe how to compute kernel expansions efficiently.

LEMMA 4.1 *A kernel expansion $\mathbf{ke}(\mathbf{R})$ is a monotone polygon.*

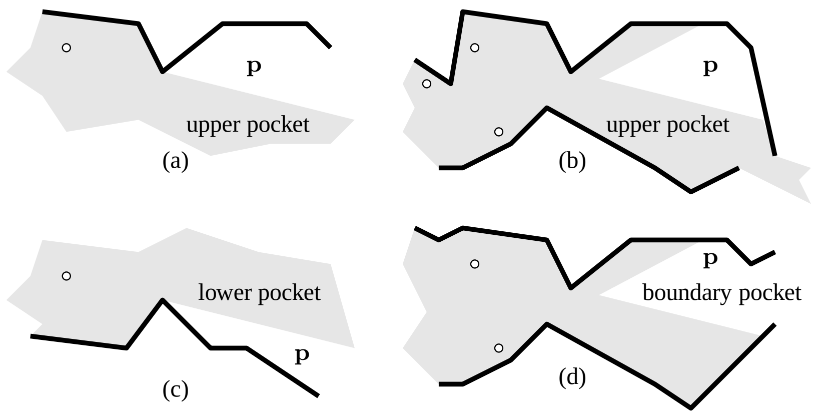


Figure 8: Illustrating pockets. Points with white centers are guards. The shaded areas are the visibility polygons.

PROOF: Let q and q' be two points in $\mathbf{ke}(\mathbf{R})$ having the same x-coordinate, hence, $\mathbf{R}_L(q) = \mathbf{R}_L(q')$ and assume that $\mathbf{R}_L(q) \neq \emptyset$. Let p be any point in $\mathbf{R}_L(q)$ seen by both q and q'. Since the three points p, q, and q' form a (possibly degenerate) triangle in $\mathbf{P}$, any point between q and q' will also see p. This means that any vertical line has a simply connected intersection with $\mathbf{ke}(\mathbf{R})$ so the region is monotone. □ □

Assume that we have a partial guard cover in $\mathbf{P}$ with the property that all guards are to the left of any pockets remaining in $\mathbf{P}$. Consider such a pocket $\mathbf{p}$. We say that $\mathbf{p}$ is a *boundary pocket* if it is adjacent to both the upper and lower boundaries U and D of $\mathbf{P}$, an *upper pocket* if it is adjacent only to the upper boundary U, a *lower pocket* if it is adjacent only to the lower boundary D and a *middle pocket* if it is adjacent to neither U nor D; see Figures 8(a)–(c).

We show in Section 4.2.1 that our incremental guarding algorithm never produces any middle pockets, so we can disregard them for now.

Since we assume that the guards all lie to the left of all pockets, it is easy to see that the cover can only generate one boundary pocket. Let $\mathbf{p}$ be such a boundary pocket. In Lemma 4.7 of Section 4.3.1 we show that the kernel expansion of a polygonal region is completely determined by the vertices of the region. We subdivide the vertices of $\mathbf{p}$ into three sets, $\mathcal{V}^M$, $\mathcal{V}^U$ and $\mathcal{V}^D$, where $\mathcal{V}^M$ are the vertices interior to $\mathbf{P}$, $\mathcal{V}^U$ are the vertices coinciding with the upper boundary U of $\mathbf{P}$ and $\mathcal{V}^D$ are the vertices coinciding with the lower boundary D of $\mathbf{P}$. We let $\mathcal{P}_U(\mathbf{p})$ be the straight line path visiting the vertices in $\mathcal{V}^M \cup \mathcal{V}^U$ in order from left to right, i.e., all vertices of $\mathcal{V}^U$ are visited along the upper boundary U. Similarly, we let $\mathcal{P}_D(\mathbf{p})$ be the straight line path visiting the vertices in $\mathcal{V}^M \cup \mathcal{V}^D$ in order from left to right.

Consider the upper pockets resulting from a partial guard cover and enumerate them $\mathbf{p}_1^U, \mathbf{p}_2^U, \ldots$ from left to right. Similarly enumerate the lower pockets $\mathbf{p}_1^D, \mathbf{p}_2^D, \ldots$ from left to right and denote the boundary pocket $\mathbf{p}^B$. Define two sets as follows:

$$\begin{aligned}
\mathbf{Q}_U &\overset{def}{=} \mathbf{p}_1^U \cup \mathbf{p}_2^U \cup \cdots \cup \mathcal{P}_U(\mathbf{p}^B), \\
\mathbf{Q}_D &\overset{def}{=} \mathbf{p}_1^D \cup \mathbf{p}_2^D \cup \cdots \cup \mathcal{P}_D(\mathbf{p}^B),
\end{aligned}$$

with which we can establish the main regions of interest in the presentation.

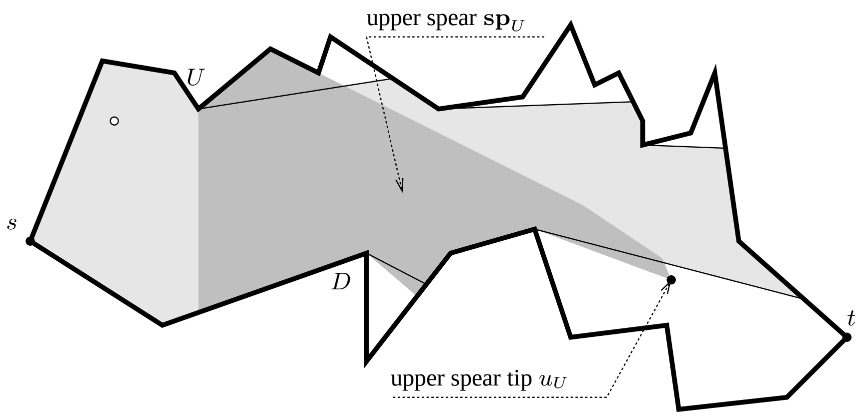


Figure 9: The dark shaded area is the upper spear.

DEFINITION 4.2 *For X being one of U or D, let the kernel expansion of $\mathbf{Q}_X$ be the spear with respect to X, i.e.,*

$$\mathbf{sp}_X \overset{def}{=} \mathbf{ke}(\mathbf{Q}_X).$$

Hence, we have the two spear types $\mathbf{sp}_U$ and $\mathbf{sp}_D$ corresponding to the sequences of pockets we are currently considering; see Figure 9 for an example of the upper spear $\mathbf{sp}_U$.

Since the intersection of monotone polygons is also monotone, a spear can be computed by a plane sweep algorithm going from left to right, maintaining the upper and lower boundaries of the kernel expansions; see Section 4.3.1.

The rightmost intersection point between the upper and lower boundary of a spear is called the *spear tip* and we denote spear tips by u_U and u_D, corresponding to the two types of spears; see Figure 9. By the definition of these points, u_X has the property of being the rightmost point that sees all the points of the pockets of type X to the left of it.

The spears are dependent on the placement of the previously placed guards so we will henceforth refer to them as $\mathbf{sp}_U(\mathcal{G}^p)$ and $\mathbf{sp}_D(\mathcal{G}^p)$ given the partial guard set $\mathcal{G}^p$. For each spear $\mathbf{sp}_X(\mathcal{G}^p)$, we similarly parameterize the spear tip $u_X(\mathcal{G}^p)$. If $\mathcal{G}^p = \emptyset$, the upper and lower spears $\mathbf{sp}_U(\emptyset)$ and $\mathbf{sp}_D(\emptyset)$ together with the upper and lower spear tips $u_U(\emptyset)$ and $u_D(\emptyset)$ are well defined since all of $\mathbf{P}$ is considered a boundary pocket.

We can now give the details of our guarding algorithm, displayed in Figure 10. Each iteration of the algorithm begins by computing the spears and the spear tips, which we show how to do efficiently in Section 4.3.1. Step 4 selects the leftmost of $u_U(\mathcal{G})$ and $u_D(\mathcal{G})$, placing a guard g at this point in Step 5. Step 6 results in the addition of g' to the guard set only if $\mathbf{sp}_{\bar{X}}(\mathcal{G})$ actually intersects l_g, where $\bar{X}$ denotes the remaining pocket type different from X. We show how to perform Step 7 efficiently in Section 4.3.2.

Since all upper and lower pockets are guarded after the algorithm has concluded, we know that the complete boundary of $\mathbf{P}$ is seen by the guards placed. In fact, we prove in Lemma 4.3 that also the interior of $\mathbf{P}$ is seen.

We claim the following theorem and dedicate Sections 4.2 and 4.3 to proving it.

THEOREM 2 *The algorithm* GUARD-MONOTONE-POLYGON *computes a guard cover of size at most $30OPT$ for a monotone polygon $\mathbf{P}$ in polynomial time, where OPT is the size of the smallest guard cover for $\mathbf{P}$.*

```
Algorithm GUARD-MONOTONE-POLYGON
Input:  A monotone polygon $\mathbf{P}$
Output: A guard cover for $\mathbf{P}$
1  $\mathcal{G} := \emptyset$
   while not all pockets are guarded do
2      Compute $\mathbf{sp}_U(\mathcal{G})$ and $u_U(\mathcal{G})$ if there are upper pockets
3      Compute $\mathbf{sp}_D(\mathcal{G})$ and $u_D(\mathcal{G})$ if there are lower pockets
4      if $u_U(\mathcal{G})$ is to the left of $u_D(\mathcal{G})$ then $X := U, \bar{X} := D$ else $X := D, \bar{X} := U$ endif
5      Place a guard $g$ at $u_X(\mathcal{G})$ and let $l_g$ be the vertical line segment through $g$
6      Place a guard $g'$ at an intersection point of $l_g$ and $\mathbf{sp}_{\bar{X}}(\mathcal{G})$, if they intersect
7      Place a guard $\hat{g}$ on $l_g$ so that $u_X(\mathcal{G} \cup \{g, g', \hat{g}\})$ lies as far to the right as possible
8      $\mathcal{G} := \mathcal{G} \cup \{g, g', \hat{g}\}$
   endwhile
   return $\mathcal{G}$
End GUARD-MONOTONE-POLYGON
```

Figure 10: The algorithm for guarding monotone polygons.

To help the reader, we provide a table of the notation we introduce.

Symbol	**Name**	**Explanation**
$\mathbf{VP}(p)$		the visibility polygon of the point p
$\mathbf{VP}_L(p, q)$,		the part of $\mathbf{VP}(p)$ to the left of q
$\mathbf{VP}_R(p, q)$		the part of $\mathbf{VP}(p)$ to the right of q
$\mathbf{VP}_B(p, q, r)$		the part of $\mathbf{VP}(p)$ between q and r
$\mathbf{R}_L(q)$		the set of points of $\mathbf{R}$ to the left of q
$\mathbf{ke}(\mathbf{R})$	*kernel expansion*	the set of points p that see all of $\mathbf{R}_L(p)$
$\mathbf{Q}_X$		the union of type X pockets
$\mathbf{sp}_X$	*spear*	the kernel expansion of $\mathbf{Q}_X$, $\mathbf{ke}(\mathbf{Q}_X)$
u_X	*spear tip*	the rightmost point of a spear, $\mathbf{sp}_X$
l_p		the line or segment through the point p
v_X	*base*	v_X lies on the boundary of $\mathbf{VP}(u_X)$ and u_X lies on the boundary of $\mathbf{VP}(u_X)$

4.2 Correctness and Approximation Factor

4.2.1 Correctness

We know from the construction of algorithm GUARD-MONOTONE-POLYGON that it will guard the boundary of the polygon. However, we need to prove that it will also guard the interior of the polygon. To prove this, it is sufficient to show that our algorithm never produces a middle pocket. We do this in two steps. The first step is to show that all guards cannot be on one side of a middle pocket, i.e., a middle pocket can never be generated to the right of the guards as they are placed by the algorithm. The second step is to show that when the algorithm

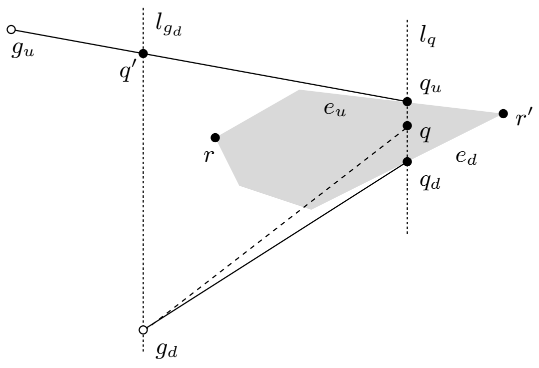


Figure 11: Illustrating the proof of Lemma 4.2.

places new guards, a middle pocket to the left of these guards can never be generated.

LEMMA 4.2 *Consider a middle pocket* $\mathbf{p}$ *of a partial guard set in a monotone polygon. Let* r *be the leftmost point in* $\mathbf{p}$*. Not all guards of the partial guard set can be to the left of* r*.*

PROOF: Assume for a contradiction, that all the guards are to the left of r. Let r' be the rightmost point of $\mathbf{p}$ and let e_u and e_d be the upper and lower edges respectively of $\mathbf{p}$ that are also adjacent to r'. Neither e_u nor e_d can be vertical since this would immediately give a contradiction, requiring a guard to the right of r. Let q be a point interior to $\mathbf{p}$, below e_u and above e_d, let l_q be the vertical line through q and let q_u and q_d be the intersection points of l_q with e_u and e_d respectively. Since all guards are to the left of r, no single guard can see both points q_u and q_d, so there are at least two different guards g_u and g_d that see these points. Furthermore, the lines of sight from g_u to q_u and from g_d to q_d cannot cross, otherwise, either g_u or g_d will see points inside the middle pocket giving us a contradiction. Assume without loss of generality that g_u is further to the left than g_d, otherwise we reverse the roles of g_u and g_d in the following argument.

Let l_{g_d} be the vertical line through g_d. It intersects the line of sight from g_u to q_u at q'. The four points g_d, q_d, q_u and q' form a convex polygon with the guard g_d in one corner. Hence, q is seen by g_d, contradicting our assumption that q is in a middle pocket; see Figure 11. □ □

Lemma 4.2 shows that as algorithm GUARD-MONOTONE-POLYGON places guards incrementally in the polygon, it can never generate a middle pocket to the right of the rightmost guard placed so far. Next, we show that the algorithm will not generate a middle polygon to the left of this guard either, thus giving us the following lemma.

LEMMA 4.3 *The algorithm* GUARD-MONOTONE-POLYGON *never introduces middle pockets, and hence, produces a complete guard cover.*

PROOF: We make a proof by contradiction and assume that in iteration i of the algorithm, as guards g_i, g'_i and $\hat{g}_i$ are positioned in the polygon, a middle pocket $\mathbf{p}_i$ is generated between guard triples g_j, g'_j, $\hat{g}_j$ and g_{j+1}, g'_{j+1}, $\hat{g}_{j+1}$, where $j < i$. We assume furthermore that i is the first iteration index that generates a middle pocket and hence that $\mathbf{p}_i$ is generated in this iteration. Let $\mathcal{G}_i$ denote the set of guards placed by the algorithm from iteration 1 until iteration i has completed.

Let r be a point in $\mathbf{p}_i$ and consider the situation just after iteration $i-1$. The point r belongs to a pocket $\mathbf{p}_{i-1}$ that is either an upper, a lower or a boundary pocket after this iteration. Consider the situation as the algorithm places guards g_i, g'_i and $\hat{g}_i$ during iteration i. By Lemma 4.2, r lies to the left of g_i since there are no other guards to the right of g_i. Without loss of generality, we can assume that g_i is placed at $u_U(\mathcal{G}_i)$ in Step 5 of the algorithm, as the argument when g_i is placed at $u_D(\mathcal{G}_i)$ is completely analogous. We make a case analysis on whether $\mathbf{p}_{i-1}$ is an upper, a lower or a boundary pocket.

Assume first that $\mathbf{p}_{i-1}$ is an upper pocket, then we have an immediate contradiction since $g_i = u_U(\mathcal{G}_{i-1})$ and $u_U(\mathcal{G}_{i-1})$, by definition, sees all points in the upper pockets to the left of itself, and hence, also the point r.

Assume next that $\mathbf{p}_{i-1}$ is a lower pocket, then either the vertical line l_{g_i} through g_i intersects $\mathbf{sp}_D(\mathcal{G}_{i-1})$, in which case g'_i sees r, giving us a contradiction, or l_{g_i} does not intersect $\mathbf{sp}_D(\mathcal{G}_{i-1})$. In this case, $u_D(\mathcal{G}_{i-1})$ is to the left of $u_U(\mathcal{G}_{i-1})$, contradicting the selection in Step 4.

If we assume that $\mathbf{p}_{i-1}$ is a boundary pocket, it lies to the right of g_{i-1}. Assume for a contradiction that the point r, to the left of g_i, is not seen by g_i or g'_i. This means that some part of the polygon boundary hides r from g_i and g'_i. Assume first that this is U, i.e., the shortest path $SP(r, g_i)$ touches U at some vertex v. This means that the vertex v' on U to the left of v is not seen, contradicting that $g_i = u_U(\mathcal{G}_{i-1})$, since $v' \in \mathcal{P}_U(\mathbf{p}_{i-1})$; see the definition of $\mathbf{Q}_U$ in Section 4.1. On the other hand, if $SP(r, g_i)$ touches D at some vertex v, then the vertex v' on D to the left of v is not seen, giving us that $u_D(\mathcal{G}_{i-1})$ is to the left of $u_U(\mathcal{G}_{i-1})$, since $v' \in \mathcal{P}_D(\mathbf{p}_{i-1})$, contradicting the selection in Step 4.

Therefore, algorithm GUARD-MONOTONE-POLYGON produces a guard cover that sees all the boundary of the polygon and it never generates a middle pocket. Hence it produces a complete guard cover for the monotone polygon. □ □

4.2.2 Bases and Shadows

We continue with a discussion that becomes fairly technical. We associate a specific region, a *shadow*, to the right of a spear and show that if two spears of the same type, i.e., upper or lower spears, generated by a partial guard set that obeys certain conditions, then the associated shadows do not intersect. We use this information in the next section to bound the number of guards that our algorithm will produce.

We begin by defining two concepts. Fix a partial guard cover $\mathcal{G}^p$ and let $\mathbf{sp}_X$ be the spears with respect to $\mathcal{G}^p$, for the pocket type X being U or D. To each spear $\mathbf{sp}_X$ we associate a point called a *base* of the spear, denoted v_X, with X being U or D.

Let l_{u_X} be the vertical line through the spear tip u_X of the spear $\mathbf{sp}_X$ and let $\mathbf{Q}_X$ be the region, the set of pockets, such that $\mathbf{sp}_X = \mathbf{ke}(\mathbf{Q}_X)$; see Definition 4.2.

DEFINITION 4.3 *A base of $\mathbf{sp}_X$ is the rightmost point v_X in $\mathbf{Q}_X$ and on the boundary X of $\mathbf{P}$ such that the spear tip u_X lies on the boundary of $\mathbf{VP}(v_X)$ and v_X lies on the boundary of $\mathbf{VP}(u_X)$.*

A point v_U is called an upper base and a point v_D is called a lower base.

Note that a base can lie on a boundary edge infinitely close to a vertex without being on the vertex. See Figure 12 for an example of an upper base.

The second concept that we define is that of a *shadow*.

DEFINITION 4.4 *For a spear $\mathbf{sp}_X$ with X being one of U or D, define the shadow of $\mathbf{sp}_X$, denoted $\mathbf{shd}_X$, to be the part of the visibility polygon of the base v_X strictly to the right of u_X. Hence, $\mathbf{shd}_X = \mathbf{VP}_R(v_X, u_X) \backslash l_{u_X}$, where l_{u_X} is the vertical line through u_X; see Figure 12.*

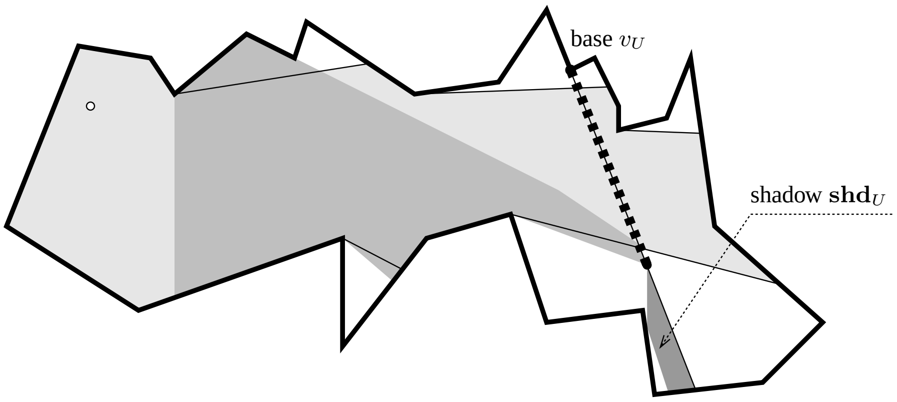


Figure 12: Example of a base and shadow for an upper spear.

We parameterize the shadows, in the same way as the spears, to be dependent on the placement of the previously placed guards and refer to them as $\mathbf{shd}_U(\mathcal{G}^p)$ and $\mathbf{shd}_D(\mathcal{G}^p)$ given the partial guard set $\mathcal{G}^p$.

We prove a technical lemma that will be useful to bound the number of guards produced by our algorithm.

LEMMA 4.4 *If $\mathcal{G}_-$ and $\mathcal{G}_+$ are two partial guard covers of $\mathbf{P}$ such that $\mathcal{G}_- \subset \mathcal{G}_+$ and $u_X(\mathcal{G}_-) \in \mathcal{G}_+$, for X being one of U or D, then $\mathbf{shd}_X(\mathcal{G}_-) \cap \mathbf{shd}_X(\mathcal{G}_+) = \emptyset$.*

PROOF: Since the spear type is fixed in each case, we can simplify our notation and let $u_- = u_X(\mathcal{G}_-)$, $u_+ = u_X(\mathcal{G}_+)$, $v_- = v_X(\mathcal{G}_-)$, $v_+ = v_X(\mathcal{G}_+)$, $\mathbf{shd}_- = \mathbf{shd}_X(\mathcal{G}_-)$, and $\mathbf{shd}_+ = \mathbf{shd}_X(\mathcal{G}_+)$.

We denote by $\bar{X}$ the opposite boundary of X in $\mathbf{P}$, i.e., if X is U then $\bar{X}$ is D and vice versa. Let $\mathbf{Q}_-$ and $\mathbf{Q}_+$ be the pocket regions of the two guard sets with respect to X.

If u_+ lies on the boundary $\bar{X}$ (except for the degenerate case when u_+ lies on a reflex vertex of $\bar{X}$), then we immediately have that $\mathbf{shd}_+ = \emptyset$ proving the lemma.

Assume now that u_+ does not lie on $\bar{X}$ (or that the degenerate case has occurred), then the spear tip u_+ is adjacent to two boundary edges e and e' of the spear, where e' is part of the line segment $[v_+, u_+]$. Consider the extension of e, the other edge, from u_+ towards the left until it reaches the exterior of $\mathbf{P}$ at p.

We claim that the line segment $[p, u_+]$ must touch the boundary $\bar{X}$ at some point. Assume that it does not, then there is a point on the extension of $[v_+, u_+]$ towards the right that sees as much of $\mathbf{Q}_+$ as u_+ does, thus contradicting that u_+ is a spear tip. This also shows that $[p, u_+]$ touches the boundary of some pocket $\mathbf{p}$ in $\mathbf{Q}_+$. Let q denote the leftmost point on $\bar{X}$ that intersects the segment $[p, u_+]$.

The line segment $[p, q]$ partitions $\mathbf{P}$ into two subpolygons, $\mathbf{P}_+$ and $\mathbf{P}_-$, where $\mathbf{P}_+$ contains u_+. This gives us two cases; see Figure 13.

v_- lies in $\mathbf{P}_-$.

If the extension of $[v_-, u_-]$ towards the right does not cross $[p, q]$, then all of $\mathbf{shd}_-$ also lies in $\mathbf{P}_-$ and cannot intersect $\mathbf{shd}_+$ which lies in $\mathbf{P}_+$.

If the extension of $[v_-, u_-]$ towards the right does cross $[p, q]$, then extend $[q, u_+]$ towards the right until it reaches the exterior of $\mathbf{P}$ at p'. In this case, all of $\mathbf{shd}_-$ in $\mathbf{P}_+$ lies on the same side of $[q, p']$ as v_+,

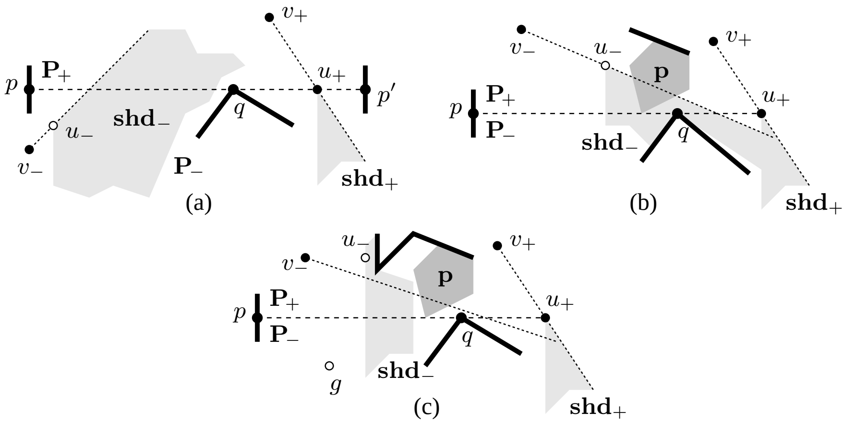


Figure 13: Illustrating the proof of Lemma 4.4 with $X = U$.

since v_+ lies on X and q lies on $\bar{X}$. Hence, $\mathbf{shd}_+$ lies on the opposite side of $[q, p']$ so $\mathbf{shd}_-$ and $\mathbf{shd}_+$ cannot intersect; see Figure 13(a).

v_- lies in $\mathbf{P}_+$.

If u_- lies in $\mathbf{P}_-$, then all of $\mathbf{shd}_-$ lies in $\mathbf{P}_-$ and the two shadows cannot intersect.

Assume now that u_- lies in $\mathbf{P}_+$ and that v_- sees points in $\mathbf{shd}_+$. We have two subcases.

If u_- also sees points in $\mathbf{shd}_+$, we have an immediate contradiction since then u_- sees points in some pocket $\mathbf{p}$ of $\mathbf{Q}_+$ or points of the path $\mathcal{P}_X(\mathbf{p})$ of $\mathbf{Q}_+$, if $\mathbf{p}$ is a boundary pocket; see Figure 13(b).

If u_- does not see points in $\mathbf{shd}_+$, there is a part of the boundary X blocking vision between u_- and $\mathbf{shd}_+$. Note that u_- cannot see q since otherwise it would also see points of $\mathbf{p}$. Therefore, there is second guard g in $\mathcal{G}_+$ seing q. The guard g must lie in $\mathbf{P}_-$ otherwise it sees points in $\mathbf{p}$. Furthermore, visibility from g into $\mathbf{p}$ must be blocked by the boundary X, which must then cross the line segment $[p, q]$, a contradiction; see Figure 13(c).

This concludes the proof. □ □

4.2.3 Serial Guard Covers

We define a special type of guard set that will help us prove the approximation factor of our algorithm.

DEFINITION 4.5 *We define restricted guards as follows:*

- *For a region $\mathbf{R}$, a guard g is $\mathbf{R}$-restricted, if we only consider the restricted visibility polygon of g to be $\overline{\mathbf{VP}}(g) \overset{def}{=} \mathbf{VP}(g) \cap \mathbf{R}$.*
- *A guard g is a left (or right) guard, if g is $\mathbf{VP}_L(g)$-restricted (or $\mathbf{VP}_R(g)$-restricted).*

Next, we define a *strip subdivision* of $\mathbf{P}$.

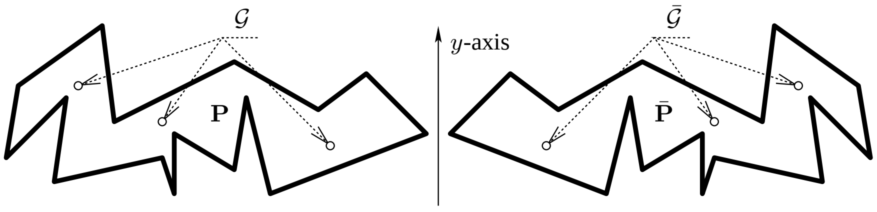


Figure 14: The reflected polygon $\bar{\mathbf{P}}$.

DEFINITION 4.6 *A strip subdivision of a monotone polygon $\mathbf{P}$ is a subdivision of the polygon by the introduction of vertical segments connecting the upper and lower boundary. Each strip is a subpolygon of $\mathbf{P}$ bounded by a left vertical edge (possibly degenerating to the left end point s), a portion of the upper boundary U, a right vertical edge (possibly degenerating to the right end point t) and a portion of the lower boundary D.*
We are now in a position to define *serial guard sets*.

DEFINITION 4.7 *A guard set is serial, if $\mathbf{P}$ is subdivided into K strips $\mathrm{s}_0, \ldots, \mathrm{s}_{K-1}$ ordered from left to right so that the left edge of strip s_0 has zero or one s_0-restricted guard, the left edge of every other strip s_i, $0 < i < K$, has exactly one s_i-restricted guard and the right edge of each strip s_i, $0 \leq i < K$, contains zero or more left guards. No other guards are placed in the polygon.*

We say that a serial guard set is an upper serial guard cover, if the s_i-restricted guard on the left edge of each strip s_i, together with the left guards on the right edges of the strips, sees the upper boundary of s_i. Similarly, a guard set is a lower serial guard cover, if the s_i-restricted guard on the left edge of each strip s_i, together with the left guards on the right edges of the strips, sees the lower boundary of s_i.

LEMMA 4.5 *A monotone polygon has an upper serial guard cover with at most $2OPT$ s_i-restricted guards and at most $3OPT$ left guards.*

PROOF: Let $\mathcal{G}$ be a guard cover for $\mathbf{P}$. Reflect $\mathbf{P}$ along the y-axis to get the reversed polygon $\bar{\mathbf{P}}$ having the guard cover $\bar{\mathcal{G}}$, the set $\mathcal{G}$ reflected along the y-axis; see Figure 14.

Our proof is constructive and iteratively places restricted guards in the polygon $\bar{\mathbf{P}}$. Do a plane sweep from left to right on $\bar{\mathbf{P}}$. Initially, let $\mathcal{H}_0$ be the empty guard set and let l_0 be the vertical line through the leftmost point of $\bar{\mathbf{P}}$. Iteratively, given the partial guard set $\mathcal{H}_j$ and the line l_j, we construct the next partial guard set $\mathcal{H}_{j+1}$ and the next vertical line l_{j+1} as follows:

Obtain the upper spear tip $u = u_U(\mathcal{H}_j)$ and let l_{j+1} be the maximal vertical line segment through u interior to $\mathbf{P}$. Let $\mathcal{H}_{j+1}$ include the guards in $\mathcal{H}_j$ and let $\bar{\mathrm{s}}_j$ be the strip in $\bar{\mathbf{P}}$ bounded by l_j and l_{j+1}. We place the following additional guards on l_{j+1},

1. an $\bar{\mathrm{s}}_j$-restricted guard $\bar{g}_j$ at u,
2. a right guard at u,
3. each guard from $\bar{\mathcal{G}}$ in $\bar{\mathrm{s}}_j$ is moved along its shortest path to s, the rightmost point of $\bar{\mathbf{P}}$, until it reaches l_{j+1}. At this points place a right guard.

Each of the guards thus placed is added to $\mathcal{H}_{j+1}$; see Figure 15. According to Lemma 2.1, any right guard $\bar{g}_R$ on l_{j+1} will see at least as much to the right of l_{j+1} in $\bar{\mathbf{P}}$ as the original guard $\bar{g}$ in $\bar{\mathcal{G}}$. Hence, we have the

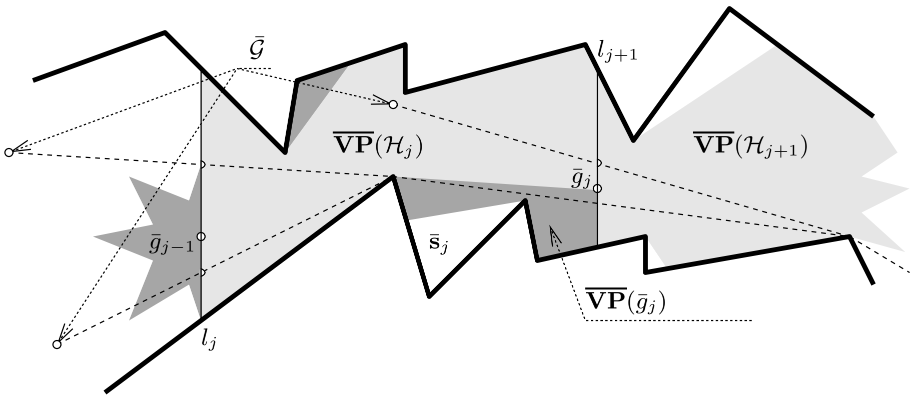

$\bar{\mathcal{G}}$
l_{j+1}
$\overline{\mathbf{VP}}(\mathcal{H}_j)$
$\overline{\mathbf{VP}}(\mathcal{H}_{j+1})$
$\bar{g}_j$
$\bar{g}_{j-1}$
$\bar{\mathbf{s}}_j$
$\overline{\mathbf{VP}}(\bar{g}_j)$
l_j

Figure 15: Illustrating the proof of Lemma 4.5.

invariant that the guard set $\mathcal{H}_{j+1}$ sees at least as much of the upper boundary of $\bar{\mathbf{P}}$ as the guards to the left of u in $\bar{\mathcal{G}}$.

The process terminates after K iterations when the plane sweep reaches the rightmost point of $\bar{\mathbf{P}}$. In the last iteration we have two possibilities. Either, the right guards in $\mathcal{H}_{K-1}$ together see the upper boundary of $\bar{\mathbf{s}}_{K-1}$, in which case we do not have to add any $\bar{\mathbf{s}}_{K-1}$-restricted guard at the right edge of $\bar{\mathbf{s}}_{K-1}$, or $\bar{\mathbf{s}}_{K-1}$ must by necessity contain guards from $\bar{\mathcal{G}}$. We differentiate between these cases when we count the number of guards placed.

The $\bar{\mathbf{s}}_j$-restricted guard $\bar{g}_j$ placed at $u_U(\mathcal{H}_j)$ will see the upper boundary of $\bar{\mathbf{s}}_j$ together with the right guards in $\mathcal{H}_j$. Hence, the restricted guard set $\mathcal{H}_K$ sees the upper boundary of $\bar{\mathbf{P}}$.

We count the number of guards placed according to their type, 1, 2, or 3, above. The number of Type 3 right guards in $\mathcal{H}_K$ is $|\bar{\mathcal{G}}| = |\mathcal{G}|$ since each of these right guards corresponds to a guard in $\mathcal{G}$.

The number of Type 2 right guards is the same as the number of Type 1 $\bar{\mathbf{s}}_j$-restricted guards since both types are placed at upper spear tips u.

It remains to count the Type 1 $\bar{\mathbf{s}}_j$-restricted guards. If $\bar{\mathbf{s}}_j$ contains guards from $\bar{\mathcal{G}}$, we can associate $\bar{g}_j$ to such a guard $\bar{g}$ in $\bar{\mathcal{G}}$. In particular, if the process places guard $\bar{g}_{K-1}$ in the last iteration, there is a guard from $\bar{\mathcal{G}}$ in $\bar{\mathbf{s}}_{K-1}$ and we can associate $\bar{g}_{K-1}$ to this guard.

On the other hand, if $\bar{\mathbf{s}}_j$ contains no guards from $\bar{\mathcal{G}}$, then an upper base $v_U(\mathcal{H}_j)$ of $\mathbf{sp}_U(\mathcal{H}_j)$ is not seen by the guards in $\bar{\mathcal{G}}$ to the left of l_j, i.e., in $\bar{\mathbf{s}}_0, \ldots, \bar{\mathbf{s}}_{j-1}$, since no right guard in $\mathcal{H}_j$ sees the base. Therefore, the base $v_U(\mathcal{H}_j)$ must be seen by a guard $\bar{g}$ in $\bar{\mathcal{G}}$ lying in the upper shadow $\mathbf{shd}_U(\mathcal{H}_j)$. Since there is a Type 2 right guard at the position of $\bar{g}_j$, the prerequisites of Lemma 4.4 are fulfilled and we know that no two upper shadows $\mathbf{shd}_U(\mathcal{H}_j)$ and $\mathbf{shd}_U(\mathcal{H}_{j'})$ intersect for $j \neq j'$, and hence, $\bar{g}$ can only see one base. We can therefore associate $\bar{g}_j$ to a guard $\bar{g}$ from $\bar{\mathcal{G}}$ in the upper shadow $\mathbf{shd}_U(\mathcal{H}_j)$.

Note that, if $\bar{\mathbf{s}}_{K-1}$ contains no guard from $\bar{\mathcal{G}}$, then this strip is completely seen by the guards in previous strips and the process places no $\bar{\mathbf{s}}_{K-1}$-restricted guard in $\bar{\mathbf{s}}_{K-1}$.

In this way, any guard $\bar{g}$ in $\bar{\mathcal{G}}$ can be associated to at most two $\bar{\mathbf{s}}_j$-restricted guards. Hence, the number of Type 1 $\bar{\mathbf{s}}_j$-restricted guards, and thus also the number of Type 2 right guards, is at most $2|\bar{\mathcal{G}}| = 2|\mathcal{G}|$.

Next, reflect the set $\mathcal{H}_K$ back along the y-axis to become a guard set $\mathcal{U}$ of $\mathbf{P}$. We claim that $\mathcal{U}$ is upper serial with at most $2|\mathcal{G}|$ $\mathbf{s}_i$-restricted guards, for strips $\mathbf{s}_i$, $0 \leq i < K$, and at most $3|\mathcal{G}|$ left guards. This follows since a strip $\bar{\mathbf{s}}_j$ in $\bar{\mathbf{P}}$ when reflected back becomes a strip $\mathbf{s}_i$ in $\mathbf{P}$, with $i = K - j - 1$. Each $\bar{\mathbf{s}}_j$-restricted guard $\bar{g}_j$ in $\mathcal{H}_K$ lies on the right edge of $\bar{\mathbf{s}}_j$ and sees the upper boundary of $\bar{\mathbf{s}}_j$ so the corresponding $\mathbf{s}_i$-restricted guard g_i in $\mathcal{U}$ lies on the left edge of $\mathbf{s}_i$ and sees the upper boundary of $\mathbf{s}_i$. Finally, the right guards of $\mathcal{H}_K$ on the left edges of strips in $\bar{\mathbf{P}}$ correspond to left guards of $\mathcal{U}$ in right edges of strips in $\mathbf{P}$. The number of guards has not changed so $\mathcal{U}$ is upper serial as claimed.

By choosing $\mathcal{G}$ to be an optimal guard cover for $\mathbf{P}$, we have that $|\mathcal{G}| = OPT$, thus proving the lemma. □ □

We can, using the same proof technique, show a corresponding lemma for lower serial guards.

4.2.4 Approximation Factor

Next, we establish the approximation factor of the algorithm.

LEMMA 4.6 *The algorithm* GUARD-MONOTONE-POLYGON *places at most* $30OPT$ *guards in* $\mathbf{P}$*, where* OPT *is the size of the smallest guard cover for* $\mathbf{P}$*.*

PROOF: To bound the total number of guards, we establish the number of guards placed by Steps 5–7 throughout the iterations of algorithm GUARD-MONOTONE-POLYGON. To do so, we compare the number of guards placed in each step with the size of an upper and a lower serial guard cover.

Let $\mathcal{G}_U$ be the set of guards assigned in Steps 5–7 in the iterations of the algorithm when the selection in Step 4 makes $X = U$ and $\bar{X} = D$; see Figure 10. Similarly, let $\mathcal{G}_D$ be the guards assigned when $X = D$ and $\bar{X} = U$.

Consider first the set $\mathcal{G}_U$ and order the guard triples in this set from left to right,

$$\mathcal{G}_U = \{g_1, g'_1, \hat{g}_1, g_2, g'_2, \hat{g}_2, \ldots\}.$$

In iteration i of the loop, our algorithm performs Steps 5–7 with $X = U$ and places guards g_j, g'_j and $\hat{g}_j$, all having the same x-coordinate, with $j \le i$ being the proper index in $\mathcal{G}_U$. The next time the algorithm performs Steps 5–7 with $X = U$ it places guards g_{j+1}, g'_{j+1} and $\hat{g}_{j+1}$.

Let $\mathcal{G}_i$ be the set of guards placed in iterations 1 to i by the algorithm. The guards g_j, g'_j and $\hat{g}_j$ are the rightmost guards in $\mathcal{G}_i$.

We compare the number of guards in $\mathcal{G}_U$ with the size of an upper serial guard cover $\mathcal{U}$ and show that $\mathcal{G}_U$ contains at most $3|\mathcal{U}|$ guards. To do this, we construct a secondary guard set $\mathcal{H}$ incrementally starting with the empty guard set $\mathcal{H}_0$. For every index $j > 0$ we go through the guard triples in $\mathcal{G}_U$ as follows:

If $\mathcal{U}$ has an $\mathbf{s}$-restricted guard g in the interval between g_j and g_{j+1} (in the case of $j = 0$ we consider the leftmost end point s of $\mathbf{P}$ to be the imaginary guard g_0), for some strip $\mathbf{s}$, then we let $\mathcal{H}_j := \{g_j, g'_j, \hat{g}_j\} \cup \mathcal{H}_{j-1}$, if $j > 0$.

If $\mathcal{U}$ has no $\mathbf{s}$-restricted guard in the interval between g_j and g_{j+1}, then this whole interval is contained in a strip $\mathbf{s}$ associated to $\mathcal{U}$. This means that the upper base $v_U(\mathcal{G}_i)$ is either seen by the $\mathbf{s}$-restricted guard $g_{\mathbf{s}}$ to the left of g_j or by a left guard in $\mathcal{U}$ in the upper shadow $\mathbf{shd}_U(\mathcal{G}_i)$.

If $g_{\mathbf{s}}$ sees $v_U(\mathcal{G}_i)$ then the shortest path from $g_{\mathbf{s}}$ to t, the rightmost point of $\mathbf{P}$, crosses the vertical line through g_j at a point p. Let $\mathcal{H}_j := \{g_j, g'_j, p\} \cup \mathcal{H}_{j-1}$, i.e., we exchange the guard $\hat{g}_j$ for a guard at p. We claim that $g_{\mathbf{s}}$ does not see $v_U(\mathcal{H}_j)$ but this follows immediately by Lemma 2.1. Furthermore, $u_U(\mathcal{H}_j)$ is not to the right of $u_U(\mathcal{G}_i)$, since by our algorithm $\hat{g}_j$ is placed so that $u_U(\mathcal{G}_i)$ is as far to the right as possible.

If $g_{\mathbf{s}}$ does not see $v_U(\mathcal{G}_i)$ then we let $\mathcal{H}_j := \{g_j, g'_j, \hat{g}_j\} \cup \mathcal{H}_{j-1}$. We have that $v_U(\mathcal{G}_i) = v_U(\mathcal{H}_j)$.

The set $\mathcal{H}$ constructed according to the rules given above obeys three important criteria:

1. $|\mathcal{H}| = |\mathcal{G}_U|$,
2. for each j, either there is an $\mathbf{s}$-restricted guard between g_j and g_{j+1} or no $\mathbf{s}$-restricted guard $g_\mathbf{s}$ in $\mathcal{U}$ sees the upper base $v_U(\mathcal{H}_j)$,
3. $u_U(\mathcal{H}_j)$ is not to the right of $u_U(\mathcal{G}_i)$, where i is the iteration index when guards g_j, g'_j and $\hat{g}_j$ are placed.

Let us count the number of guards in $\mathcal{H}$. If two subsequent triples $\{g_j, g'_j, p_j\}$ and $\{g_{j+1}, g'_{j+1}, p_{j+1}\}$ in $\mathcal{H}$ have an $\mathbf{s}$-restricted guard $g_\mathbf{s}$ in the interval between them, then we associate the triple $\{g_j, g'_j, p_j\}$ to $g_\mathbf{s}$. We call such an association an α-association. By construction, an $\mathbf{s}$-restricted guard $g_\mathbf{s}$ in $\mathcal{U}$ can only be α-associated to a guard triple in $\mathcal{H}$ once.

If two subsequent triples $\{g_j, g'_j, p_j\}$ and $\{g_{j+1}, g'_{j+1}, p_{j+1}\}$ in $\mathcal{H}$ do not have any $\mathbf{s}$-restricted guard $g_\mathbf{s}$ in the interval between them, then we know that the upper base $v_U(\mathcal{H}_j)$ is seen by a left guard g in $\mathcal{U}$ in the upper shadow $\mathbf{shd}_U(\mathcal{H}_j)$ and we associate the triple $\{g_j, g'_j, p_j\}$ to g. We call such an association a β-association. Since the upper spear $\mathbf{sp}_U(\mathcal{H}_j)$ and $\mathbf{sp}_U(\mathcal{H}_{j'})$ obey the prerequisites of Lemma 4.4, for any $j' \neq j$, the two upper shadows $\mathbf{shd}_U(\mathcal{H}_j)$ and $\mathbf{shd}_U(\mathcal{H}_{j'})$ do not intersect. This means that a left guard g can only be β-associated to a guard triple in $\mathcal{H}$ once.

From this we can deduce that the number of guard triples in $\mathcal{H}$ is at most the number of $\mathbf{s}$-restricted guards and left guards in $\mathcal{U}$ together. From Lemma 4.5, we know that this is at most $5OPT$. By a completely symmetrical argument we can construct a set of guard triples $\mathcal{H}'$ of the same size as the set $\mathcal{G}_D$ and deduce that the number of guard triples in this set is also bounded by $5OPT$.

The total number of guards constructed by our algorithm is therefore bounded by

$$|\mathcal{G}_U| + |\mathcal{G}_D| = |\mathcal{H}| + |\mathcal{H}'| \leq 3|\mathcal{U}| + 3|\mathcal{D}| \leq 30OPT,$$

as claimed. □ □

4.3 Computation

4.3.1 Computing Kernel Expansions

Let p, q and r be three points in $\mathbf{P}$. We let $\mathbf{VP}_B(p, q, r)$ denote the part of the visibility polygon $\mathbf{VP}(p)$ between the points q and r. Let $\mathbf{R}$ be a possibly disconnected polygonal region in $\mathbf{P}$ having m vertices and assume that the vertices are ordered $v_1, \ldots, v_m$ from left to right. We claim the following lemma.

LEMMA 4.7

$$\mathbf{ke}(\mathbf{R}) = \bigcup_{i=1}^{m-1} \left(\bigcap_{j=1}^{i} \mathbf{VP}_B(v_j, v_i, v_{i+1}) \right) \bigcup \left(\bigcap_{j=1}^{m} \mathbf{VP}_R(v_j, v_m) \right).$$

PROOF: From Definition 4.1 we have that $\mathbf{ke}(\mathbf{R}) = \{p \geq v_1 \mid \mathbf{R}_L(p) \subseteq \mathbf{VP}(p)\}$ where $\mathbf{R}_L(p)$ is the part of $\mathbf{R}$ to the left of p.

Let p be a point between v_i and v_{i+1}. We show that p is in $\mathbf{ke}(\mathbf{R})$ if and only if $p \in \bigcap_{j=1}^{i} \mathbf{VP}(v_j)$.

Assume first that $p \notin \bigcap_{j=1}^{i} \mathbf{VP}(v_j)$. In this case, there is a vertex v_j of $\mathbf{R}$ such that p does not see v_j. Hence, there is a point in $\mathbf{R}$ to the left of p, the vertex v_j, not seen by p, so p is not in $\mathbf{ke}(\mathbf{R})$ proving the first implication of the equivalence.

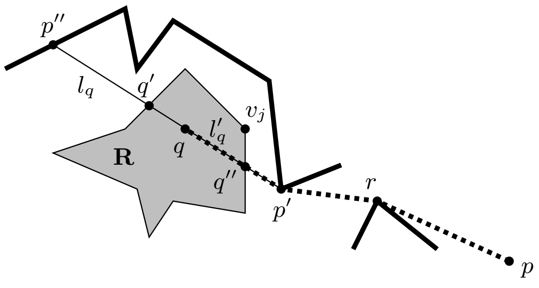


Figure 16: Illustrating the proof of Lemma 4.7.

Assume next that $p \notin \mathbf{ke}(\mathbf{R})$. In this case, there is a point $q \in \mathbf{R}_L(p)$ not seen from p. Consider the shortest path $SP(p, q)$. Let $[p', q]$ be the last segment and let $[r, p']$ be the penultimate segment of $SP(p, q)$. Since p does not see q, $SP(p, q)$ consists of at least two segments and because of the monotonicity of $\mathbf{P}$, the point p' is a vertex of $\mathbf{P}$ to the left of p.

Let l_q be the maximal line segment interior to $\mathbf{P}$ from p' through q to p''. The segment l_q contains a maximal subsegment l'_q completely contained in $\mathbf{R}$. Let q' and q'' be the two end points of l'_q, with q' to the left of q''; see Figure 16.

Assume that $[r, p', p'']$ forms a right turn. Follow the boundary of $\mathbf{R}$ in counterclockwise order from q'' to q', above the segment $[p', p'']$, until the first vertex v_j of $\mathbf{R}$ is encountered. Since v_j is above $[p', p'']$ and to the left of p', the point p does not see v_j and $j \leq i$.

If $[r, p', p'']$ forms a left turn we can make a symmetric argument to show that there is a vertex of $\mathbf{R}$ to the left of p not seen by p.

We have thus proved both directions of the equivalence. □ □

Lemma 4.7 gives us a method to compute the kernel expansion of a region. We begin by ordering the vertices of the region from left to right. For each vertex, in order, we compute the visibility polygon [15, 22, 25] and establish the appropriate intersections in successive order; see Figure 17. The complexity of the algorithm is $O(m \log m + mn)$, where m is the number of vertices of $\mathbf{R}$ and n is the number of vertices of $\mathbf{P}$. If $\mathbf{R}$ is monotone, the complexity reduces to $O(mn)$ since the sorting of the vertices of $\mathbf{R}$ can be done in linear time.

The algorithm repeatedly computes intersections between two monotone polygons and combines the result with the left part established in previous iterations. The intersection between two monotone polygons having n and n' vertices respectively can be computed in $O(n + n')$ time with a plane sweep algorithm.

Using the linear time intersection algorithm we can successively compute the intersections between visibility polygons, obtaining the kernel expansions of the appropriate pocket regions, i.e., the spears of each type. Since the number of pocket vertices is at most linear in total, we have the following lemma.

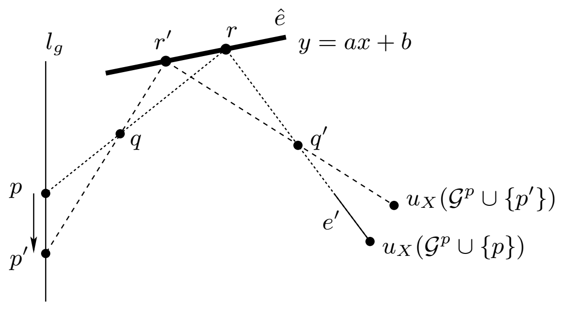
LEMMA 4.8 *A spear in a monotone polygon can be computed in quadratic time.*

4.3.2 Computing the Next Guard

Consider Step 7 of algorithm GUARD-MONOTONE-POLYGON. In an iteration, just before we reach Step 7, we have a partial guard set $\mathcal{G}^p$ with the rightmost guards at g and g' and we are supposed to place a third guard $\hat{g}$ on the same vertical line in such a way that the spear tip of type X with respect to $\mathcal{G}^p \cup \{\hat{g}\}$ is as far to the

```
Algorithm   COMPUTE-KERNEL-EXPANSION
Input:   A region $\mathbf{R}$ in a monotone polygon $\mathbf{P}$
Output: The kernel expansion $\mathbf{ke}(\mathbf{R})$
1  Order the vertices of $\mathbf{R}$ from left to right; $v_1, \ldots, v_m$
2  Let $\mathbf{K} := \mathbf{VP}_R(v_1)$
3  for each vertex $v_i$ in order from $v_2, \ldots, v_m$ do
3.1     Compute $\mathbf{VP}_R(v_i)$
3.2     Let $\mathbf{K}_L(v_i)$ and $\mathbf{K}_R(v_i)$ be the two parts of $\mathbf{K}$ to the left and right of $v_i$ respectively
3.3     Let $\mathbf{K} := \mathbf{K}_L(v_i) \cup (\mathbf{K}_R(v_i) \cap \mathbf{VP}_R(v_i))$
   endfor
4  return $\mathbf{K}$
End COMPUTE-KERNEL-EXPANSION
```

Figure 17: The algorithm for computing kernel expansions.



l_g
r'
r
$\hat{e}$
$y = ax + b$
q
q'
p
p'
e'
$u_X(\mathcal{G}^p \cup \{p'\})$
$u_X(\mathcal{G}^p \cup \{p\})$

Figure 18: Illustrating the movement of $u_X(\mathcal{G}^p \cup \{p\})$.

right as possible. Let l_g be the vertical line through g and g'. The line l_g intersects U at p_U and D at p_D. The algorithm emulates a sliding process whereby a point p slides along l_g from p_U to p_D and we maintain the spear tip $u_X(\mathcal{G}^p \cup \{p\})$ as a function of p, continuously updating the spear tip as p moves along l_g.

To accurately detect for which point p that the point $u_X(\mathcal{G}^p \cup \{p\})$ is rightmost, we let the x-coordinate of $u_X(\mathcal{G}^p \cup \{p\})$ be a function of the y-coordinate of p. We denote this function by $x_X(y)$, where y corresponds to the parameter of the vertical line $l_g = (1 - y) \cdot p_U + y \cdot p_D$.

The spear tip $u_X(\mathcal{G}^p \cup \{p\})$ is adjacent to two edges e and e' of the corresponding spear and, if $u_X(\mathcal{G}^p \cup \{p\})$ moves when p moves along l_g, at least one of these edges must move as p moves. One of the two edges, say e', extends towards the left, reaching a point r on a boundary edge $\hat{e}$ of X. The edge e, on the other hand, can either coincide with the opposite boundary $\bar{X}$ (when $u_X(\mathcal{G}^p \cup \{p\})$ lies on $\bar{X}$) or it extends towards the left, touching a vertex of the boundary $\bar{X}$ before it reaches a vertex v of a type X pocket. In the most general case, both r and v move as p moves. Consider first the movement of r on $\hat{e}$, where $\hat{e}$ is a segment on the line $y = ax + b$. The supporting segment $[p, r]$ touches X at a point q and the other supporting segment $[u_X(\mathcal{G}^p \cup \{p\}), r]$ touches X at a point q'; see Figure 18.

We want to establish the equation of the line coinciding with $[u_X(\mathcal{G}^p \cup \{p\}), r]$ in terms of the y-coordinate

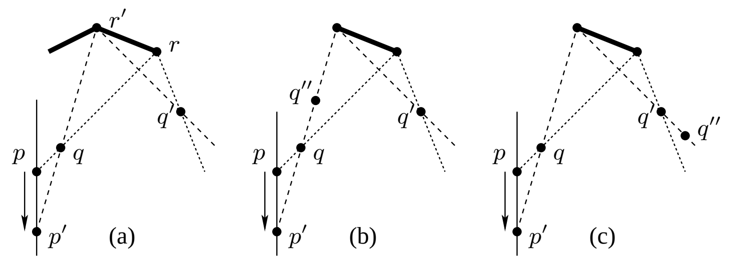


Figure 19: The cases for parameter change.

of p. If we let r be a function of p, we have that r is the intersection point between the lines $y = \frac{y(p)-y(q)}{x(p)-x(q)}x + y(q) - \frac{y(p)-y(q)}{x(p)-x(q)}x(q)$ and $y = ax + b$. So, by setting the two linear functions equal, we obtain the coordinates of the point r. The two coordinates are each the ratio between two affine functions in $y(p)$, i.e.,

$$r = \left(\frac{c \cdot y(p) + d}{y(p) + h}, \frac{c' \cdot y(p) + d'}{y(p) + h} \right),$$

where c, d, h, c', and d', are constants dependent on a, b and q. Hence, the line through r and q' can be established to be

$$y = g(y(p))x + k(y(p)),$$

where $g(y) = (\alpha y + \beta)/(y + \gamma)$ and $k(y) = (\alpha' y + \beta')/(y + \gamma)$, for constants α, β, γ, α' and β'.

With similar calculations we can establish the other supporting line that intersects the vertex v of a type X pocket to have the equation $y = g'(y(p))x + k'(y(p))$ as a function of $y(p)$. The function $x_X(y(p))$ is the x-coordinate of the intersection point between the two supporting lines, i.e.,

$$g(y)x_X(y) + k(y) = g'(y)x_X(y) + k'(y),$$

giving us

$$x_X(y) = \frac{k'(y) - k(y)}{g(y) - g'(y)} = \frac{Ay^2 + By + C}{A'y^2 + B'y + C'},$$

where the constants A, B, C, A, B' and C' only depend on the points of contact that the four supporting lines corresponding to visibility polygon edges make with the boundary.

The constant parameters A, B, C, A', B' and C' can change value as the supporting lines make contact on different vertices and edges of the polygon and pocket boundaries. We are interested in computing these points of parameter change to be able to update the function $x_X(y)$ appropriately; see Figure 19. These occur when:

- the convex vertex of an edge of $\mathbf{VP}(\mathcal{G}^p \cup \{p\})$ adjacent to a pocket becomes incident to two vertices on the polygon boundary of $\mathbf{P}$; see Figures 19(a) and (b).
- the convex vertex of an edge of $\mathbf{VP}(u_X(\mathcal{G}^p \cup \{p\}))$ adjacent to a pocket becomes incident to two vertices on the polygon boundary of $\mathbf{P}$ or to one vertex of the boundary of $\mathbf{P}$ and one vertex of a type X pocket of $\mathcal{G}^p$; see Figures 19(a) and (c).

We can establish a superset of these points on l_g that we call the *primary event points* by computing the visibility polygon of each boundary and pocket vertex to the right of l_g and obtaining the, at most two, intersection points of the visibility polygon with l_g. Hence, we have a linear number of such possible parameter changing points on l_g. Let $p_1, \ldots, p_m$ be the primary event points on l_g such that $p_i = (1 - y_i) \cdot p_U + y_i \cdot p_D$ where $x_X(y_i) = (A_i y_i^2 + B_i y_i + C_i)/(A'_i y_i^2 + B'_i y_i + C'_i)$ and A_i, B_i, C_i, A'_i, B'_i and C'_i are the parameter values of $x_X(y)$ on l_g below p_i.

Between consecutive primary event points on l_g, we can have further points of update when the visibility polygon edges e and e' adjacent to the spear tip intersect an edge e'' of a third visibility polygon at the same point. In this case, the spear tip changes from being the intersection point of e and e' to being the intersection point of either e and e'' or e' and e''. We call the points on l_g corresponding to these points of update the *secondary event points*.

For each primary event point p_i, we compute the visibility polygon $\mathbf{VP}(\mathcal{G}^p \cup \{p_i\})$. For each vertex in each pocket we compute the visibility polygon and establish the edge e'' that intersects one of e or e', if there is one. Next, we find the point on l_g where e'' intersects $u_X(\mathcal{G}^p \cup \{p\})$ when p moves from p_i to p_{i+1} on l_g. In the case when e'' is issued from a vertex v of a pocket $\mathbf{p}^X$ where v is incident to the visibility polygon $\mathbf{VP}(p)$, e'' will move, i.e., sweep in a particular direction, since v moves as p slides on l_g. In the other cases, e'' remains a fixed segment as p moves. To establish the intersection point, we solve the equations

$$x_X(y) = x'_X(y) \qquad \text{and} \qquad x_X(y) = x''_X(y),$$

where $x'_X(y)$ and $x''_X(y)$ correspond to the x-coordinates of the intersection point between e and e'' and the intersection point between e' and e'', respectively. This requires solving a quartic and two quadratic equations, giving at most four solutions for each equation, that can be computed analytically [3]. Thus, between each pair of primary event points we can have at most a linear number of secondary event points and we can determine each one of them in linear time.

From the discussion above, we know that between each pair of consecutive event points, primary or secondary, the function

$$x_X(y) = (A_i y^2 + B_i y + C_i)/(A'_i y^2 + B'_i y + C'_i); \qquad \text{with } y_i \leq y < y_{i+1}$$

does not change parameters A_i, B_i, C_i, A'_i, B'_i or C'_i. By differentiating $x_X(y)$ in this interval, we can establish the points y, and therefore also the points p on l_g, that are local maxima. This requires, in the worst case, solving one cubic equation and two quadratic equations, giving at most two solutions that can be computed analytically.

Hence, it is sufficient to compute the spear $\mathbf{sp}_X(\mathcal{G}^p \cup \{p\})$ for p being in at most $O(n^2)$ positions on l_g. From Lemma 4.8, we know that each such computation takes at most $O(n^2)$ time. The computation of Step 7 in our algorithm is dominated by the cost of computing these spears, which takes a total of $O(n^4)$ time. Since the algorithm places at most a linear number of guards in the polygon, we have the following lemma.

LEMMA 4.9 *The computation in algorithm* GUARD-MONOTONE-POLYGON *can be performed in $O(n^5)$ time.*

5 Interior Guarding Rectilinear Polygons

In this section, we give an algorithm to compute a small guard set in a simple rectilinear polygon $\mathbf{P}$. Our algorithm consists of two main steps. First, we find a subdivision of the polygon into monotone pieces, then, we use the previously given algorithm to compute a guard cover in each monotone piece. Let us assume to begin with that $\mathbf{P}$ is not starshaped, since if this is the case, the single guard can be computed in linear time with the algorithm by Lee and Preparata [27].

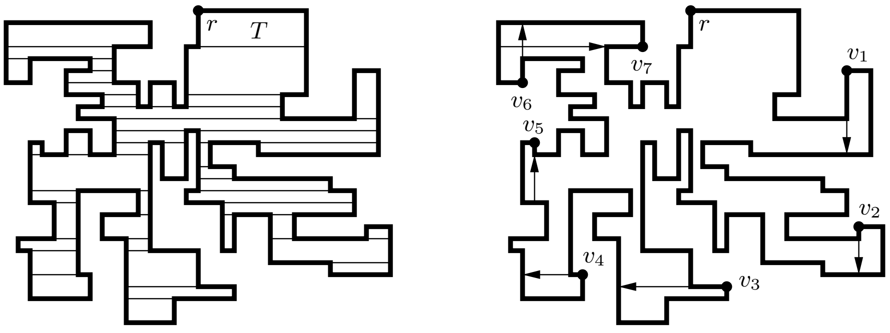


Figure 20: A trapezoidation, the root vertex, the essential extensions and vertices.

A *trapezoidation* of a polygon is a partitioning of the polygon into trapezoids, or horizontal strips, with the horizontal edges of each trapezoid connecting the vertices of the polygon; see Figure 20. Chazelle [4], as a subroutine for triangulation, shows how such a trapezoidation can be computed in linear time in a simple polygon. Since $\mathbf{P}$ is rectilinear, the trapezoidation is a partition of $\mathbf{P}$ into rectangles. Let each trapezoid (or rectangle) correspond to a node in a graph $\mathcal{T}$ and let two nodes be connected if they share a horizontal edge of the trapezoidation. It is well known that $\mathcal{T}$ is a tree, so let T be a trapezoid corresponding to a leaf in $\mathcal{T}$. Let r be one of the vertices of T not adjacent to the neighboring trapezoid of T; see Figure 20. We call r the *root vertex*.

To every reflex vertex v in $\mathbf{P}$ we can associate two *extensions*, i.e., the two maximal line segments in $\mathbf{P}$ through v and collinear to the two edges adjacent to v. We associate a direction to an extension e collinear to an edge e_v by giving e the same direction as e_v gets when $\mathbf{P}$ is traversed in counterclockwise order. This allows us to refer to the regions to the left and right of an extension, meaning to the left or right of e if e is directed upwards. Let $\mathbf{P}_e^l$ denote the part of $\mathbf{P}$ to the left of e and $\mathbf{P}_e^r$ to the right. We say that e is a *visibility extension* if the root vertex r is in $\mathbf{P}_e^r$.

The visibility extensions capture visibility information in the sense that not all guards can be to the right of a visibility extension. Hence, every visibility extension has at least one guard to the left of it.

We say that an extension e *dominates* another extension e', if $\mathbf{P}_e^l$ is properly contained in $\mathbf{P}_{e'}^l$.

DEFINITION 5.1 *A visibility extension e is essential, if e is not dominated by any other visibility extension.*
Using the algorithm of Chin and Ntafos [6] in conjunction with Chazelle's triangulation algorithm [4], we can efficiently compute the essential extensions. Assume that this computation gives us k essential extensions, e_i, for $1 \leq i \leq k$; see Figure 20.

An essential extension e_i is collinear to an edge with one reflex and one convex vertex.

DEFINITION 5.2 *Let v_i denote the convex vertex of the edge collinear to the essential extension e_i. We call the convex vertices v_i, for $1 \leq i \leq k$, the essential vertices; see Figure 20.*
de Berg [10] shows how to construct a data structure to obtain the shortest rectilinear path between any pair of points in $\mathbf{P}$. Using this structure, we compute, for $1 \leq i \leq k$, the shortest rectilinear paths $SP_R(r, v_i)$ from r to each v_i efficiently. The paths $SP_R(r, v_i)$ will be used to help constructing a subdivision of the polygon into monotone pieces.

To each rectilinear path $SP_R(r, v_i)$ connecting r with v_i we define a vertical and a horizontal histogram expansion.

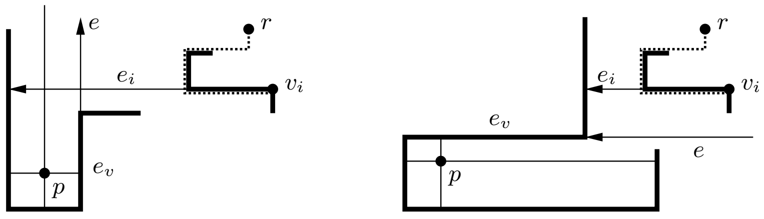


Figure 21: Illustrating the proof of Lemma 5.1.

DEFINITION 5.3 *The horizontal histogram expansion $\mathbf{H}_i^H$ consists of those points in $\mathbf{P}$ that can be connected to the path $SP_R(r, v_i)$ with vertical line segments contained in $\mathbf{P}$. We define the vertical histogram expansion $\mathbf{H}_i^V$ in a corresponding manner.*
Although the union of histogram expansions do not cover the whole polygon $\mathbf{P}$; see Figure 24; we can show that each point in $\mathbf{P}$ is seen by some path $SP_R(r, v_i)$.

LEMMA 5.1 *Every point in $\mathbf{P}$ is seen by some point of a shortest rectilinear path $SP_R(r, v_i)$.*

PROOF: Let p be a point in $\mathbf{P}$. Through p we can draw maximal vertical and horizontal line segments interior to $\mathbf{P}$, dividing $\mathbf{P}$ into four quadrants around p. (If p is on the boundary, some quadrants may be degenerate.) If the root vertex r is in one quadrant and there is an essential vertex v_i in a different quadrant, then $SP_R(r, v_i)$ crosses one of the segments through p, and hence, p sees some point of $SP_R(r, v_i)$.

Otherwise, the root vertex r and all the essential vertices lie in the same quadrant. Without loss of generality, we can assume that it is the upper right quadrant. This means that p lies in a left polygon $\mathbf{P}_{e_i}^l$ with essential vertex v_i.

Assume that p does not see $SP_R(r, v_i)$. On the boundary of $\mathbf{P}_{e_i}^l$ in the upper right quadrant of p there is at least one edge e_v adjacent to a reflex vertex v' such that its associated extension e has the root vertex r in $\mathbf{P}_e^r$. Let v be the other edge of e_v. The extension e is a visibility extension, since r lies in $\mathbf{P}_e^r$, and it is not dominated by any other visibility extension, since all essential vertices lie in $\mathbf{P}_e^r$. This gives us a contradiction since v, by definition, is an essential vertex and p sees v; see Figure 21. □ □

A histogram expansion can be computed in linear time using an algorithm by Levcopoulos [28]. Each horizontal histogram expansion consists of a number of x-monotone polygons with the property that no guard in one monotone polygon can see anything in any of the others. Furthermore, a guard outside the horizontal histogram expansion can see into at most two of the x-monotone polygons in the horizontal histogram expansion. Similarly, a vertical histogram expansion subdivides into y-monotone pieces with the same properties; see Figure 22.

LEMMA 5.2 *If $\mathbf{P}$ can be guarded with OPT guards, then a histogram expansion can also be guarded with at most $2OPT$ guards interior to the histogram expansion.*

PROOF: Let p be a point that sees into a monotone piece $\mathbf{R}$ of a histogram expansion $\mathbf{H}$. Assume that $\mathbf{R}$ is x-monotone and that p lies in a region adjacent to the lower boundary D of $\mathbf{R}$. Let l_p be the line segment that separates $\mathbf{R}$ from the piece containing p. Consider the intersection $\mathbf{VP}(p) \cap \mathbf{R}$. The intersection subdivides $\mathbf{R}$ into left pockets and right pockets. Traversing the boundary of $\mathbf{VP}(p)$ clockwise starting at a point outside $\mathbf{R}$

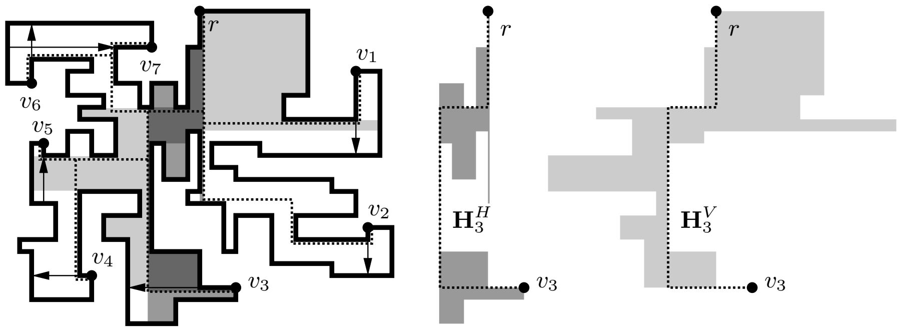


Figure 22: Illustrating the algorithm.

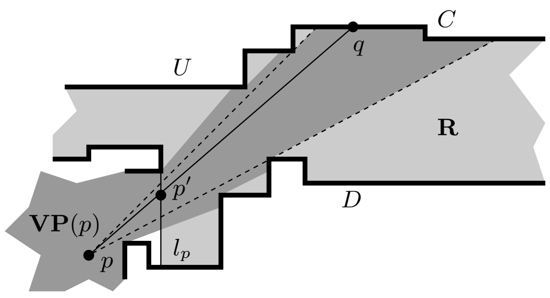


Figure 23: Illustrating the proof of Lemma 5.2.

will first reach the edges that are incident to left pockets, then a boundary chain C of $\mathbf{R}$, and finally the edges that are incident to right pockets of $\mathbf{R}$. Take any point q of C and let p' be the intersection of the line segment between p and q with l_p. Any point in $\mathbf{R}$ seen by p will also be seen by p'.

Hence, any guard outside $\mathbf{R}$ that sees points in $\mathbf{R}$ can be moved to the boundary of $\mathbf{R}$ and it sees at least as much of $\mathbf{R}$ as in its previous location.

Consider a set of OPT guards that cover $\mathbf{P}$ and let $\mathbf{R}_1, \dots \mathbf{R}_h$ be the h monotone pieces of the histogram expansion $\mathbf{H}$ computed from $SP_R(r, v_i)$. Let $\mathcal{H}$ be the subset of guards that see points in $\mathbf{H}$. Each guard in $\mathcal{H}$ outside $\mathbf{H}$ can see into at most two monotone pieces $\mathbf{R}_j$ and $\mathbf{R}_{j+1}$ that are consecutive along the path $SP_R(r, v_i)$ and by our previous argument two copies can be placed on each boundary of $\mathbf{R}_j$ and $\mathbf{R}_{j+1}$. This gives a new guard set $\mathcal{H}'$ consisting of the guards in $\mathcal{H}$ inside $\mathbf{H}$ and the ones copied and moved to the boundary of $\mathbf{H}$. By our argument, $\mathcal{H}'$ sees all of $\mathbf{H}$ and contains at most $2|\mathcal{H}| \leq 2OPT$ guards. □ □

We use the GUARD-MONOTONE-POLYGON algorithm of the previous section to guard each monotone piece with at most $O(m)$ guards, where m is the smallest guard cover for the monotone piece. From Lemma 5.2 we know that each histogram expansion can be guarded with $2OPT$ guards interior to the histogram expansion, and hence, our algorithm guards it with at most $O(OPT)$ guards.

LEMMA 5.3 *No point in* $\mathbf{P}$ *sees more than two essential vertices.*

PROOF: Any point p that sees an essential vertex v_i lies in $\mathbf{P}^l_{e_i}$, e_i being the essential extension associated to v_i. Assume that e_i is vertical, then no other vertical essential extension e_j can have p to the left. This is because, if

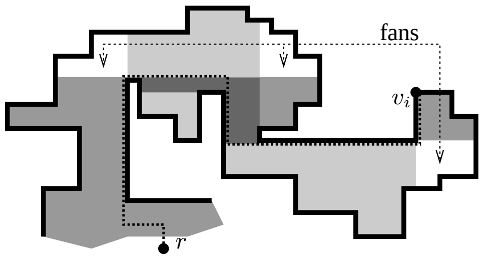


Figure 24: Illustrating the proof of Lemma 5.4.

p is to the left of both e_i and e_j then, either $\mathbf{P}^l_{e_i}$ is contained in $\mathbf{P}^l_{e_j}$ or $\mathbf{P}^l_{e_j}$ is contained in $\mathbf{P}^l_{e_i}$, contradicting that both e_i and e_j are essential extensions.

By a similar argument p cannot be to the left of more than one horizontal essential extension. Hence, p can see at most two essential vertices. □ □

By the preceeding lemma, one guard can see at most two essential vertices, hence, $k/2 \leq OPT$. Since we construct a total of $2k$ horizontal and vertical histogram expansions, the union $\bigcup_{i=1}^{k} \mathbf{H}_i^H \cup \mathbf{H}_i^V$ can be guarded by at most $O(OPT^2)$ guards.

The set $\mathbf{P} \setminus (\bigcup_{i=1}^{k} \mathbf{H}_i^H \cup \mathbf{H}_i^V)$ partitions into a number of connected regions that we call *fans*. We show that each of these fans is starshaped.

LEMMA 5.4 *A fan is always starshaped.*

PROOF: The boundary of a fan consists of the vertical segment of an x-monotone piece from a horizontal histogram expansion and a horizontal segment of a y-monotone piece from a vertical histogram expansion forming a 90 degree wedge connected at a point q; see Figure 24.

The two wedge edges of the fan are connected by a part of the boundary of $\mathbf{P}$. Consider any point p properly in the interior of the fan, i.e., not on the boundary. A vertical or horizontal line through p does not intersect any path $SP_R(r, v_i)$. Therefore, the boundary part of the fan in common with the boundary of $\mathbf{P}$ must be monotone with respect to both the x- and y-axes. The fan is thus completely visible from q, and hence, starshaped. □ □

Since the fans are starshaped, we can guard each of them with one extra guard. A fan is also adjacent to one horizontal and one vertical histogram expansion and each monotone piece in a histogram expansion can be adjacent to at most two fans. Hence, to count the number of fans, i.e., the number of additional guards we have to place to cover the complete polygon, we associate each fan with the horizontal or vertical monotone piece of a histogram expansion that is closer to the root vertex r. Since each monotone piece must contain at least one guard, the number of guards placed to see all of $\mathbf{P}$ has at most doubled.

We have proved the following theorem.

THEOREM 3 *There is a deterministic polynomial time algorithm that computes a guard cover of size $O(OPT^2)$ in a rectilinear polygon $\mathbf{P}$, where OPT is the size of the smallest guard cover for $\mathbf{P}$.*

Combining this result with another guarding algorithm gives us an approximation result.

THEOREM 4 *There is a deterministic polynomial time algorithm that computes a guard cover with approximation factor $O(\sqrt{n})$ in a rectilinear polygon $\mathbf{P}$.*

PROOF: We run the algorithm that we have developed above and the classical algorithm by Fisk [17] in conjunction and return the smallest of the two guard covers obtained. Let k_R be the size of the guard cover returned by our algorithm and let k_F be the size of the cover returned by Fisk's algorithm. Fisk proves that $k_F \leq \lfloor n/3 \rfloor$.

To calculate the approximation ratio, assume first that $k_R \leq k_F \leq n/3$ then, since $k_R \in O(OPT^2)$, we have that $OPT \geq c\sqrt{k_R}$, for some constant c. The ratio becomes

$$\frac{k_R}{OPT} \leq \frac{k_R}{c\sqrt{k_R}} = \frac{\sqrt{k_R}}{c} \leq \frac{\sqrt{n/3}}{c} \in O(\sqrt{n}).$$

On the other hand, if $k_R > k_F$, we have that $OPT \geq c\sqrt{k_R} > c\sqrt{k_F}$ and the ratio then becomes

$$\frac{k_F}{OPT} \leq \frac{k_F}{c\sqrt{k_F}} = \frac{\sqrt{k_F}}{c} \leq \frac{\sqrt{n/3}}{c} \in O(\sqrt{n}).$$

□ □

6 Conclusions

We have proved that vertex guarding a monotone polygon is NP-hard. We have also constructed two polynomial time deterministic algorithms for guarding. One for approximate interior guarding of monotone polygons and one for approximate guarding of rectilinear polygons. Our contribution is that the approximation factors for both algorithms are independent of the size of the polygon.

Interestingly, King and Krohn have generalized the NP-hardness proof to vertex guarding two-dimensional monotone terrains [24].

Other open problems are to improve the approximation bounds for monotone and rectilinear polygons, to find approximation algorithms for other classes of polygons, and ultimately approximate guarding of simple polygons in general.

We wish to thank the anonymous referees for their efforts in proof-reading earlier versions of this article. Their comments have helped us improve the presentation considerably.

References

[1] A. AGGARWAL. *The Art Gallery Theorem: Its Variations, Applications and Algorithmic Aspects.* PhD thesis, Johns Hopkins University, 1984.

[2] B. BRODÉN, M. HAMMAR, B.J. NILSSON. Guarding Lines and 2-Link Polygons is APX-hard. In *Proc. 13th Canadian Conference on Computational Geometry, CCCG'01*, pages 45–48, 2001.

[3] G. CARDANO. *Artis Magnæ, Sive de Regulis Algebraicis Liber Unus*, 1545. (English translation reprinted by Dover Publications in 1993 as *Ars Magna or The Rules of Algebra*).

[4] B. CHAZELLE. Triangulating a Simple Polygon in Linear Time. In *Proc. 31st Symposium on Foundations of Computer Science*, pages 220–230, 1990.

[5] D.Z. CHEN, V. ESTIVILL-CASTRO, J. URRUTIA. Optimal Guarding of Polygons and Monotone Chains. In *Proc. 7th Canadian Conference on Computational Geometry, CCCG'95*, pages 133–138, 1995.

[6] W. CHIN, S. NTAFOS. Optimum Watchman Routes. *Information Processing Letters*, 28:39–44, 1988.

[7] V. CHVÁTAL. A Combinatorial Theorem in Plane Geometry. *Journal of Combinatorial Theory B*, 13(6):395–398, 1975.

[8] K.L. CLARKSON, K. VARADARAJAN. Improved Approximation Algorithms for Geometric Set Cover. In *Proc. 21st ACM Symposium on Computational Geometry*, 2005.

[9] J.C. CULBERSON, R.A. RECKHOW. Covering Polygons is Hard. In *Proc. 29th Symposium on Foundations of Computer Science*, pages 601–611, 1988.

[10] M. DE BERG. On Rectilinear Link Distance. *Computational Geometry: Theory and Applications*, 1(1):13–34, 1991.

[11] A. DESHPANDE, T. KIM, E.D. DEMAINE, S.E. SARNA. A Pseudopolynomial Time $O(\log n)$-Approximation Algorithm for Art Gallery Problems. In *Proc. 10th Workshop on Algorithms and Data Structures, WADS'07*, volume 4619 of *Lecture Notes in Computer Science*, pages 163–174. Springer-Verlag, 2007.

[12] A. EFRAT, S. HAR-PELED. Guarding Galleries and Terrains. *Information Processing Letters*, 100:238–245, 2006.

[13] S. EIDENBENZ. Inapproximability Results for Guarding Polygons without Holes. In *Proc. 9th Annual International Symposium on Algorithms and Computation*, pages 427–436, 1998.

[14] S. EIDENBENZ. *Inapproximability of Visibility Problems on Polygons and Terrains*. PhD thesis, ETH, Zurich, 2000.

[15] H. ELGINDY, D. AVIS. A Linear Algorithm for Computing the Visibility Polygon from a Point. *J. Algorithms*, 2:186–197, 1981.

[16] L. FEJES TÓTH. Illumination of Convex Discs. *Acta Mathematica Academiae Scientiarum Hungaricae*, 29:355–360, 1977.

[17] S. FISK. A Short Proof of Chvátal's Watchman Theorem. *Journal of Combinatorial Theory B*, 24:374, 1978.

[18] M.R. GAREY, D.S. JOHNSON. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, 1979.

[19] S.K. GHOSH. Approximation Algorithms for Art Gallery Problems. In *Proceedings of the Canadian Information Processing Society Congress*, 1987.

[20] J.E. GOODMAN, J. O'ROURKE, editors. *Handbook of Discrete and Computational Geometry*. CRC Press, 1997.

[21] F. HOFFMANN, M. KAUFMANN, K. KRIEGEL. The Art Gallery Theorem for Polygons with Holes. In *Proc. 32nd IEEE Symposium on the Foundations of Computer Science*, pages 39–48, 1991.

[22] B. JOE, R.B. SIMPSON. Correction to Lee's Visibility Polygon Algorithm. *BIT*, 27:458–473, 1987.

[23] M.J. KATZ, G.S. ROISMAN. On Guarding Rectilinear Domains. In *Proc. 10th Scandinavian Workshop on Algorithm Theory*, volume 4059 of *Lecture Notes in Computer Science*, pages 220–231. Springer-Verlag, 2006.

[24] J. KING, E. KROHN. Terrain Guarding is NP-hard. In *Proc. 21st ACM-SIAM Symposium on Discrete Algorithms, SODA'10*, pages 1580–1593, 2010.

[25] D.T. LEE. Visibility of a Simple Polygon. *Computer Vision, Graphics, and Image Processing*, 22:207–221, 1983.

[26] D.T. LEE, A.K. LIN. Computational Complexity of Art Gallery Problems. *IEEE Transactions on Information Theory*, IT-32:276–282, 1986.

[27] D.T. LEE, F.P. PREPARATA. An Optimal Algorithm for Finding the Kernel of a Polygon. *Journal of the ACM*, 26:415–421, 1979.

[28] C. LEVCOPOULOS. *Heuristics for Minimum Decompositions of Polygons*. PhD thesis, University of Linköping, Linköping, Sweden, 1987.

[29] C. LEVCOPOULOS, A. LINGAS. Covering Polygons with Minimum Number of Rectangles. In *Proc. 1st Symposium on the Theoretical Aspects of Computer Science*, volume 166 of *Lecture Notes in Computer Science*, pages 63–72. Springer-Verlag, 1984.

[30] B.J. NILSSON. *Guarding Art Galleries — Methods for Mobile Guards*. PhD thesis, Lund University, 1995.

[31] B.J. NILSSON. Approximate Guarding of Monotone and Rectilinear Polygons. In *Proc. 32nd International Colloquium on Automata, Languages, and Programming, ICALP'2005*, volume 3580 of *Lecture Notes in Computer Science*, pages 1362–1373. Springer Verlag, 2005.

[32] J. O'ROURKE. *Art Gallery Theorems and Algorithms*. Oxford University Press, 1987.

[33] J.R. SACK, J. URRUTIA, editors. *Handbook on Computational Geometry*. Elsevier Science Publishers, 1999.

[34] T.C. SHERMER. Recent Results in Art Galleries. *Proceedings of the IEEE*, pages 1384–1399, September 1992.