M. Tech. (Computer Science) Dissertation Series

Art Gallery Problem for Monotone Polygons

a dissertation submitted in partial fulfillment of the
requirement for the M. Tech. (Computer Science)
degree of the Indian Statistical Institute

By

Diptapriyo Majumdar
Roll No: CS1107

under the supervision of

Prof. Subhas C. Nandy
Advanced Computing and Microelectronics Unit

INDIAN STATISTICAL INSTITUTE
203, Barackpore Trunk Road
Calcutta - 700108

Acknowledgment

At the end of this course, it is my pleasure to thank everyone who has helped me along the way.

First of all, I want to express my sincere gratitude to my supervisor, Prof. Subhas C. Nandy for introducing me to the world of Computational Geometry and giving me interesting problems. I have learnt a lot from him. For his patience, for all his advice and encouragement and for the way he helped me to think about problems with a broader perspective, I will always be grateful.

I would like to thank all the professors at ISI Kolkata who have made my educational life exciting and helped me to gain a better outlook on Computer Science. I would also like to express my gratitude to Prof. Sandip Das, Prof. Arijit Bishnu, Prof. Gautam K. Das for interesting discussions.

I would like to thank everybody at ISI for providing me a nice platform for pursuing my studies. I thank all my classmates who have made the academic and non-academic experience very delightful. Special thanks to my friends Sukanta-da, Minati-di, Aniket-da, Soumen-da, Aritra-da, Soumyottam-da, Sayan, Girish, Sunny, Hirak and many others who made my campus life so enjoyable. It has been great to have them around at all times.

My most important acknowledgement goes to my family and friends who have filled my life with happiness, most significantly, to my parents who have always encouraged me to pursue my passions and instilled a love of knowledge in me. I am indebted to my friends Suman, Raviteja, Priyanka, Bipul, Sanchita, Arani, Abhishek, Bipin for their endless supply of encouragement, moral support and entertainment.

Contents

Chapter 1

Introduction

Art Gallery problem is one of the well known problems in Computational Geometry. Given a polygon P in R^2, we are asked to find the minimum number of points interior of P to guard the entire polygon P. Our study is on a restricted version of the problem, where the given polygon P is monotone with respect to x-axis. Here also different variations may be studied, namely point guarding, vertex guarding and edge guarding. Vertex Guarding deals with the case when guards are placed only at vertices of the polygon. Edge guarding deals with the case when guards are placed only at the boundary of the polygon. Point guarding deals with the case when guards can be placed anywhere inside the polygon. It is a restricted version of the Set Cover problem which is known to be NP Complete and can not be approximated to a constant approximation factor unless P = NP. For any simple polygon, point guarding problem can be formulated to set cover as given in [15]. Initially Chen *et al*[6] proved vertex guarding to be NP Hard. But, their proof is still omitted and is under verification. After that, Erik Krohn and Bengt J. Nillson has proved its vertex guarding of monotone polygon to be NP Hard in [2]. But, its interior guarding does not immediately follow from that claim. The same authors Erik Krohn and B. J. Nillson[7] have proved its interior guarding version to be NP Hard. It has a related problem which deals with guarding a terrain. Guarding a terrain is also NP Hard. Erik Krohn and James King [8] gave a proof of that. About guarding interior of a polygon, we know some basic results[1] that n/3 guards are always sufficient and occasionally necessary to guard a polygon. In Chapter 3, we give an approximation algorithm to guard a terrain and briefly describe other works related to it referring them. In Chapter 4, we have discussed interior guarding of monotone polygon. First we describe that a monotone polygon can be guarded with minimum number of guards when the polygon is y-monotone and also axis parallel(also called as horizontally convex). Also,

we have discussed a constant factor approximation algorithm given by Bengt J. Nillson[5] when the polygon is x-monotone. It provides approximation factor 12. After that we propose an algorithm for a special sub-case when the polygon is x-monotone and also their two extreme points are mutually visible to each other. In that algorithm, we have conjectured that it is expected to give a 4-factor approximation algorithm. We have given a brief informal justification why it should give 4-factor approximation algorithm.
In this thesis, we have reviewed these following works in detail. These are 4-factor approximation algorithm for terrain guarding problem[10], algorithm to guard monotone orthogonal polygon[14] and constant factor algorithm for monotone polygon given by Bengt J. Nillson[5]. And finally we have provided our approach in the following sections of Chapter 4. In Section 4.1, we have given an approach where the input polygon in uni-monotone and in Section 4.4, for a special sub-case of the x-monotone polygon.

Chapter 2

Preliminaries

2.1 Monotone Polygons and Terrains

:

Definition (Monotone Polygon) A polygon P is said to be monotone with respect to a given line ℓ if any line perpendicular to ℓ intersects that polygon into at most two vertices. So, if any vertical line intersects P into at most two vertices then P is said to be x-monotone. Therefore every x-monotone polygon has an upper chain and a lower chain. In both the chains, the vertices are stored in increasing order of x co-ordinates. Therefore an x-monotone polygon has two extreme points **s** and **t** having the minimum and maximum co-ordinates respectively.

Definition (1.5D Terrain) Usually when we use *terrain* we generally use a special version of it which is 1.5D Terrain. It is an x-monotone chain T consisting of a set of points $p_1(x_1, y_1), p_2(x_2, y_2), ..., p_n(x_n, y_n)$ where (p_i, p_{i+1}) are connected by a line segment, $i = 1, 2, ..., n-1$ and the line segment joining p_1 and p_n does not intersect the chain.

Definition (Visibility Polygon) Given a polygon P and an interior point q, the visibility polygon of the point q is said to be the area inside the polygon P such that if a point p in this area is taken then the line segment joining p and q does not intersect the boundary of P.

2.2 Approximation Algorithm

As we focus on approximation algorithm for the Art Gallery problems, we now define various types of approximation schemes. Even though for our work we do not use all the schemes as PTAS is available for Art Gallery problem.

Definition (Approximation Algorithm) Let P be a maximization (respectively minimization) problem. Then an algorithm A is said to be an α-factor approximation algorithm for P if and only if for any instance A(x) of P, A(X) runs in polynomial in —X— time and returns a feasible solution SOL(X) such that $SOL(X) \geq \alpha * OPT$ (respectively $SOL(X) \leq \alpha * OPT$) where OPT denotes the optimum solution for the problem P for the given instance X.

Definition (Polynomial Time Approximation Scheme) Let P be a maximization (respectively minimization) problem. Algorithm A is a polynomial-time approximation scheme(PTAS) for P if and only if for any instance X of P and for any (fixed) $\epsilon > 0$, $A(X; \epsilon)$ runs in time polynomial in $|X|$ and delivers a feasible solution $SOL(X, \epsilon)$, such that $SOL(X, \epsilon) \geq (1 + \epsilon)OPT$ (respectively $SOL(X, \epsilon) \leq (1 - \epsilon)OPT$).

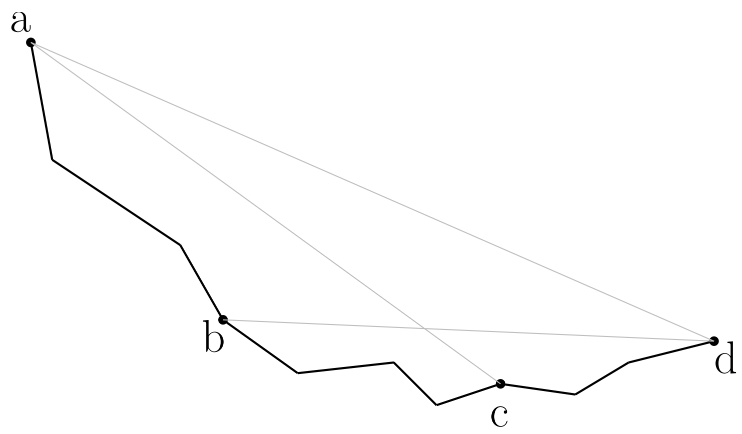
Chapter 3

Approximation Algorithms for Terrain Guarding

We have already seen that a terrain is said to be a monotone chain. We consider the case when the chain is x-monotone. A chain x-monotone indicates that their x-coordinates are in increasing order. Finding an optimal set of guards was not known to be NP Hard before [5]. Neither attempt to prove its NP Completeness nor attempt to find a polynomial time algorithm has been successful for that before James King and Erik Krohn gave a hardness proof of that in [8]. The first constant factor approximation algorithm has been given by Ben-Moshe in [3]. Clarkson and Varadarajan gave another constant factor approximation for the problem in [4] based on solving a linear programming relaxation and rounding. No attempt has been made to minimize the constant factor in either paper. King finds in [12] that the factor in [3] can be brought down to 6 with some modifications into the algorithm. James King in [12] has provided a 5-approximation factor. Later on K.Elbassoioni, E. Krohn[10] gave a 4-approximation algorithm for terrain guarding problem. Matt Gibson, Gaurav Kanade, Erik Krohn and Kasturi Varadarajan in [11] have provided an approximation scheme that uses local search technique.

3.1 4-factor approximation algorithm

Till now it is the best known approximation algorithm for guarding a terrain. Before going to the algorithm we need to point out some few observations. Vertices are arranged in increasing order of x-coordinates. We denote this notation $a < b$ to indicate that a is to the left of b. We say that a sees b if the line segment $\bar{ab}$ lies on or above the terrain. We have the following observation.



Lemma 3.1.1 *Four points $a < b < c < d$. If a sees c and b sees d, then a sees d.*

Proof We know that a sees c. So, there is no point in terrain that lies above the line segment ac. Also, no point lies above the segment bd. So, b must be below ac and c must be below bd. Therefore there can not be any point in the terrain that lies above the segment ad. Hence a sees d.

3.1.1 LP based Algorithm for one sided guarding and its approximation factor:

We consider the one sided guarding version. In this version, the guards can see in only one of the two directions, left or right. Specially given three set of points T, G_L, G_R, we want to find sets $B_L \subset G_L$ and $B_R \subset G_R$ such that for all $p \in T$, there is $g \in B_L$ such that $g < p$ and g sees p or $g \in B_R$ such that $g > p$ and g sees p.
Let us denote $S_L(p) = \{g \in G_L | g \text{ sees } p\}$ and $S_R(p) = \{g \in G_R | g \text{ sees } p\}$. While solving left-guarding(respectively right-guarding), we set $G_L = G$(respectively $G_R = G$). Now for the left guarding, we can map this problem as an integer Linear Programming problem as follows:

minimize $\sum_{g \in G} x_g$
subject to $\sum_{g \in S_L(p)} x_g \geq 1 \; \forall p \in T$
$x_g \in \{0, 1\} \; \forall g \in G$

Denote the above formulation as LP1.

Without loss of generality, we assume that each point in T can be seen by a guard on its left or by a guard on its right. $x_{g,L}$(respectively $x_{g,R}$) represents the indicator variable that can take value from $\{0,1\}$. $x_{g,L} = 0$(respectively $x_{g,R} = 0$) indicates that g is not chosen as a guard in G_L(respectively in G_R). On the other hand $x_{g,L} = 1$(respectively $x_{g,R} = 1$) indicates that g is chosen as a guard in G_L(respectively in G_R). Now in order to find the optimal set of left and right guards, we can map this problem as an integer Linear Programming problem as follows:

$$\begin{array}{ll} \text{minimize} & \sum_{g \in G_L} x_{g,L} + \sum_{g \in G_R} x_{g,R} \\ \text{subject to} & \sum_{g \in G_L \cap S_L(p)} x_{g,L} + \sum_{g \in G_R \cap S_R(p)} x_{g,R} \geq 1 \ \forall p \in T \\ & x_{g,L} \in \{0,1\} \ \forall g \in G_L \\ & x_{g,R} \in \{0,1\} \ \forall g \in G_R \end{array}$$

We know that Integer Linear Programming problem is NP Hard. So, we relax the integrality of the variables, and assume that these can take any real number in $[0,1]$. Thus, the problem is

$$\begin{array}{ll} \text{minimize} & \sum_{g \in G_L} x_{g,L} + \sum_{g \in G_R} x_{g,R} \\ \text{subject to} & \sum_{g \in G_L \cap S_L(p)} x_{g,L} + \sum_{g \in G_R \cap S_R(p)} x_{g,R} \geq 1 \ \forall p \in T \\ & x_{g,L} \geq 0 \ \forall g \in G_L \\ & x_{g,R} \geq 0 \ \forall g \in G_R \end{array}$$

Algorithm first finds an optimal fractional solution x^*. Guided by x^*, we divide the points into two sets

$$T_L = \{p \in T | \sum_{g \in S_L(p) \cap G_L} x^*_{g,L} \geq 1/2\}$$

$$T_R = \{p \in T | \sum_{g \in S_R(p) \cap G_R} x^*_{g,R} \geq 1/2\}$$

$\sum_{g \in G_L} x^*_g \leq OPT$ as OPT is the optimal solution of the integer program. But, $\sum_{g \in G_L} x^*_g$ is the optimal solution of the linear program.

Lemma 3.1.2 *Let B^*_L and B^*_R be the optimal solutions for the pairs (T_L, G_L) and (T_R, G_R) respectively. Then $|B^*_L| \leq 2\sum_{g \in G_L} x^*_g$ and $|B^*_R| \leq 2\sum_{g \in G_R} x^*_g$.*

Proof Set $x_{g,L} = 2x_g^*$ we get a fractional solution for the linear program to guard T_L. Hence, its cost is $\sum_{g \in G_L} x_{g,L} \leq 2 \sum_{g \in G_L} x_g^* \leq 2OPT$.
Other inequality $\sum_{g \in G_L} x_{g,R} \leq 2 \sum_{g \in G_R} x_g^* \leq 2OPT$ also holds symmetrically. Thus we have a 4-factor approximation. Finally we prove the following theorem.

Theorem 3.1.3 *There exists a polynomial time algorithm for terrain guarding that provides 4-factor approximation.*

Chapter 4

Algorithms to Guard Monotone Polygons

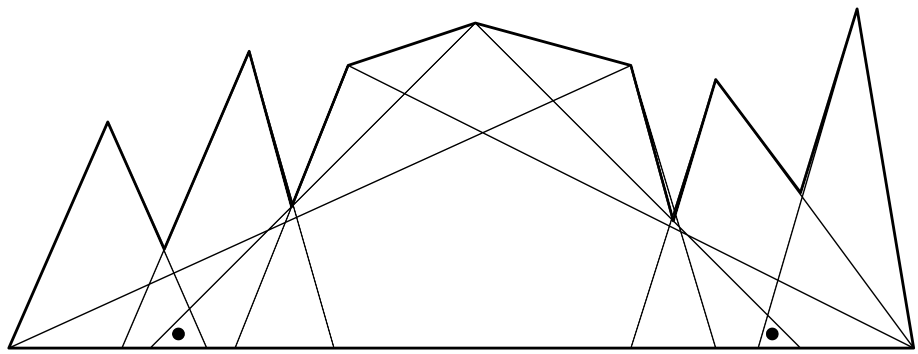
Art Gallery problem was for vertex guarding was not known to be NP Hard for Monotone polygon until E.Krohn and B.J.Nillson[2] proved it to be NP Hard. NP Hardness of the interior guarding does not immediately follow from that claim. However, Erik Krohn and B.J.Nillson[7] gave the NP Hardness proof of its interior guarding.

4.1 Guarding an Uni-monotone Polygon

A polygon is said to be uni-monotone with respect to x-axis if its upper chain is a x-monotone chain and the lower chain is the x axis. In an analogous way we can define that a polygon is said to be uni-monotone with respect to a line ℓ, if one of its chain is l-monotone and the other chain is the line ℓ.

4.1.1 Algorithm

Consider the sequence of convex pieces of the upper chain $C_1, C_2, \ldots, C_k$ arranged in order, where each piece consists of at least 3 vertices among which the first and last vertices are reflex, and $\cup_{i=1}^{k} C_i$ consists of all the vertices of the terrain. The projection for each edge e_i on the on the line ℓ is an interval $[a_i, b_i]$ (the portion at the line ℓ from where e_i is visible)where a_i and b_i are the points of intersection of the extensions of the first and last edges of C_i with the line ℓ. Note that, from the interval $[a, b]$, the entire C_i is visible. Thus, the art-gallery problem of uni-monotone polygon can be formulated as finding the minimum clique cover of an interval graph with the set of intervals $\{[a_i, b_i], i = 1, 2, \ldots, \}$ on the line ℓ. This can be solved in



Guarding an uni-monotone polygon

Figure 4.1: Figure

$O(n + k \log k)$ time, where n is the number of vertices in the uni-monotone polygon.

4.2 Guarding Monotone Orthogonal Polygons

Here, we need to cover the monotone orthogonal polygon into minimum number of orthoconvex polygons. Each orthoconvex polygon can be guarded by a single guard. Gewali et al.[13] and Lingas et al.[14] separately proposed linear time algorithm for this problem. Thus, this problem can be solved in $O(n)$ time. Finally, after getting the orthoconvex partitioning, the guard placement scheme is given below.

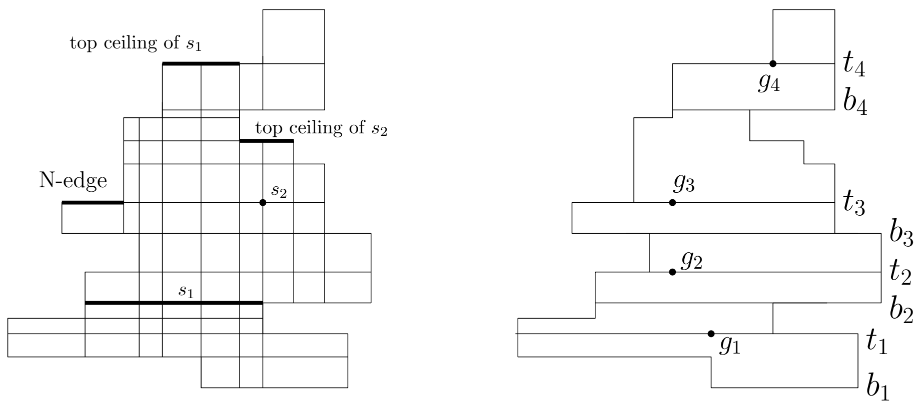


Figure 4.2: Guarding Orthogonal Monotone Polygon

The basic approach for positioning guards:
First, let us briefly recall the idea of Gewali et al.'s algorithm. Let P(i) be

the sub-polygon of P consisting of the portion of P that lies below the i'th grid line. The idea of the algorithm is to perform a plane sweep, moving up row by row. A new guard g is placed in P whenever the sweep line reaches a horizontal grid line i such that the sub-polygon P(i) contains an uncovered portion. The new guard is positioned at a grid point so that the current covering set covers *P(i)* and as many additional consecutive rows of cells above the line i as is possible. Formally, the idea of the algorithm is as follows.

$GUARD - PLACEMENT[P]$

- G is initially empty.

- for all $i = 1, 2, ..., n$
  If $P(i)$ is not covered by G then a new guard g is added to G. The new guard is added on a vertical grid segment that intersects the $(i-1)$st horizontal grid segment. In particular, the new guard g is placed on any of these vertical grid segments that allow g to maximize $Ymax(g)$, the level of the top ceiling of g. And the guard g is positioned on the highest horizontal grid line j $(j \geq i)$ such that when this point is added to G, the new set covers $P(j)$.

- Return the final set G as an optimal guard placement for P.

4.3 Constant factor Approximation Algorithm for x-monotone polygons

In this section, we discuss a constant factor approximation algorithm for the art gallery problem of a x-monotone polygon. This algorithm has been provided by Bengt J. Nillson in full detail[9].

4.3.1 Terminology and Notation

An x-monotone polygon polygon is bounded by two x-monotone chains, namely an upper chain U and a lower chain D. We use $VP(p)$ to denote the visibility polygon of a point p inside the polygon; $VP_R(p)$ to denote the

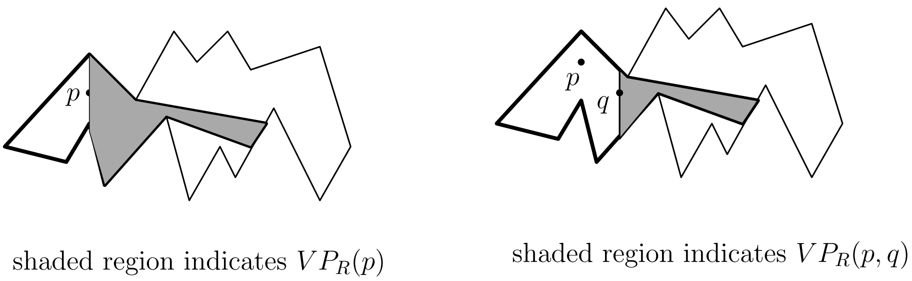


Figure 4.3: Visibility Polygon, Dark Boundary indicates $VP(p)$

region of $VP(p)$ that lies to the right of p, and $VP_R(p,q)$ to denote the region of $VP(p)$ that lies to the right of the point q inside the polygon. So, it is trivial to see that $VP_R(p,p) = VP_R(p)$.

Definition (Guard Cover) For a given polygon P, a set of interior points $G = \{g_1, \ldots, g_k\}$ is said to be a guard cover of P when $VP(G) = VP(g_1) \cup VP(g_2) \cup \ldots \cup VP(g_k) = P$. Note that G may not have minimum number of guard points.

Definition (Pocket) Let H be a set of guards placed inside a polygon. The area $P \setminus VP(H)$ (may be disjoint pieces) are known as pockets. The pockets which are adjacent to the upper boundary U are called as upper pockets. Similarly, lower pockets are defined. Note that an upper pocket can be adjacent to D, but a lower pocket can never be adjacent to U.

Definition (Kernel Expansion) For a region R inside the polygon, Kernel Expansion of R is defined as $ke(R) = \{p \in P | p$ sees everything in R to the left of themselves $\}$.

Let us assume that we have a partial guard cover G' that guards some parts of the polygon P (not the entire P), and also the entire region of P to the left of the rightmost guard in G' is seen. Consider the upper pockets resulting from this guard cover and enumerate them from left to right in this order $U_1, \ldots, U_k$. Similarly the lower pockets $D_1, \ldots, D_\ell$ are enumerated in the same way. Consider an upper pocket U_i. The *kernel expansion* $ke(U_i)$ consists of all the points in P that see everything in U_i to the left of themselves. Similarly, we define the kernel expansion for the lower pockets.

Definition (Spear) Let r be the largest index such that $\cap_{i=1}^{r} ke(U_i)$ is non-empty. $USP(G_p) = \cap_{i=1}^{r} ke(p_i)$ is the upper spear of G_p, It means that from this region $USP(G_p)$, all points of the upper pockets of P that are to the left

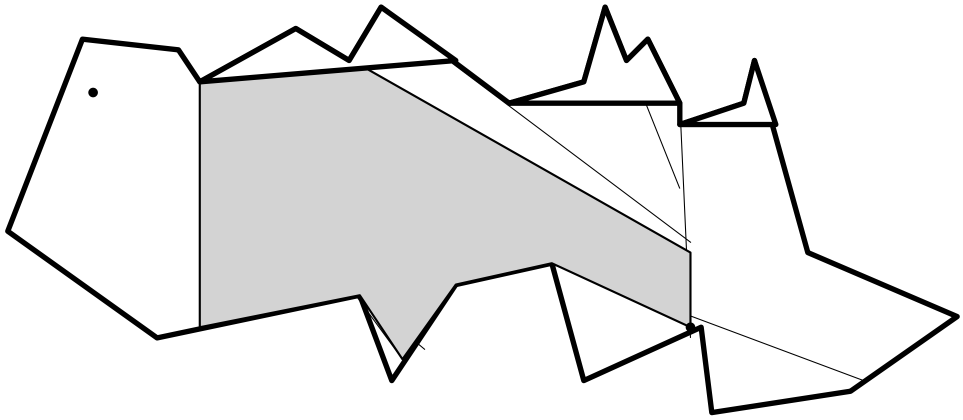


Figure 4.4: Computing spear of a set of pockets

of G_p can be seen. Similarly, we can define lower spear $DSP(G_p)$ of a partial guard cover G'. The rightmost point of an upper spear (resp. lower spear) is said to be the spear tip denoted by $USP\_TIP(G_p)$ (resp. $DSP\_TIP(G_p)$).

Definition (Shadow) To every spear sp, we associate a region called *shadow* of the spear denoted as $shd(sp)$. It is defined in the way as shown in Figure 4.4. If the spear tip lies on the boundary, then $shd(sp)$ is empty.

4.3.2 Algorithm

Our algorithm works as follows. It first computes the position of the guards in the entire upper chain as described below. The same method wors or computing the position of the guards the entire lower chain.

Initialize $G = \emptyset$. Repeat the following until all upper pockets are guarded. Compute VP(G). Pockets are enumerated. Compute $USP\_TIP(G)$. Put a guard g at $USP\_TIP(G)$. $G = G \cup g$. Compute VP(G). Consider the first upper pocket U_1. Draw a vertical line segment through the leftmost point of U_1. Place a guard $g\prime$ on that line so that $USP\_TIP(G \cup g\prime)$ is as right as possible. Check if entire upper chain is guarded. If not, then repeat this step.

Now we will discuss how to compute Step 2.1 and Step 2.5 in brief detail.

Computing Kernel Expansion of a given region R: Let $v_1, v_2, \ldots, v_m$ be the vertices of the region R ordered from let to right. Initialize $K = VP(v_1)$. Then for each vertex from v_2 to v_m in this order, compute $VP_R(v_i)$. Denote $K_L(v_i)$ and $K_R(v_i)$ be the region of K that lie to the left and right

$GuardMonotonePolygon(P)$

1 Let $G = \emptyset$;

2 **while** all upper pockets are guarded do

2.1 Compute $USP\_TIP(G)$;

2.2 Place a guard g at $USP\_TIP(G)$;

2.3 $G = G \cup g$;

2.4 Compute $\cup_{g \in G} VP(G)$; let U_1 be the first upper pocket in P and let ℓ is the vertical line segment through the leftmost boundary of U_1.

2.5 Place a guard $g\prime$ on l so that $USP\_TIP(G \cup g\prime)$ lies as far top the right as possible; $G = G \cup g\prime$;

3 Repeat step 2 for lower pockets;

4 Return G;

---

of v_i respectively. Update K as $K = K \cup (K_R(v_i) \cap VP_R(v_i))$. Repeat this process for i = 2,....,m. At the end return K. The pseudo-code of the method is as follows:

$ComputeKernelExpansion(R)$

1 Order the vertices of R as $v_1, ..., v_m$ from left to right. Initialize $K = VP(v_1)$

2 for$(i = 2; i \leq m; i = i + 1)$

2.1 Compute $VP_R(v_i)$;

2.2 $K = K \cup (K_R(v_i) \cap VP_R(v_i))$;

3 Return K;

After computing kernel expansion of all the pockets, we can compute the spear easily.

Computing Step 2.5 (as stated in the algorithm): We use a plane sweep approach on the vertical line segment ℓ and keep on updating the

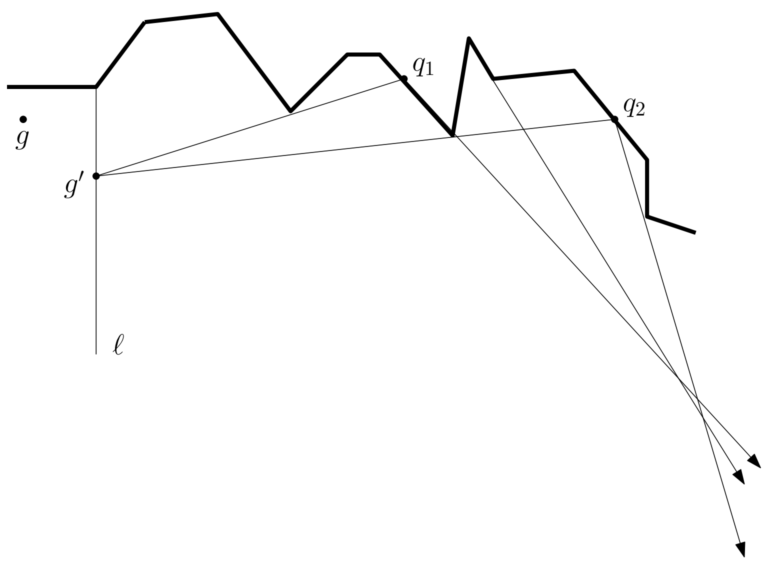


Figure 4.5: Computing the right most spear tip

$USP(G \cup \{g\prime\})$ continuously. The change in the combinatorial structure takes place at the following points. We compute only those points moving in between will move the spear tip monotonically to the left or monotonically to the right.

1 A convex vertex of $VP(G) \cup VP(g\prime)$ on an edge adjacent to an upper pocket falls on a vertex of U.

2 An edge of the boundary of $USP(G \cup \{g\prime\})$ falls on two vertices of the upper boundary U.

3 Three consecutive half lines issuing from pockets meet at one point.

These are the types of points where changes will occurs. Number of such points is of the $O(n^3)$ (there are total n lines, we compute at maximum number of possible points where 3 lines can intersect). In this way, the rightmost spear tip is possible.

4.3.3 Approximation Factor of this algorithm

Now we discuss about the approximation factor of this algorithm. Before that we need to know the following term.

Serial Guard Cover: Given a guard cover $G = g_1, g_2, ..., g_m$ and they are ordered from left to right, where g_m is placed either at the upper spear of $g_1, g_2, ..., g_{m-1}$ or at the lower spear of $g_1, g_2, ..., g_{m-1}$.

Lemma 4.3.1 *If H is an arbitrary guard cover for P, then there is a serial guard cover H^* for P such that $|H^*| \leq 3|H|$.*

Proof Given an arbitrary guard cover H, we transform it to serial guard cover in the following manner. During this transformations, guards are added to H_U, H_D and H_r. Initialize all H_U, H_D and H_r to empty at the beginning of the construction. Use a vertical line for plane sweep approach. Start the vertical sweep line from s. When the sweep reaches a guard h, that gets attached to the sweep line. Then, it starts moving through the shortest path from h to t(the right most point of the polygon). Also at some point of time, some guard may be released from that sweep line. H_r consists of those set of points that are released from the sweep line. Now, as the sweep proceeds, we keep on considering the set $USP(H_r)$ and also $DSP(H_r)$ then the following things can happen.

Case 1: A guard h becomes the last guard to leave a spear ($USP(H_r)$ or $DSP(H_r)$). If that happens then we release h from the sweep line and add a guard h' to H_r at the point where h was leaving the spear. And then repeat the same procedure from that point again.

Case 2: If a guard(attached to the sweep line) reaches a $USP\_TIP(H_r)$ (respectively $DSP_TIP(H_r)$), then we place a guard hs at $USP\_TIP(H_r)$(respectively $DSP_TIP(H_r)$) and add it to H_U(respectively H_D).

As long as the sweep reaches t, we stop and return $H^* = H_r \cup H_U \cup H_D$ as output. Now, we have to compute how many extra guards this construction has placed. Number of guards added to H_r can be at most the number of guards that are in H. So, $|H_r| \leq |H|$. Similarly, $|H_U| \leq |H|$ and $|H_D| \leq |H|$. Therefore $|H^*| \leq |H_r| + |H_U| + |H_D| \leq |H| + |H| + |H| = 3|H|$.

Theorem 4.3.2 *The above guarding algorithm provides us at most 12OPT number of guards for any x-monotone polygon.*

Proof Let us first compare the cardinality of the produced guard cover with the serial guard cover. Assume that a serial guard cover is ordered from left to right in this way $g_1, g_2, ..., g_m$. We see that at each step in the algorithm, for the guard g_i first it places a guard at the spear tip and then on the vertical line corresponding to the leftmost pocket. So, the next spear tip is placed as right as g_{i+1}. Therefore for the upper chain its size is 2 times the serial guard cover. Same occurs for lower chain. So, $|G| \leq 4|G^*| \leq 12 * OPT$. The reason is that we can choose the serial guard cover such that it is at most 3 times any optimal guard cover. Therefore, this algorithm provides us a 12 factor approximation algorithm proving the theorem.

4.4 Our Approach for a Special Sub Case

Now we discuss about our approach to guard a x-monotone polygon when s and t are mutually visible.

4.4.1 Algorithm

The main idea of our algorithm is as follows. We split the upper chain into convex parts, namely $C_1, C_2, \ldots, C_m$. We consider these convex parts in order. For each convex part C_i, we compute the portion R_i in the upper chain where from C_i is completely visible. Let $C_1, C_2, \ldots, C_i$ be the convex parts such that $R_1, R_2, \ldots, R_i$ have a non-empty intersection $\mathcal{R}$, and R_{i+1} does not intersect $\mathcal{R}$. We choose a point g_1 in $\mathcal{R}$ to see $C_1, C_2, \ldots, C_i$ completely. Now, we compute the non-visible portions of $C_{i+1}, C_{i+2} \ldots, C_m$ from g_1. Again split these regions into convex parts, and execute the same algorithm to place the next guard g_2. Next time, we compute the non-visible portions from g_1 and g_2 and so on. The process stops when there is no non-visible portion in the upper chain. We repeat the same process for the entire lower chain.

4.4.2 Analysis of Our Algorithm

Our algorithm will not provide optimal solution. We have some non-trivial observations about our algorithm.

Observation If there is a contiguous convex sub-chain consisting of 3 or more edges, then for that sub-chain, our greedy algorithm will put at most an extra guard than the minimum number of guards to guard that sub-chain.

Proof Suppose, there is a contiguous convex sub-chain C_k. On its left there is a monotone chain C_{k-1} and also on its right there is one convex sub-chain C_{k+1}. Our greedy algorithm finds that C_{k-1} and C_k must be treated separately. Also, it finds that C_k and C_{k+1} must be treated separately as

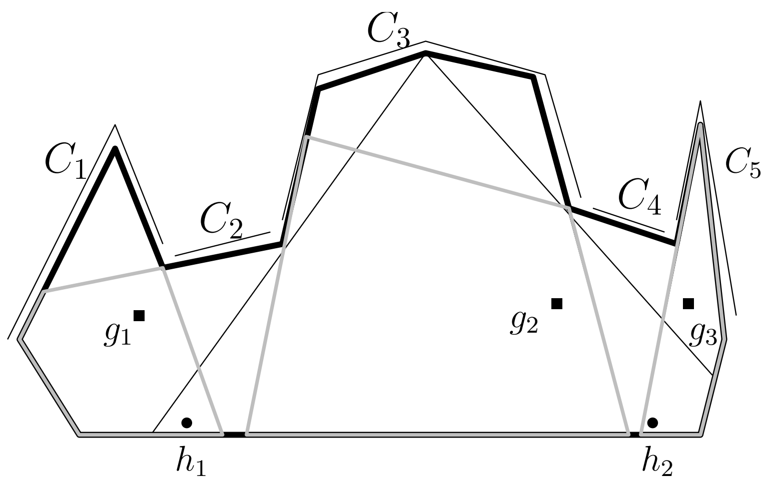


Figure 4.6: A monotone polygon

we can see in the above figure (small points indicates the places where our greedy algorithm places guards, big points indicate where the minimum set of guards are placed). C_k can be divided into two adjacent sub-chains so that those two sub-chains can be separately guarded by one from its left and one from its right. We are placing an extra guard for each such sub-chains. This proves our observation.

The set of guards returned by our algorithm will give a constraint about the guards that every edge is completely seen by at least one guard. Because at each iteration, it computes the intersection of completely visibility polygon of the edges. But, when a sub-chain is convex and both its lower and upper parts are disjoint from its left most and right most edge, then an interesting thing occurs. Our algorithm will place at most one extra guard for each such sub-chains. Therefore, our chain gives twice the actual number of guards that is actually required. Thus we conjecture that approximation factor of our algorithm is 2. This comes from the extension of our earlier observation that we have. Now, we can state the following conjectures. Using similar approach to guard the entire lower chain also will involve an approximation factor of 2. So, overall approximation factor is 4.

Conjecture 4.4.1 *When s and t are mutually visible to each other, then our greedy algorithm provides* $2OPT_U$ *guards for the upper chain.*

Conjecture 4.4.2 *Our greedy algorithm gives an approximation factor 4 to guard a monotone polygon when s and t are mutually visible to each other.*

Bibliography

[1] Computational Geometry: Algorithms and Applications.

[2] Erik A. Krohn, Bengt J. Nillson. Approximate Guarding of Monotone and Rectilinear Polygons. Algorithmica 2012.

[3] B. Ben-Moshe, M. Katz, J. Mitchell. A constant factor approximation algorithm for optimal terrain guarding. Symposium on Discrete Algorithms 2005

[4] K.L.Clarkson, K.Varadarajan. Improved Approximation Algorithms for Geometric Set Cover. Proc 21st ACM Symposium on Computational Geometry 2005.

[5] Bengt J. Nillson. Approximate Guarding of Monotone and Rectilinear Polygons. ICALP 2005.

[6] D. Z. Chen, V. Estivill-Castro, and J. Urrutia. Optimal guarding of polygons and monotone chains. In Proceedings of the 7th Canadian Conference on Computational Geometry, pages 133138, 1995.

[7] Erik Krohn, Bengt J. Nillson. The Complexity of Guarding Monotone Polygons. Canadian Conference of Computational Geometry, August 8-10, 2012.

[8] James King, Erik Krohn. Terrain Guarding is NP Hard. SIAM 2009.

[9] Erik Krohn. Survey of Terrain Guarding and Art Gallery Problems.

[10] K. Elbassioni, Erik Krohn, Domagoj Matijevic, Julian Mestre, Domagoj Severdija. Improved Approximations for Guarding 1.5-Dimensional Terrains. Algorithmica 2011

[11] Matt Gibson, Gaurav Kanade, Erik Krohn, Kasturi Varadarajan. An Approximation Scheme for Terrain Guarding. Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques. Lecture Notes in Computer Science Volume 5687, 2009, pp 140-148

[12] James King. A 4-approximation algorithm for Guarding 1.5 Dimensional Terrains. Lecture Notes in Computer Science (3887), 629-640, 2006.

[13] L. Gewali, M. Keil, S.C.Ntafos. On covering orthogonal polygons with star-shaped polygons. Information Sciences 65(1992) 45-63.

[14] Andrej Lingas, Agnieszka Wasylewicz, Pawel Zylinski. Note on covering monotone orthogonal polygons with star-shaped polygons. Information Processing Letters 104 (2007) 220-227.

[15] Subir Kumar Ghosh. Approximation Algorithms for Art Gallery Problems in Polygons and Terrains. WALCOM: Algorithms and Computation 2010.