Survey of Terrain Guarding and Art Gallery Problems

Erik Krohn

Abstract

The terrain guarding problem and art gallery problem are two areas in computational geometry. Different versions of terrain guarding involve guarding a discrete set of points or a continuous set of points on a terrain. The art gallery problem has versions including guarding an entire polygon by a set of discrete points at the vertices or any point inside the polygon itself. This paper will give a survey as to what is known about each of these problems and what improvements could be worked on. This paper also provides a 4-approximation to the vertex terrain guarding problem.

1 Introduction

The terrain guarding problem and the art gallery problem are two areas in computational geometry. The terrain guarding problem and the art gallery problem are instances of the set cover problem that are induced on a geometric setting.

1.1 Terrain Guarding Problem Introduction

An instance of the terrain guarding problem contains a terrain T that is an x-monotone polygonal chain. An x-monotone chain in $\Re^2$ is a chain that intersects a vertical line at most once. The terrain is defined by a set of points $P = \{v_1, v_2, ..., v_n\}$. A vertex v_i is defined with coordinates (x_i, y_i). The points are ordered such that $x_i < x_{i+1}$. There is an edge connecting each (v_i, v_{i+1}) pair where $i = 1, 2, ..., n-1$. We say a point p on the terrain sees another point q on the terrain if the line segment $\overline{pq}$ lies entirely above the terrain T.

We wish to find a set of vertices $G \subseteq P$ such that each of the vertices in P is seen by a point in G. We call this set G a guarding set. It is obvious to

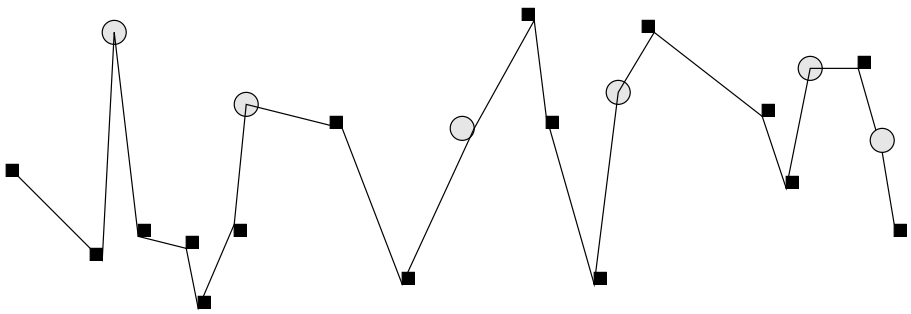

Figure 1: Terrain Guarding Example

see that P itself is a guarding set. The optimization problem is thus defined as finding the smallest such G.

We restrict guards to be placed on the terrain and not above the terrain. This is a reasonable restriction considering a guard placed sufficiently high will guard the entire terrain.

An example is the best way to visualize how the terrain guarding problem works. In Figure 1, there are 22 vertices that must be guarded, thus n=22. The optimization problem thus asks what is the minimum number of guards that must be placed in order to see the entire terrain. The decision version of the problem asks whether k guards are sufficient to guard the entire terrain. It is unknown whether these questions can be answered in polynomial time.

The terrain in Figure 1 consists of the points $\{x_1, x_2, ..., x_n\}$ and an edge between x_i and x_{i+1} where $i = 1, 2, ..., n-1$. The guarding set G consists of the following points: $\{x_3, x_8, x_{11}, x_{15}, x_{19}, x_{21}\}$. The terrain has 22 vertices that also act as potential guards. The black squares are points that need to be guarded, the circles are guards. The circles are also points that need to be guarded. Consider the vertices on the terrain in order, x_1 is the leftmost point, x_2 is the point to the right of it and so on until x_n is the rightmost point. In the figure it is clear to see that x_1 is guarded by x_3 since $\overline{x_1 x_3}$ lies entirely above the terrain. x_2 is also guarded by x_3 for the same reason.

1.2 Art Gallery Problem Introduction

The art gallery problem has similar features to the terrain guarding problem. The art gallery problem is defined by placing points inside a polygon such that the entire polygon is guarded. An instance of the art gallery problem P contains a set of points $\{v_1, v_2, ..., v_n\}$ where an edge connects (v_i, v_{i+1}) when $i = 1, 2, ..., n-1$. There is also an edge that connects v_n to v_1. We

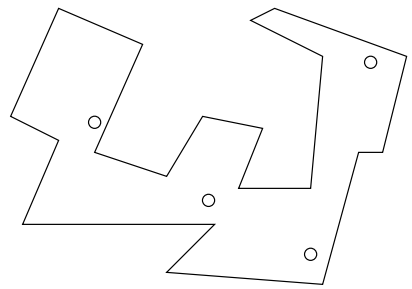

Figure 2: Art Gallery Example

assume that no two edges intersect. The exception to this is that consecutive edges (v_{i-1}, v_i) and (v_i, v_{i+1}) intersect at one point, v_i. This type of polygon is considered simple. The curve is formed by connecting the edges (v_{i-1}, v_i) to (v_i, v_{i+1}) at point v_i for $i = 2, 3, ..., n-1$. We also connect (v_{n-1}, v_n) and (v_n, v_1) at point v_n. These connections will separate the plane into two connected regions, an interior and an exterior. A simple polygon is defined as the interior and the curve itself, which forms the boundary.

The point guarding problem is defined as the following. Consider all points on the curve and inside the curve and call that set Q. The goal in the point guarding problem is to come up with a guardset $G \subseteq Q$. The property that must hold is that $\forall q \in Q$, $\exists g \in G$ such that g sees q. G is said to be the minimum cover of P if $|G|$ is the smallest among all guard covers of P. The problem then becomes finding the minimum cover.

The vertex guarding problem is a little more restrictive than point guarding. In this problem, we are restricting guards to be at vertices of the polygon. We wish to find a $G \subseteq P$ such that $\forall q \in Q$, $\exists g \in G$ such that g sees q. The problem then, similar to the point guarding problem, is finding the smallest such guardset G.

A good way to visualize this is through an example in Figure 2.

Consider the polygon in Figure 2. The goal is to place guards so that the entire polygon is seen. It is easy to see that points placed at the circles will see the entire polygon. Therefore the polygon can be seen using 4 guards. The decision question of whether any given polygon can be seen by k guards is known to be NP-hard, even if the polygon is simple, as shown in [11]. The problem is NP-hard whether we place guards anywhere or we restrict guards to be placed only at the vertices. The vertex guarding problem is a restricted version of the art gallery problem in that we are only allowed to place guards on the vertices. An example of this is shown in Figure 3.

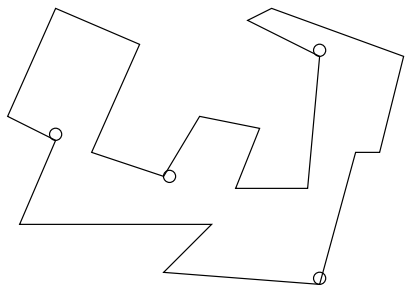

Figure 3: Art Gallery Example Vertex Guarding

A convention used throughout this paper will be the notion of a d-approximation. As an example take the polygon shown in Figure 3. The optimal solution to the problem is 4. Let us say we have an algorithm that provides a 3-approximation. This means our 3-approximation algorithm will place no more than 12 (4x3) guards and solve the problem.

The remainder of the paper is set up as follows. Section 1.3 gives some motivations to solving the art gallery and terrain guarding problems. Section 2 will deal with the terrain guarding problem and will provide a simple 4-approximation to the vertex guarding problem. Section 4 will get into the art gallery problem, show why it's NP-complete, provide a reason why $\frac{n}{3}$ guards are necessary and sufficient for guarding an art gallery and describe some other results. Section 5 will give a conclusion and a look into some open problems in the areas.

1.3 Motivations

Art gallery problems are motivated by problems such as line of sight transmission networks, signal communications, cell phone tower placement and other network related problems. A terrain guarding problem is motivated from guarding or covering a road with either security cameras or lights. Some of the greedy algorithms for set cover type problems lead to very bad results. The original set cover problem can have a bad solution if it chooses greedily. Terrain guarding and art gallery problems are also not immune to this problem. An terrain guarding example illustrating this problem is given by King in [7].

2 Terrain Guarding

The terrain guarding problem can be divided into two problems, a discrete version and a continuous version. The discrete version focuses on guarding only the vertices of the terrain (or some discrete set of chosen points). Whereas the continuous version focuses on guarding the entire terrain. Finding an optimal set of guards is not known to be NP-hard. A proof proposed by Chen et al in [2] suggested an NP-hardness result was obtainable using a modification of Lee and Lin's proof in [8]. However, the details were omitted and were never verified. Neither attempt at proving NP-completeness or finding a polynomial time algorithm has been successful. The first constant factor approximation came from Ben-Moshe in [9]. Clarkson and Varadarajan gave another constant factor approximation for the problem based on solving a linear programming relaxation and rounding in [3]. No attempts were made to minimize the constant in either paper. King notes in [7] that the factor in [9] can be brought down to 6 with careful bookkeeping and minor modifications. King's paper in [7] provides a 5-approximation to the terrain guarding problem and until now was the best known approximation.[1] Some constant factor approximations have been given but no polynomial time approximation scheme is known such that, for any $\epsilon > 0$, we are guaranteed a $(1 + \epsilon)$-approximation. This paper will provide a very simple 4-approximation to the terrain guarding problem.

3 4-approximation Algorithm

The following is the best known approximation algorithm for guarding a terrain. It is also worth noting that many other algorithms are much more complex than our algorithm. Our algorithm produces a 4-approximation in polynomial time.

3.1 Preliminaries

Let T be an x-monotone polygonal chain having n vertices. T is specified by n vertices with an edge connecting (v_i, v_{i+1}) for i=$1, 2, ..., n-1$. A vertex v_i is made up of coordinates $(x_i$,$y_i)$. For convenience, the leftmost point of T will be referred to as s and the rightmost point of T will be referred to as t. We are interested in guarding the n vertices as opposed to the entire

[1] The paper originally published claims a 4-approximation. However, an error in the paper was discovered and the approximation guarantee has been weakened to 5. The errata describes this in [7].

terrain. Let p and q be two vertices on T. We say that p sees q if and only if the line segment $\overline{pq}$ lies entirely above the terrain. We say $p < q$ if $x_p < x_q$. For a given point $p \in T$, $L(p)$ will denote the leftmost point in T that sees p. $R(p)$ is defined similarly. A subterrain $[u, v]$ is defined as the the part of the terrain between u and v inclusive where u and v are points on the chain T. A *guard* for point $p \in T$ is a point on T, say g, that sees p. A guardset $G \subseteq T$ is defined as the following. For each vertex $x \in T$, there is at least one vertex $g \in G$ where g sees x. We use the order claim as defined below:

Lemma 1. *Let a, b, c and d be 4 points on T and $a < b < c < d$. If a sees c and b sees d, then a sees d.*

Proof. It is easy to see that there is no point that lies above the line segment $\overline{ac}$ since a *sees* c. It is also clear that no point lies above the segment $\overline{bd}$ since b *sees* d. By the ordering of a, b, c, and d, it is clear to see that no point can lie above the segment $\overline{ad}$. Therefore a *sees* d. □

3.2 Exact Algorithm For One Sided Guarding

This section we will provide an exact algorithm for choosing guards in the following manner. Pick a point on T that must be guarded. Call that point p. We must guard the point p with a guard to the left of p. In other words, any guard g for p must have the feature that $x_g \leq x_p$. The leftmost vertex of T will always be a guard since $L(v_1) = v_1$. We use the following lemma to show that it is possible to select the optimal set of guards in polynomial time.

Lemma 2. *Let T' be any nonempty set of vertices on the terrain. There exists a vertex $p \in T'$ such that p is not $L(q)$ for any $q \in T'$ and there is no $r \in T'$ where $L(p) \leq L(r) < r < p$.*

Proof. Assign p to be v_n. It is true that p will not be the $L(q)$ for any $q \in T$. If there does not exist an r such that $L(p) \leq L(r) < r < p$, then we are done. If such an r exists, assign p to be the rightmost such r and continue on. We can also see that the r will not be $L(q)$ for any vertex q. Suppose a q exists such that r is $L(q)$ for some q. Since $r = L(q) > L(p)$, we must have that $q < p$. This contradicts the fact that r is the rightmost vertex with $L(p) \leq L(r) < r < p$. We will eventually reach the base case where there is no vertex r where $L(p) \leq L(r) < r < p$. When we reach this case we are done. □

Algorithm

The algorithm takes as a parameter a set of points T' and returns an optimal guarding set for the terrain.

leftSideTG(T')

1. If T' is empty, return $\emptyset$.
2. Find a $p \in T'$ such that p is not $L(q)$ for any $q \in T'$ and there is no $r \in T'$ such that $L(p) \leq L(r) < r < p$.
3. $T'' \leftarrow T' \setminus$ all points to the right of $L(p)$ that $L(p)$ sees.
4. $G \leftarrow L(p)$.
5. Return leftSideTG(T'') $\cup$ G.

3.3 Proof of Algorithm

We prove by induction on the size of T' that leftSideTG(T') produces an optimal set of guards for T'. The base case is when T' is empty. We proceed to the inductive step where T' contains some set of points that we need to guard. Let OPT be an optimal guard set for T'. Let us consider a point $p \in T'$ chosen by leftSideTG(T') in step 2. There is a guard $g \in OPT$ that sees p. We show that the point $L(p)$ sees any point in T' that g sees. This is clearly true if $g = L(p)$. It must be the case that $L(p) \leq g \leq p$. Assume $g \neq L(p)$.

We observe that there is no $r \in T'$ such that $L(p) < r < p$. By the order claim, such an r must have $L(p) \leq L(r)$. This would give us $L(p) \leq L(r) < r < p$. This is not possible because of our choice of p.

We only have to show that a $q \geq p$ that is seen by g is also seen by $L(p)$. If $g < p$, it follows that $L(p)$ will see such a point q by the order claim. We are left with the case where $g = p$.

Suppose there is a point $q > p$ that p sees that $L(p)$ does not see. q must then lie below the line segment $\overline{L(p)p}$ otherwise $L(p)$ would see q. There are no vertices in T' that lie to the left of $L(p)$ that lie above the line segment $\overline{L(p)p}$ by the definition of $L(p)$. It must be the case the $p = L(q)$. However, this contradicts our choice of p in step 2.

We thus conclude that $L(p)$ sees any point in T' that g sees. We now have $OPT \setminus \{g\}$ as a guarding set for T''. T'' is all points in T' that $L(p)$ does not see. By the inductive hypothesis, leftSideTG(T'') is a guarding set for

T'' whose size is at most $OPT \setminus \{g\} = OPT - 1$. Therefore leftSideTG($T'$) $= \{L(p)\} \cup$ leftSideTG(T'') is a guarding set for T' whose size is at most OPT.

An algorithm called rightSideTG(T, G) works similarly and is shown below.

rightSideTG(T, G)

1. Find a $p \in T$ such that p is not $R(q)$ for any $q \in T$ and there is no $r \in T$ such that $p < r < R(r) \leq R(p)$.
2. $T' \leftarrow T \setminus$ all points to the left of $R(p)$ that $R(p)$ sees.
3. $G \leftarrow G \cup R(p)$.
4. If T' is empty, return G, else return rightSideTG(T', G)

3.4 Algorithm Example

Consider Figure 4 to help explain Lemma 2 and the algorithm leftSideTG. We would not be allowed to pick a initially because a is $L(b)$. However b is able to be chosen as our p. We place a guard at $a = L(b)$ and continue. A next guess would be choosing point c to be our next p. However, this is also not allowed since we have an r such that $L(p) \leq L(r) < r < p$, namely $L(c) \leq L(d) < d < c$. Therefore we must chose d as our next p and place a guard at $L(d)$. These are the two cases in picture form.

Theorem 3. *There exists a polynomial time algorithm that optimally guards a terrain from the left (or right).*

3.5 LP-approach

The following section will describe the 4-approximation algorithm for the problem of finding the smallest subset of vertices that guard all terrain vertices. The approach is based on solving the linear program (LP) relaxation of terrain guarding. We then round a fractional solution to an integer solution. The objective function of the LP:

$$\min \sum_{i=1}^{n} w_i.$$

The variable w_i's range over the non-negative real numbers. The constraints of the LP are:

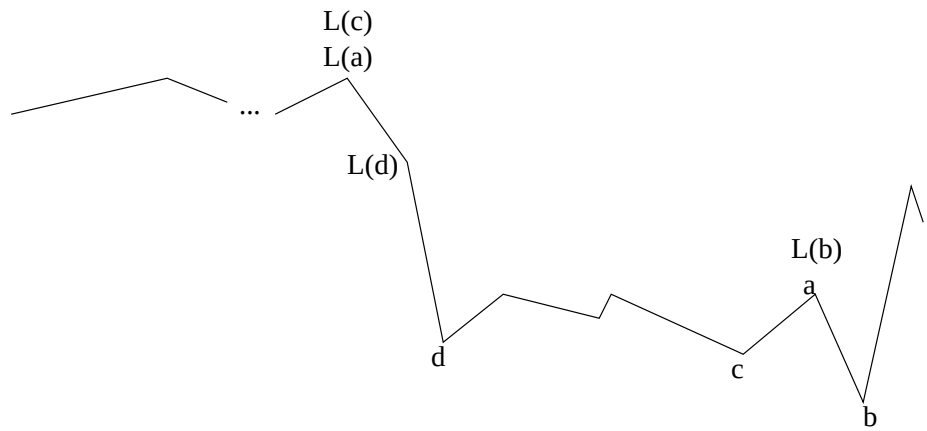

Figure 4: Sample terrain

$$\forall y \in P, \sum_{i:i \text{ sees } y} w_i \geq 1.$$

Each guard i may be thought of as a fractional guard whose fraction is w_i. The constraint for y says that each vertex y sees a bunch of fractional vertices on the terrain; the sum of those fractional vertices must be equal to or greater than 1.

Let $(\overline{w_1}, \overline{w_2}, ..., \overline{w_n})$ be the optimal solution of the LP. It is clear that $\sum_{i=1}^{n} \overline{w_i} \leq OPT$ where OPT is the optimal terrain guard cover. We now place each vertex that must be guarded into two sets, L and R. A vertex will be placed in the set L if the following is true:

$$\sum_{i:i \leq y \text{ and } i \text{ sees } y} \overline{w_i} \geq \frac{1}{2}.$$

If the vertex is not placed in the set L, it is placed in the set R. We now have two sets of vertices that must be guarded. Let us set $l_i = 2\overline{w_i}$ for each vertex i. Note that for each $q \in L$, we have $\sum_{i:i \leq q \text{ and } i \text{ sees } q} l_i \geq 1$. We call the procedure leftGuard(L, l) which produces a set of at most $\sum_{i=1}^{n} l_i$ guards that see each vertex L from its left. We call a similar procedure rightGuard(R, l) that produces a set of at most $\sum_{i=1}^{n} l_i$ guards that see each vertex R from the right. We return the union of these two sets of guards. There size is at most $2\sum_{i=1}^{n} l_i \leq 4\sum_{i=1}^{n} w_i \leq 4OPT$, thus we have a 4-approximation.

We now describe the procedure leftGuard(L', l'). It takes as input a set of guards L' and a non-negative vector $l' = (l'_1, l'_2, ..., l'_n)$ such that for any

$q \in L'$ we have $\sum_{i:i \leq q \text{ and } i \text{ sees } q} l'_i \geq 1$. As we will show, the procedure returns a set of at most $\sum_{i=1}^{n} l'_i$ guards that sees each vertex in L' from its left.

leftGuard(L', l')

1. If $L' = \emptyset$, return $\emptyset$.
2. Find a $p \in L'$ such that p is not $L(q)$ for any $q \in L'$ and there is no $r \in L'$ such that $L(p) \leq L(r) < r < p$.
3. $L'' \leftarrow L' \setminus$ all points to the right of $L(p)$ that $L(p)$ sees.
4. $l''_i \leftarrow 0$ for each vertex i in the range $[L(p), p]$. Let $l''_i \leftarrow l'_i$ for every other vertex.
5. Return $L(p) \cup$ leftGuard(L'', l'').

Lemma 4. *Assuming that $\sum_{i:i \leq q \text{ and } i \text{ sees } q} l'_i \geq 1$ for each $q \in L'$, leftGuard(L', l') returns a set of at most $\sum_{i=1}^{n} l'_i$ guards that see each point in L' from its left.*

Proof. We do this by induction on $|L'|$. The base case is when $|L'| = 0$. Consider the inductive step. Any guard that sees the p chosen in step 2 must lie in the range $[L(p), p]$. Thus, the sum of the l' values of vertices in this range is at least 1. Therefore, $\sum_{i=1}^{n} l'' \leq \sum_{i=1}^{n} l'_i - 1$. From our argument of the leftSideTG algorithm, we can assert that any $q \in L'$ that is seen from its left by some $g \in [L(p), p]$ is also seen by $L(p)$. It follows that any $q \in L''$ is not seen from its left by any $g \in [L(p), p]$. So $l''_i = l'_i$ for any i that sees $q \in L''$ from its left. Thus we have $\sum_{i:i \leq q \text{ and } i \text{ sees } q} l''_i \geq 1$ for any $q \in L''$.

Using the inductive hypothesis, we conclude that leftGuard(L'', l'') computes a set of at most $\sum_i l''_i$ guards that see each point in L'' from its left. So leftGuard(L', l') computes a set of at most $1 + \sum_i l''_i \leq \sum_i l'_i$ guards that see each point in L' from its left. □

The procedure rightGuard(R', r') works analogously. We conclude with the main result of this section.

Theorem 5. *There exists a polynomial time algorithm that provides a 4-approximation to the vertex terrain guarding problem.*

4 Art Gallery Problem

The art gallery problem as described in the introduction can come with different restrictions. The two that will be looked at in this paper are the point guarding problem and the vertex guarding problem. This section will contain an NP-completeness proof for the art gallery problem and an explanation on why $\frac{n}{3}$ guards are sufficient and sometimes necessary to guard a polygon.

4.1 NP-Completeness Proof

The general art gallery problem was shown to be NP-complete by Lee and Lin in [8]. They show that the problem is NP-hard for both the vertex guard problem and the point guard problem. The proof is a nice reduction from the 3SAT problem.

4.1.1 Preliminaries

The following terminology will be used for the NP-completeness proof of the art gallery problem. A simple polygon is defined as a set of points $P = v_0, v_1, ..., v_n$ where $v_i = (x_i, y_i)$. A straight line connects the points (v_i, v_{i+1}) and where $v_0 = v_n$. There is also the constraint that no two edges intersect, except at the endpoints of consecutive edges; thus the polygon is simple. Two points p and q are said to be visible from each other, or p sees q, if the line segment $\overline{pq}$ lies completely inside the curve formed by the points in P (it is ok if the line segment touches the edges of P or points in P).

4.1.2 Reduction from $3SAT$

The following section will show a reduction from $3SAT$ to the vertex guarding problem in polynomial time. The $3SAT$ problem consists of a set of n Boolean variables, say $U = \{u_1, u_2, ..., u_n\}$ and a set of m clauses $C = \{c_1, c_2, ..., c_m\}$ where each clause $c \in C$ is composed of literals formed by three Boolean variables. For example, c_1 could be defined as $(u_2 \vee \overline{u_6} \vee u_9)$.

The question we want to know is if there is a satisfying truth assignment for C. More formally, does a truth assignment for the Boolean variables in U exist such that $c_1 \wedge c_2 \wedge ... \wedge c_m$ is true?

The transformation from $3SAT$ to the vertex guard problem consists of creating a couple of gadgets and finding a way to connect them so the final product is a simple polygon. The polygon will then have the feature

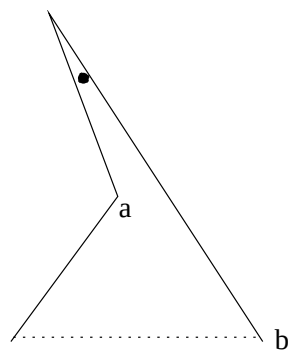

Figure 5: Literal Pattern

that if the minimum cover of the polygon is of size K, then there is a truth assignment that satisfies the $3SAT$ instance. The converse is true such that if there is a truth assignment that satisfies the $3SAT$ instance, the polygon is coverable with K guards. In the following sections, $K = 3m+n+1$ where m is the number of clauses and n is the number of Boolean variables. The first gadget that is created is called a literal pattern. Literal patterns are connected to a clause junction. We then create a gadget called a variable pattern and then put all of them together to create our polygon.

A literal pattern example is shown in Figure 5. It is clear to see from the figure that only vertex a or vertex b can cover the entire pattern.

The next gadget is called a clause junction. Consider any clause $c_i \in C$ and assume the variables of c_i are u_i, u_j and u_k. The pattern for the clause is shown in Figure 6. The figure has $a1, a2, a3$ and $a4$ instead of $u_{i1}, u_{i2}, etc.$ Similarly for b and c. It can be seen that none of the guards g_i can cover the entire literal pattern for u_i, u_j, or u_k. It is also true that no two vertices of the literal pattern gadget are sufficient to cover the entire clause junction. In the figure, the following sets of points are collinear: $(g_2, g_8, a_4, a_1, b_4, b_1, d_4, d_1, g_9, g_1)$, (g_3, g_4, g_7, g_1), (g_8, g_4, g_5), and (g_9, g_7, g_6). It is also the case that the length of $\mid g_2, g_8 \mid = \mid g_8, a_4 \mid$. We obtain the following two lemmas.

Lemma 6. *At least three vertices are required to cover the entire clause*

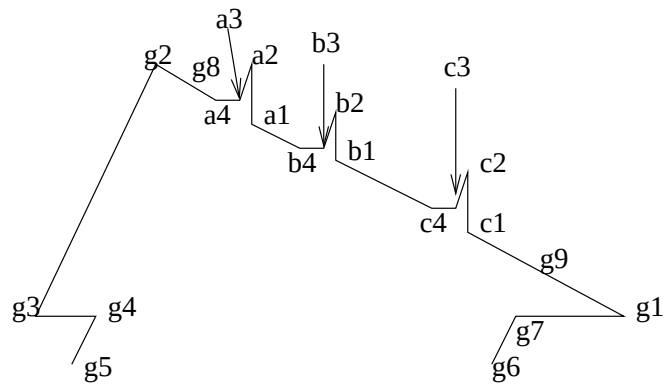

Figure 6: Cause Junction

junction.

Lemma 7. *Only seven three vertex covers exist that can cover the entire clause junction.*

Proof. Consider the vertices that are part of the literal pattern u_i, or in our picture, a_i for $i = 1, 2, 3, 4$. Choosing a guard at a_4 or a_2 does not cover the entire literal pattern. Therefore we should only select guards at a_1 or a_3. We also should not place a guard at both spots otherwise we will need two more guards to see the u_j and u_k (in the picture, b and c) literal pattern giving us four guards. We need to pick a guard in the set $\{a_1, a_3\}$ to see the entire literal pattern for a. We must do the same for b and c. Thus it is obvious that Lemma 6 is true. We end up with the following guard choices if we want a three vertex cover, $\{a_1, b_1, c_1\}$, $\{a_1, b_1, c_3\}$, $\{a_1, b_3, c_1\}$, $\{a_1, b_3, c_3\}$, $\{a_3, b_1, c_1\}$, $\{a_3, b_3, c_1\}$ and $\{a_3, b_1, c_3\}$. If we place guards at $\{a_3, b_3, c_3\}$, we will not see part of the clause junction, for example g_2. $\square$

This leads to a fairly obvious choice of how to label these vertices. If a given clause contains the variable u_i (a in the figure), we place a guard at a_1 to correspond to true and a_3 to correspond to false. If a given clause contains the variable $\overline{u_i}$, we place a guard at a_3 to correspond to true and a_1 to correspond to false. It is easy to see that three guards placed at $\{a_3, b_3, c_3\}$ will result in the clause being evaluated to false. It is also easy to verify that a clause that has a truth assignment of true will guard the entire clause polygon.

The final gadget that must be explained is the variable pattern gadget. The variable pattern is used as a consistency check for each variable. In

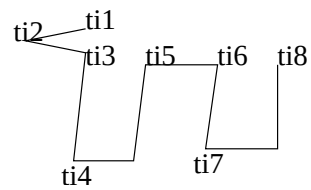

Figure 7: Variable Gadget

other words, a Boolean variable u_i should not be chosen to be true in one clause junction and false in another. The variable pattern for u_i is shown in Figure 7. Note that a point inside the triangle consisting of t_{i1}, t_{i2} and t_{i3} can only be guarded from the following points: $t_{i1}, t_{i2}, t_{i3}, t_{i5}, t_{i6}$ or t_{i8}.

The important parts of the variable pattern are the distinguished point that only a certain set of vertices can see and the two rectangle-like objects that protrude downwards. Now that all of the gadgets have been accounted for, three steps are needed to put them all together to create our polygon.

4.1.3 Putting variable patterns and clause junctions together

The variable patterns and clause junctions are put together as shown in the Figure 8. It should be noted that the following sets of lines are collinear $(t_{11}, g_{h5}, g_{h4}, g_{h8})$ as well as $(t_{n8}, g_{h6}, g_{h7}, g_{h9})$ for $h = 1, 2, ..., m$. The reason we have extra numbers/letters, ie t_{n8} as opposed to t_8, is to show the difference between each variable and clause. t_{n8} refers to the t_8 vertex for variable u_n however t_{18} refers to the vertex t_8 for variable u_1.

4.1.4 Adding spikes to variable patterns

Adding spikes to our variable pattern is a way to check consistency with the variables. Figure 9 shows the spike for a assuming $\overline{a}$ is in a clause c_j. It should be noted that the following sets of points are collinear (p, q, t_{i5}, a_3) and (p_1, q_1, t_{i8}, a_1) for $t = 1, 2, ..., n$. A way to think about the pattern is to say that t_{i5} represents false and t_{i8} represents true for a particular variable a. If a is in the clause c_j, then different sets of points would be collinear, namely (p, q, t_{i5}, a_1) and (p_1, q_1, t_{i8}, a_3). This gadget is not complete though. A spike is not allowed so it must be expanded to form a polygonal region. This is done in the next step.

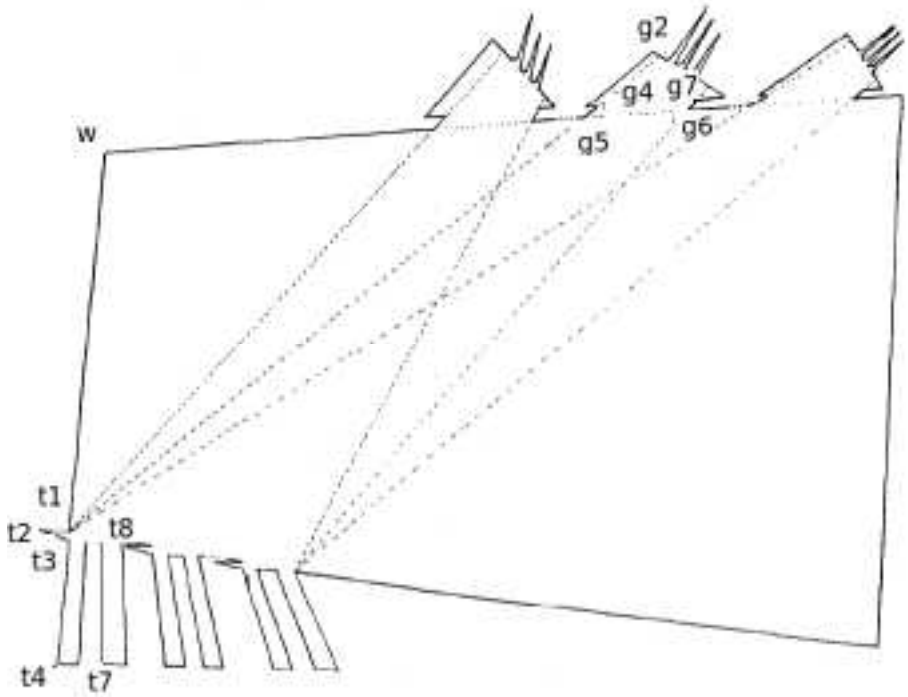

Figure 8: Putting the literal and variable patterns together, see Figure 7 and Figure 6 for naming of vertices for each vertex in each pattern

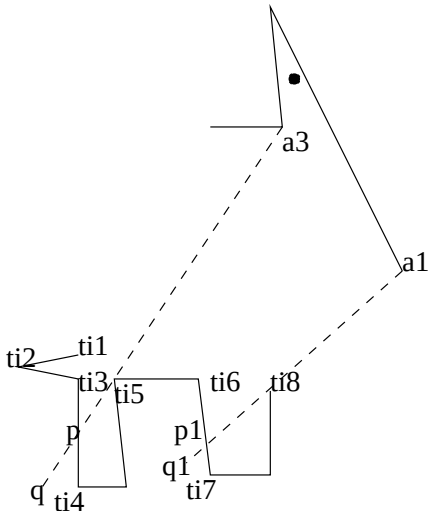

Figure 9: Augmenting spikes when $\overline{a} \in c_j$

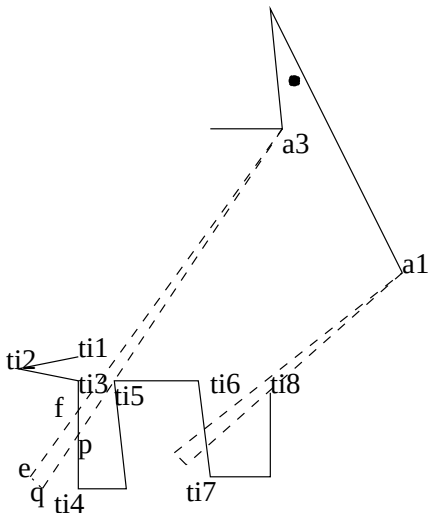

Figure 10: Each spike replaced with a region

4.1.5 Replacing spikes by polygonal regions

Let us take a to be a variable in clause junction c_h. Each spike is replaced by a polygonal region as shown in Figure 10. The region is defined as a triangle with corners at a_3, q and f; similarly defined for a_1. It should also be noted that (a_3, t_{i5}, p, q) are collinear and the same is true for a_1, e, f. These regions are used for consistency checks to ensure that we always choose a variable's truth assignment consistently. For example, we can not choose a to be true in a clause c_i and choose a to be false in a clause c_k. The next section will explain why this is not possible.

4.1.6 Putting the pieces together

The last section regarding the NP-completeness of the art gallery problem revolves around putting all of the pieces together and showing that the art gallery can be covered with K guards if and only if there is a solution to the $3SAT$ instance. We must have at least $3m + n + 1$ vertices to cover the polygon. We have m clause junctions made up of three literal patterns each. Each literal pattern has a distinguished point that can only be covered by one of two points. From Lemma 6 above, at least three guards are required per clause junction. This gives us at least $3m$ points being needed to cover all clause junctions. The variable patterns in Figure 7 also have a distinguished point that can only be guarded by a certain set of vertices. Each of these

vertices can only guard one distinguished point. There are n of these variable patterns because we have n variables; thus we need n more points. We need to add one more point at W from Figure 8 so we can see all of the rectangle areas of the variable patterns that the previous guards do not see.

The number of guards needed to cover this simply connected polygon region is $K = 3m + n + 1$ if and only if the $3SAT$ instance C is satisfiable. The first direction is rather easy to see. It says that if a $3SAT$ instance C is satisfiable, then the polygon can be covered with $K = 3m + n + 1$ guards. If C is satisfiable, there is a truth assignment to the variables in C that satisfy C. Pick any variable u_i and choose guards based on whether it is true or false. If u_i is true, then t_{i8} of the variable pattern for u_i is chosen, t_{i5} otherwise. Choosing this will cover all of the polygon regions that we created in the replacing spikes by polygon regions. If u_i is in a particular clause, we place a guard at a_{h1}. If $\overline{u_i}$ is in the clause, we place a guard at a_3. From our construction of the spikes, we are able to cover all of the polygonal regions created using $3m + n$ guards. What remains is covering the rectangle regions of the variable patterns. We place a guard at W and the entire polygon is guarded. We thus end up with a minimum cover of $K = 3m + n + 1$ guards.

We must now show that if a cover for our polygon exists of size $K = 3m + n + 1$, then a $3SAT$ instance C is satisfiable. If we have a cover of that size, we know that W exists so we only concern ourself with the remaining $3m + n$ vertices. In each clause, we have three literal patterns. We also have n variable patterns. This gives us $3m + n$ distinguished points. We know that any vertex that covers a distinguished point can not cover other distinguished points. Therefore each distinguished point can only be covered by one vertex.

However, we can not simply place guards randomly to cover each of these distinguished points. We must be certain that the consistency property is held. All of the consistency patterns in one of the two rectangles (see Figure 10) are covered by the $3m$ vertices from the literal patterns. Let us assume that u_i is assigned false in one clause and true in another. This will leave a certain spike unguarded in one of the rectangles. For the guarding to be consistent, all of the spikes in rectangle must be covered by one of the $3m$ guards. All of the spikes in the other rectangle must be guarded by either t_{i5} or t_{i8}. This keeps our choices consistent. If we have guards that are correct, we look at the n variable patterns. If t_{i5} is a guard, the variable u_i is true. Otherwise u_i is false.

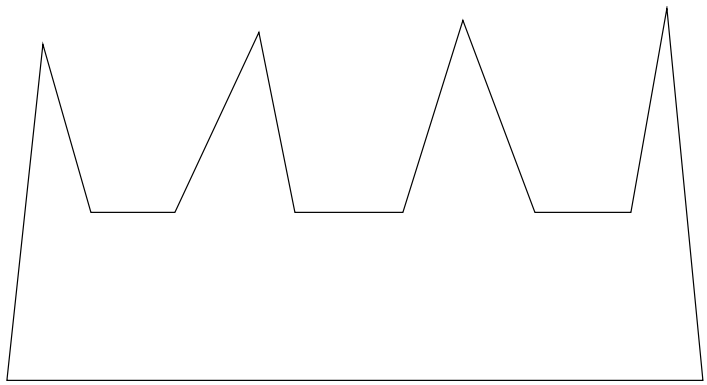

Figure 11: Chvatal's Art Gallery Theorem Example

4.2 Recent Art Gallery Results

The art gallery problem, along with being NP-complete, has also been shown to be APX-hard in [4]. This means that there exists a constant $\epsilon > 0$ such that no polynomial time algorithm can guarantee an approximation ratio of $1 + \epsilon$ unless $P = NP$. Ghosh provides a O($logn$)-approximation for the minimum vertex guard cover where guards can only be placed at the vertices of the art gallery in [6]. The point guarding problem seems to be much harder than the vertex guarding problem and precious little is known about it. Art gallery problems where the polygon is x-monotone have been shown to have a 12-approximation by Nilsson in [10]. Based on his result Nilsson also provides a $O\left(OPT^2\right)$ approximation for rectilinear polygons.

4.3 Chvatal's Art Gallery Theorem

The question of how many guards are needed or necessary to guard an entire polygon was proposed by Victor Klee in 1973. Vasek Chvatal soon after established a solution to what we know as Chvatals Art Gallery Theorem. An obvious answer would be n guards. Simply place a guard at every corner and we will certainly see the entire polygon. However, Chvatal showed that $n/3$ is an upper bound on the number of guards needed to guard a simple polygon and this bound is sometimes necessary. To see why $n/3$ is sometimes necessary, consider the polygon in Figure 11.

The polygon requires four guards with n=12. It is also an upper bound on the number of guards. Chvatal first noted this in [1]. However, Fisk's proof in [5] is much simpler and easier to understand. The first step in Fisk's proof was to triangulate a polygon by adding internal diagonals. The

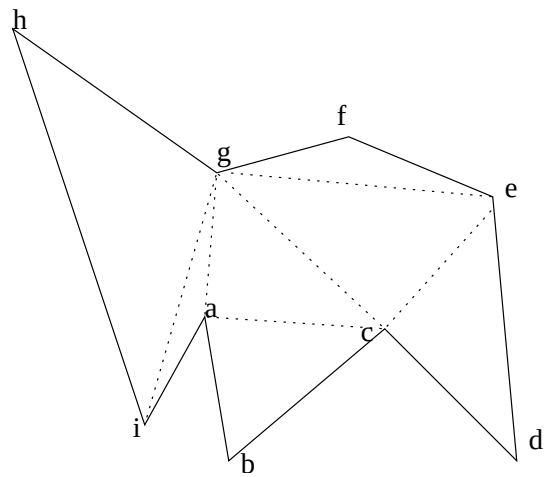

Figure 12: n/3 bound

triangulation theorem in [11] states that a polygon of n vertices may be partitioned into $n-2$ triangles by the addition of $n-3$ internal diagonals. We take any polygon and triangulate it so it contains $n-2$ triangles. An example of this is shown in the Figure 12.

It is easy to see that the vertices of a polygon that have been broken up into triangles can easily be 3-colored. A k-coloring of a graph is simply an assignment of colors to vertices such that no adjacent vertices have the same color. Consider Figure 12. We can pick any arbitrary triangle and 3-color it however we want. By doing so, we then force the remaining vertices to be picked in a certain way. For example we pick the triangle (a, c, g). We color it $1, 2$ and 3 accordingly. This then forces e to be 1, d to be 3, f to be 2 and so on. Since our polygon has no holes and is simple, we are able to 3-color the polygon. The last step is to notice that at least one of the colors is not used more than $\frac{1}{3}$ of the time. Let a, b and c be the number of occurrences of the different colors. We can say that $a \leq b \leq c$. The total number of nodes is n so $a+b+c=n$. Therefore $a \leq \frac{n}{3}$. Let us say color 1 was the least used color. The last step is to then place guards at each vertex colored 1. Every triangle has three nodes and therefore every triangle contains three colors, $1, 2$ and 3. Each vertex is thus guarded by some guard. It follows that for any polygon, $\frac{n}{3}$ vertices is sufficient to cover it.

5 Conclusion and Future Work

There are many open problems in the art gallery and terrain guarding area that are worth considering. The most obvious one I would like to consider is whether the vertex guarding version of terrain guarding is NP-complete. Many attempts have been made to show NP-completeness but many of the reductions get hung up by the order claim. However, attempts to show a polynomial time algorithm have been hard as well. If terrain guarding is shown to be NP-hard, it would be interesting to study if the problem is APX-hard or if there is a PTAS available. If the problem is APX-hard, it would be nice to lower the constant factor approximation as much as possible.

On the art gallery side a little more is known. The general art gallery problem is NP-complete. However, no NP-completeness proof has been provided for the guarding of monotone polygons. It would be interesting to see if the guarding of monotone polygons is indeed NP-complete and then possibly APX-hard. If so, how low can the approximating constant get? If not, is there a $1 + \epsilon$ approximation? It would be interesting to study the possibility of constant factor approximations for both the point guarding problem and vertex guarding problem and seeing how low the constant factor could get.

References

[1] V. Chvatal. *A combinatorial theorem in plane geometry*. J. Combinatorial Theory Series B, vol. 18, 39-41, 1975.

[2] D. Z. Chen, V. Estivill-Castro, and J. Urrutia. *Optimal guarding of polygons and monotone chains (extended abstract)*. 1996.

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

[4] S. Eidenbenz. *Inapproximability Results for Guarding Polygons without Holes*. Lecture Notes in Computer Science, vol. 1533 (ISAAC'98), 427-436, 1998.

[5] S. Fisk. *A short proof of Chvatal's watchman theorem*. J. Combinatorial Theory Series B, vol. 24, 374, 1978.

[6] S. Ghosh. *Approximation algorithms for art gallery problems*. Proc. Canadien Information Processing Society Congress, 1987.

[7] J. King. *A 4-Approximation Algorithm for Guarding 1.5-Dimensional Terrains*. Lecture Notes in Computer Science (3887), 629-640, 2006.

[8] D. Lee and A. Lin. *Computational complexity of art gallery problems*. IEEE Trans. Inform. Theory, vol. 32, 276-282, 1986.

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

[10] B. Nilsson. *Approximate guarding of monotone and rectilinear polygons*. Proceedings of ICALP 2005, 1362-1373, 2005.

[11] J. O'Rourke. *Art Gallery Theorems and Algorithms*. The International Series of Monographs on Computer Science. Oxford University Press, New York, NY, 1987.