Fall 2009

Surveilling roads and protecting art

Erik Allyn Krohn
University of Iowa

SURVEILLING ROADS AND PROTECTING ART

by

Erik Allyn Krohn

An Abstract

Of a thesis submitted in partial fulfillment of the
requirements for the Doctor of Philosophy
degree in Computer Science
in the Graduate College of
The University of Iowa

December 2009

Thesis Supervisor: Associate Professor Kasturi Varadarajan

ABSTRACT

Placing security cameras in buildings, finding good locations for cameras to enforce speed limits or placing guards to defend a border are some of the problems we face everyday. A nation that wishes to defend its border with armed guards wants to be sure the entire border is secure. However, hiring more guards than necessary can be costly. A start-up company moving into a new building wants to be sure every room in the building is seen by some security camera. Cameras are expensive and the company wants to install the smallest number of cameras; at the same time the company wants to be sure the building is secure.

These problems, and many other visibility type problems, are not easy to solve in general. In some specific cases, optimal solutions can be obtained quickly. In general, finding an optimal solution may take a very long time.

The original results of this thesis address some of these problems. We show some positive results for solving some of these visibility problems. We also give some negative results for some of these problems. These negative results are useful because they tell us that we are unlikely to find a fast algorithm to solve a particular problem optimally.

Abstract Approved: _____

Thesis Supervisor

_____

Title and Department

_____

Date

SURVEILLING ROADS AND PROTECTING ART

by

Erik Allyn Krohn

A thesis submitted in partial fulfillment of the
requirements for the Doctor of Philosophy
degree in Computer Science
in the Graduate College of
The University of Iowa

December 2009

Thesis Supervisor: Associate Professor Kasturi Varadarajan

Graduate College
The University of Iowa
Iowa City, Iowa

CERTIFICATE OF APPROVAL

PH.D. THESIS

This is to certify that the Ph.D. thesis of

Erik Allyn Krohn

has been approved by the Examining Committee for the thesis requirement for the Doctor of Philosophy degree in Computer Science at the December 2009 graduation.

Thesis Committee: _____
Kasturi Varadarajan, Thesis Supervisor

_____
Sriram Pemmaraju

_____
Steve Bruell

_____
Juan Pablo Hourcade

_____
Laurent Jay

To Jim Peterson

ACKNOWLEDGEMENTS

Even though my name is on the front of the document, there are many people who helped shape this thesis. I want to first thank my advisor, Kasturi Varadarajan, for his guidance and contributions throughout my time at Iowa. This thesis would not have happened without his constant encouragement and support. I would also like to thank Bengt Nilsson for his contributions and help with the monotone polygon results. He provided many creative ideas that got many of the monotone polygon algorithms off the ground. I also want to thank the members of my reading group, both past and present. Matt Gibson, Gaurav Kanade and Imran Pirwani lent me their ears and listened to many strange ideas about terrains and art galleries and gave valuable input. Matt and Gaurav were also instrumental in obtaining the $(1 + \epsilon)$-approximation for terrain guarding. I want to thank James King for his help with the NP-hardness result for terrain guarding. I want to thank each of my committee members, Sriram Pemmaraju, Steve Bruell, Juan Pablo Hourcade and Laurent Jay for valuable input and encouragement along the way. I want to thank Teo Rus for his support in my early graduate career. Teo helped me get into the PhD program and passed the first hurdle of the qualifying exam. I want to especially thank Alberto Segre for many different reasons. The numerous conversations we had and his trust in me as a student and a teacher are much appreciated.

Lastly I want to thank my friends and family for encouragement and reassurance the past few years. My wife, Jennifer, has been a constant source of support.

For that I am extremely grateful.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

CHAPTER 1
INTRODUCTION

The terrain guarding problem and the art gallery problem are two problems in computational geometry. Both problems are geometric set covering problems. A brief introduction to the set cover problem will be given in Section 1.1. The geometric set cover problem will be introduced in Section 1.2. An introduction to the terrain guarding problem will be covered in Section 1.3. The art gallery problem will be introduced in Section 1.4.

1.1 Set Cover

In the set cover problem, we are given as input a collection of subsets $\mathcal{S}$ of another set U. We are also given a set $M \subset U$ that we wish to cover. We say that a collection of subsets S *covers* M if for every element $m \in M$, there exists a subset $S' \in S$ such that $m \in S'$. We will assume that $\mathcal{S}$ covers M.

The decision version of the set cover problem also takes an integer k as input and asks if there are k such subsets from $\mathcal{S}$ that cover M. In other words, does there exist a collection of subsets $S \subseteq \mathcal{S}$ such that S covers M and $|S| \leq k$?

Consider the following example:

- $U = \{1, 2, 3, 4\}$
- $M = \{2, 4\}$
- $S_1 = \{1, 4\}$

- $S_2 = \{2, 3\}$
- $S_3 = \{3, 4\}$
- $S_4 = \{2\}$
- $S_5 = \{1, 3\}$
- $\mathcal{S} = \{S_1, S_2, S_3, S_4, S_5\}$
- $k = 1$

In this example, we ask if M can be covered with 1 set chosen from the set $\mathcal{S}$. In this example, there does not exist an $S \in \mathcal{S}$ such that S covers M. Therefore the output would be NO. However, if k is changed to 2, the output would be YES. A possible solution with 2 sets chosen from $\mathcal{S}$ is $\{S_1, S_2\}$. In the minimization version of the set cover problem, the input is the same except we are not given the integer k. The goal of the minimization version is to find the smallest value of k such that the answer to the corresponding decision version is YES.

For an instance of the set cover problem, let n be the size of the input. We say that n is the number of elements in M plus the number of subsets in $\mathcal{S}$. A polynomial time algorithm is an algorithm whose running time is upper bounded by a polynomial in n.

The set cover problem is NP-hard. A problem that is NP-hard is unlikely to have a polynomial time algorithm that solves every instance of the problem optimally unless $P = NP$. A polynomial time approximation scheme ($PTAS$) is an algorithm

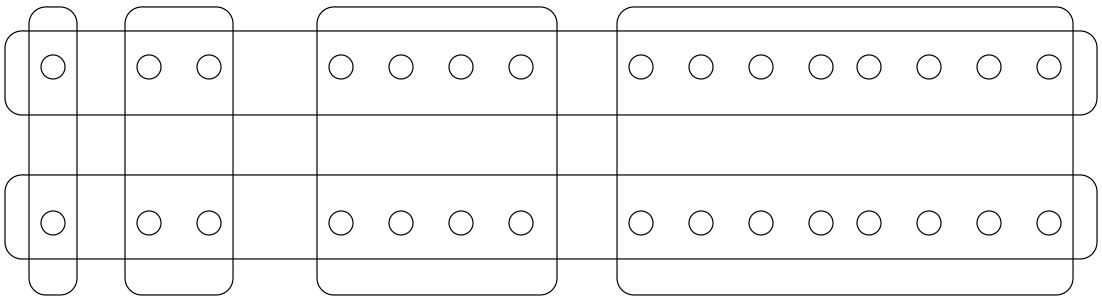


Figure 1.1: Example where a greedy algorithm performs poorly.

that takes an instance of a problem and a parameter $\epsilon > 0$ and in polynomial time produces a solution within an ϵ factor of the optimal solution. Some problems that are NP-hard have been shown to be APX-hard. An APX-hard problem is a problem that does not admit a $PTAS$ unless $P = NP$.

An approximation algorithm is an algorithm that will guarantee the solution obtained is within a certain range. For example, saying an algorithm for a given problem A is a 5-approximation means that for every instance of problem A, the algorithm will output a solution that is within a factor of 5 of OPT. Here, OPT is the optimal value for a given problem instance. A $PTAS$ is a $(1 + \epsilon)$-approximation where $\epsilon > 0$ is given with the input.

The set cover problem has been shown to be NP-hard [27]. The problem has been shown to be hard to approximate [37] within a $c*ln(n)$ [40, 3] by any polynomial time algorithm where c is some constant unless $P = NP$. For the set cover problem, a simple greedy algorithm performs very close to the best possible polynomial time algorithm. A greedy algorithm may return a solution that is within a $log(n)$ factor of the optimal solution. To see how a greedy algorithm can fail to produce an optimal

solution, consider the following problem:

- $U = \{1, 2, 3, ..., 13, 14\}$
- $M = U$
- $S_1 = \{1, 2, 3, 4, 5, 6, 7\}$
- $S_2 = \{8, 9, 10, 11, 12, 13, 14\}$
- $S_3 = \{4, 5, 6, 7, 11, 12, 13, 14\}$
- $S_4 = \{2, 3, 9, 10\}$
- $S_5 = \{1, 8\}$
- $\mathcal{S} = \{S_1, S_2, S_3, S_4, S_5\}$
- $k = 1$

Each element in M is initially uncovered. The greedy algorithm chooses the set $S \in \mathcal{S}$ that contains the largest number of uncovered elements. The optimal solution is to choose S_1 and S_2. In the example, the greedy algorithm chooses S_3 to be in the solution first. This leaves $\{1, 2, 3, 8, 9, 10\}$ uncovered. The algorithm then chooses S_4 leaving $\{1, 8\}$ uncovered. The algorithm finally chooses S_5. An instance of this problem is easy to see in figure form. In Figure 1.1, the optimal solution is to take the 2 long "skinny" horizontal sets but each time the "fatter" sets to the right are chosen. The optimal solution is 2 but here the greedy algorithm chooses 3 sets.

For this example in general, the number of elements in the fat rectangles from left to right is $2, 4, 8, 16, ..., 2^{n'/2}$ where n' is the number of remaining uncovered elements. Initially $n' = n$. The key thing to note is that rightmost fat rectangle will always be of size $2^{n'/2}$. The size of the 2 skinny rectangles will always be $2^{n'/2} - 1$. Therefore the greedy algorithm will always choose the rightmost fat rectangle and give a solution with $\Omega(log(n))$ rectangles when 2 is the size of the optimal solution.

1.2 Geometric Set Cover

The geometric set cover problem is the set cover problem in some geometric setting. In general, the geometric set cover problem contains as input an integer d, a value k, a collection $\mathcal{S}$ of subsets of $\Re^d$; it is implicit that $U = \Re^d$. The decision version of the geometric set cover problem is to answer YES or NO to the question of whether k subsets chosen from $\mathcal{S}$ cover M where $M \subset U$. The minimization problem is to find the smallest such k such that the answer to the decision problem is YES.

Even in the geometric setting, many versions of the problem are believed to be NP-hard [33, 31, 32]. The focus of current work is to find a polynomial time algorithm that guarantees a "good" approximation. For many such geometric set cover problem, one can obtain a polynomial time algorithm that guarantees a $O(log(n))$ approximation by reducing the problem to the combinatorial set cover problem [10, 26, 36].

Consider the following example: Let M be a collection of points in the plane, $\Re^2$. We let S be a collection of rectangles in the plane and k be some integer. The problem is to decide if it is possible to cover M with k rectangles chosen from set

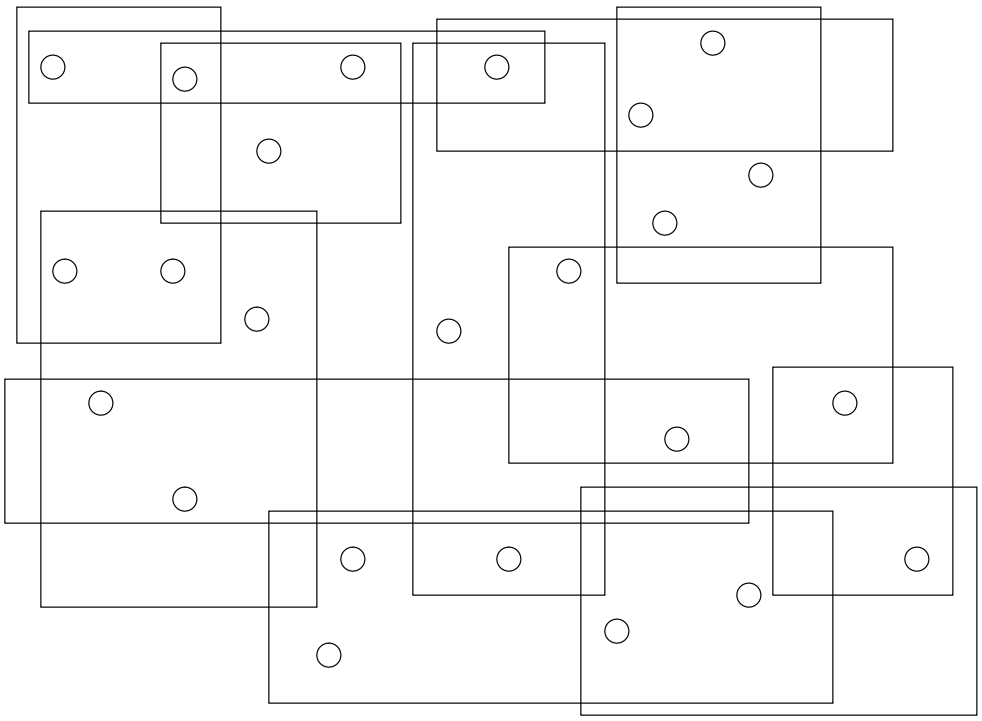


Figure 1.2: A geometric set cover example.

S. We say a rectangle $S' \subset S$ covers an element $m \in M$ if m is inside S'. An example is shown in Figure 1.2. The optimal solution is shown in Figure 1.3 by using 8 rectangles.

Some geometric set cover problems, like the combinatorial set cover problem, can perform badly using a greedy algorithm. An example was already shown in Figure 1.1.

The goal of many algorithms for geometric set cover problems is to exploit the geometry and structure of the problem to obtain a solution better than a simple greedy algorithm. Exploiting the structure of these problems can produce constant factor approximations [32], a PTAS [23] or even an algorithm that optimally solves

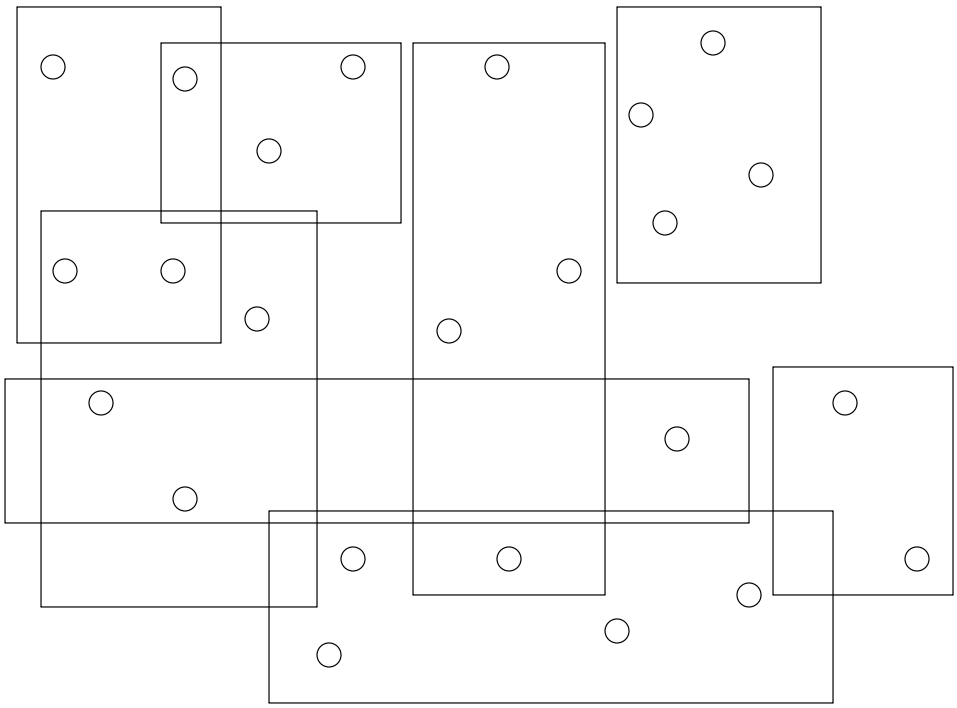


Figure 1.3: A solution to Figure 1.2.

the problem in polynomial time [8]. The following sections describe two problems that will be explored in detail in this thesis: the terrain guarding problem and the art gallery problem.

1.3 Terrain Guarding

An instance of the terrain guarding problem contains a terrain T that is an x-monotone polygonal chain. An x-monotone polygonal chain has the property that any vertical line intersects the chain at most once. The terrain guarding problem gets as input a set of points $P = \{v_1, v_2, ..., v_n\}$. A point v_i is defined with coordinates (x_i, y_i). The points are ordered from left to right such that $x_i < x_{i+1}$. An edge e_i is defined as a straight line connecting v_i to v_{i+1}. The edge set $E = \{e_i | 1 \leq i \leq n-1\}$.

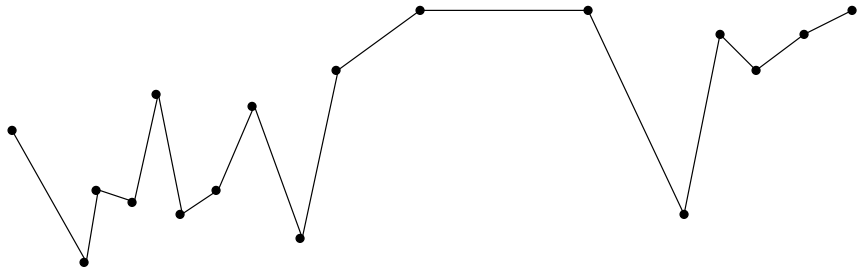


Figure 1.4: Example terrain guarding instance.

The terrain T consists of the points in P and the points in all of the edges in E. For any two points $p, q \in T$, we say that p *sees* q if the line segment $\overline{pq}$ lies entirely above or on T. 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.

1.3.1 Discrete Terrain Guarding

In the *discrete terrain guarding problem*, along with the set P, we are given a finite set G of candidate guards and a set X of points that need to be seen where $X, G \subseteq T$. The decision version of the problem also gives an integer k as input. The output to the decision problem is either YES or NO. The output is YES if there exists a $G' \subseteq G$ with $|G'| \leq k$ so that every point in X is seen by at least one guard in G'. We call this set G' our guarding set. The minimization version of the problem tries to find the smallest such k where the answer to the decision problem is YES.

As an example, consider the terrain in Figure 1.4. In this terrain, the black dots are the set X and also the set G. The minimization problem wants to find the smallest such $G' \subseteq G$ such that every point in X is seen by at least one guard in G'.

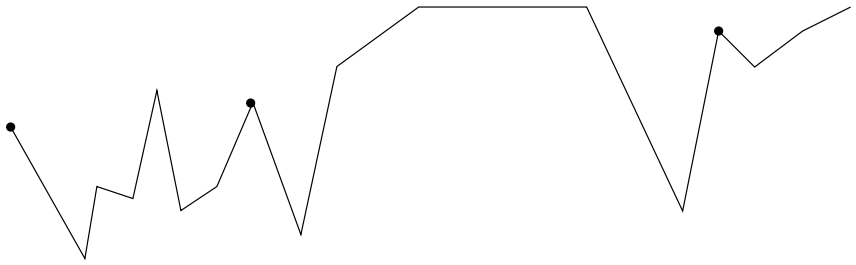


Figure 1.5: Example discrete terrain guarding solution for Figure 1.4.

As shown in Figure 1.5, the minimum number of guards required is 3. It is important to note that in the discrete version of the terrain guarding problem, not all of T needs to be guarded; only the points in X need to be guarded.

1.3.2 Continuous Terrain Guarding

In the *continuous terrain guarding problem*, all points on the terrain T are eligible guard locations. However, in this problem, all points in T must be guarded. More formally, $X = G = T$. The minimization problem wants to find the smallest such $G' \subseteq T$ such that every point in X is seen by at least one guard in G'. Consider the example from Figure 1.6. In this Figure, all points on the terrain are potential guards. The smallest number of guards required to see the entire terrain is 4 and this solution is shown in Figure 1.7.

There are several things worth nothing about solutions to the continuous problem, $G = X = T$, and a common version of the discrete problem where $G = X = P$. A solution Z to the discrete version is not necessarily a solution to the continuous version. In Figure 1.5, all vertices of the terrain are seen but a small portion of the terrain is unseen. This is emphasized in Figure 1.8. An optimal solution Y to the

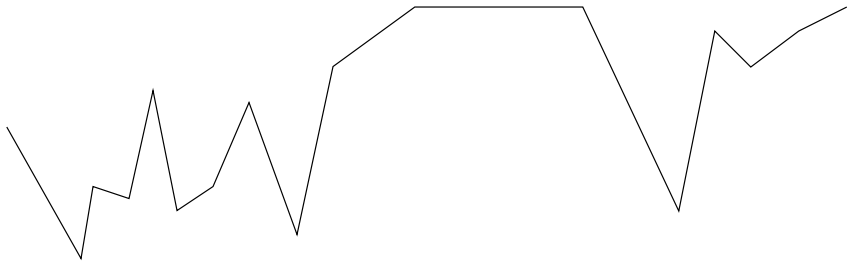


Figure 1.6: Example continuous terrain guarding instance. All points on the terrain are candidate guards.

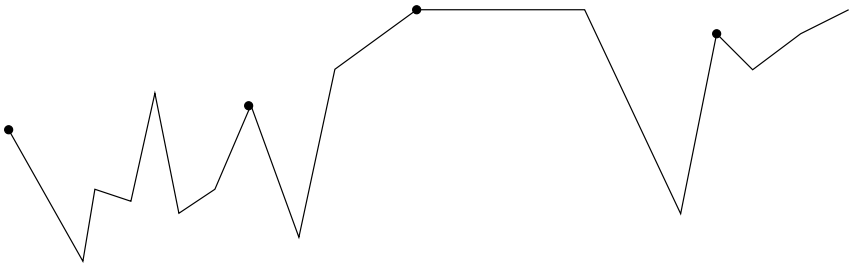


Figure 1.7: Continuous terrain guarding solution to Figure 1.6.

continuous version where all guards in Y belong to P is clearly a solution to the discrete version. However, Y may not be an optimal discrete solution. This is seen clearly in Figures 1.5 and 1.7.

Consider an instance with Z being an optimal discrete solution and Y being an optimal continuous solution. Considering the previous example, it may the case that $Z \leq Y$ or that all optimal solutions for the continuous problem contain guards only at vertices in P. However, it is possible that $Z > Y$. It is also possible that the optimal solution for the continuous problem contains guards that are not in P. These claims can be seen in Figure 1.9. In can be easily verified that no 1 vertex sees every other vertex. This example shows that $|Y| = 1$ where $|Z| = 2$.

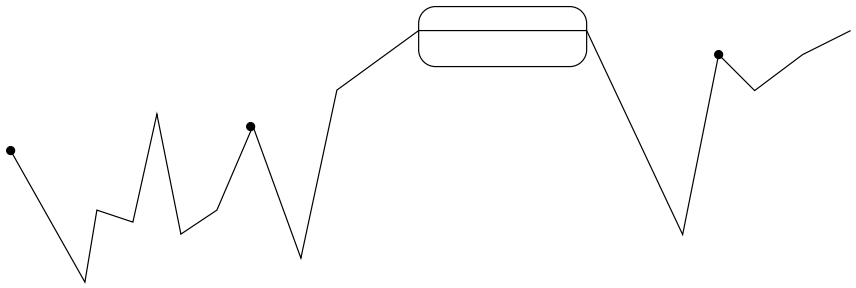


Figure 1.8: A solution to the discrete problem is not necessarily a solution to the continuous problem. The boxed in portion of the terrain is not seen by any guard.

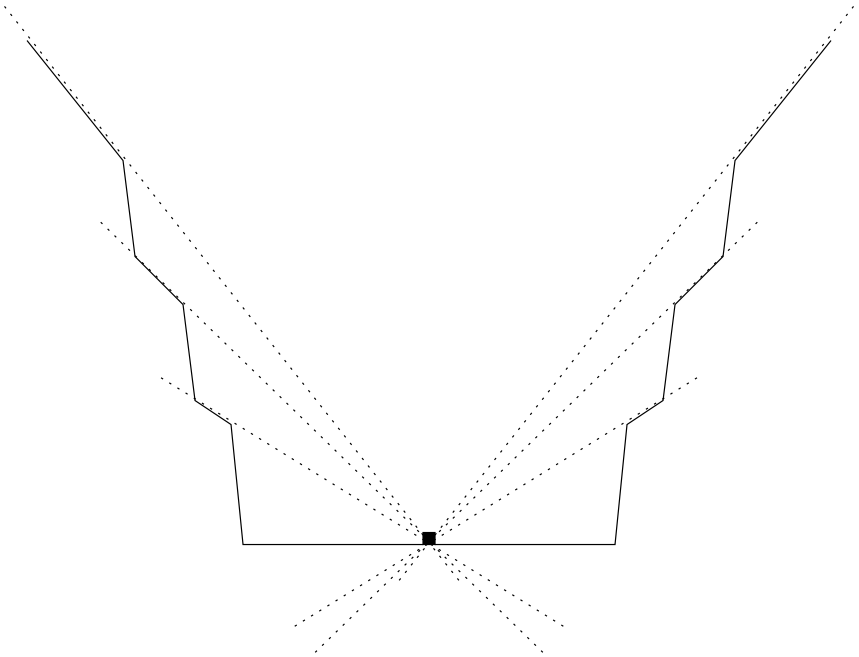


Figure 1.9: Terrain showing the optimal continuous solution contains less guards than the optimal discrete solution. Dotted lines show visibilities.

1.3.3 Motivations

Motivation for guarding terrains come from scenarios that include covering a road with street lights or cameras. The goal for these applications is to find locations to place lights to maximize light coverage. Another goal could be to find good locations for cameras to help enforce speed limits. Other applications include finding a configuration for line-of-sight transmission networks for radio broadcasting, cellular telephony and other communication technologies [4].

1.3.4 Previous Results

The terrain guarding problem has received considerable attention in the last few years from the point of view of approximation algorithms. The first constant factor approximation that ran in polynomial time for the terrain guarding problem was shown by Ben-Moshe *et al.* in [4]. The algorithm presented in [4] initially divides the terrains into independent pieces called subterrains. A property of these subterrains is that every point in the subterrain is seen by some guard outside the subterrain. For each subterrain that is not completely guarded by guards outside of the subterrain, then that subterrain is either reduced to a smaller subterrain or is split up into several subterrains. The algorithm is a fairly sophisticated combinatorial approach. No analysis was made to show a small constant factor but it was proposed to be as low as 6 in [28].

Clarkson and Varadarajan give a constant factor approximation in [11] based on solving a linear programming relaxation. They partition the terrain into maximal

intervals such that for two points p and q in an interval, the leftmost point that sees p and the leftmost point that see q are the same. Ordering the intervals from left to right, they note that we end up with a $(r, 2)$ Davenport-Schinzel sequence [41]. Such a sequence has a length at most $2r$. They use this to find appropriately sized nets. They are able to apply a method of [6] to round the linear program using these nets and obtain a constant factor approximation. No analysis was made to show how small the constant factor could be.

A 4-approximation was proposed by King in [29] but further analysis increased the approximation factor to 5 [28]. The algorithm worked by finding an unguarded point u and a set of points S that would dominate any optimal guard $g(u)$ for u. In other words, every point that was seen by $g(u)$ is also seen by some guard in S. As further analysis showed, $|S|$ was increased to 5 instead of 4.

In [18], Elbassioni et al. give a 4-approximation for the terrain guarding problem in the weighted case. In previous approximations, the unweighted case was considered. They decompose the constraint matrix of the linear program into totally balanced matrices.

One of the interesting things about this problem is that despite the numerous approximations that appeared for the problem, it was not known whether the problem was NP-hard. Chen et al. [8] claimed that terrain guarding is NP-hard, but the proof was never completed formally [29].

1.4 Art Galleries

The history of the art gallery problem can be traced back to a question that Victor Klee posed during a conference at Stanford in 1976:

How many guards are always sufficient to guard a polygon with n vertices?

Before we get to the answer, we first define what the art gallery problem is. The input to the art gallery problem is a set of vertices $V = \{v_1, v_2, ..., v_n\}$. A single point v_i is defined with x and y coordinates, (x_i, y_i). The polygon P is defined by the vertices in V along with a set E of edges. An edge is given by specifying two of the vertices and is interpreted as the line segment connecting the two vertices. If the polygon is simple, the edges are (v_i, v_{i+1}), $1 \leq i \leq n-1$ along with (v_n, v_1).

The art gallery problem can come in many different flavors. Two common versions are the *point (interior) guarding* version and the *vertex guarding* version. Along with these versions, we can impose restrictions on what type of polygon is accepted as input. Polygons could be monotone, rectilinear or have holes [24], see Figure 1.10. Polygons with holes have an added set of vertices given as input to define where the holes are located.

The edges shown in Figure 1.10 give us two disjoint regions: inside the polygon and outside the polygon. The boundary for P is the points in V and the points in all of the edges of E. The interior of the polygon defines the rest of the polygon P. For any two points $p, q \in P$, we say that p sees q if the line segment $\overline{pq}$ does not go outside the polygon P.

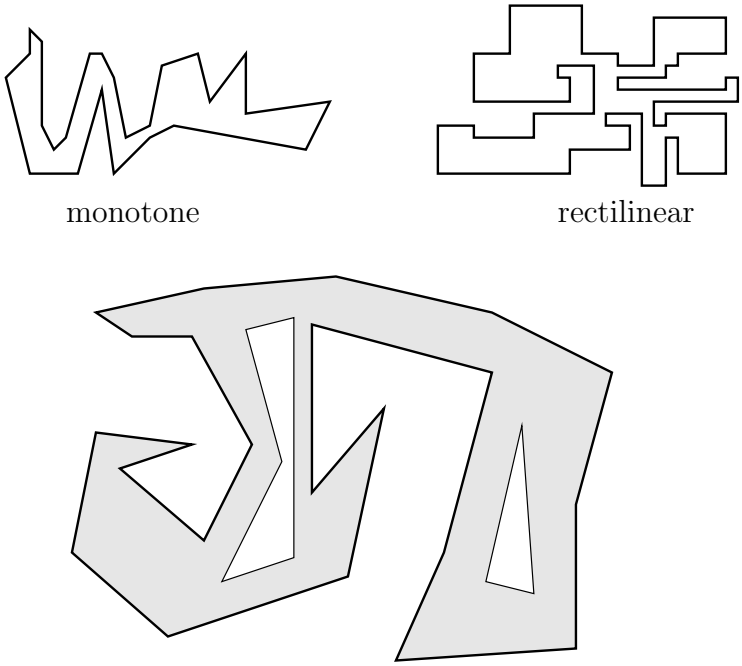


Figure 1.10: Example with different polygon types.

The *vertex guarding* version of the problem forces guards to be placed at points in V. The decision version of the problem accepts as input a polygon P, a positive integer k and asks if there exists a guardset $G \subseteq V$ such that $|G| \leq k$ and every point $p \in P$ is seen by at least one guard in G. If such a guardset G exists, the output is YES. If no guardset exists, the output is NO. The minimization version of the problem tries to find the smallest such k for which the answer to the decision problem is YES.

As an example, consider Figure 1.11. To vertex guard this polygon we need to place at least 4 vertex guards. Guards are shown as black dots on the vertex they represent. It can be easily verified that no set of 3 vertex guards can guard this

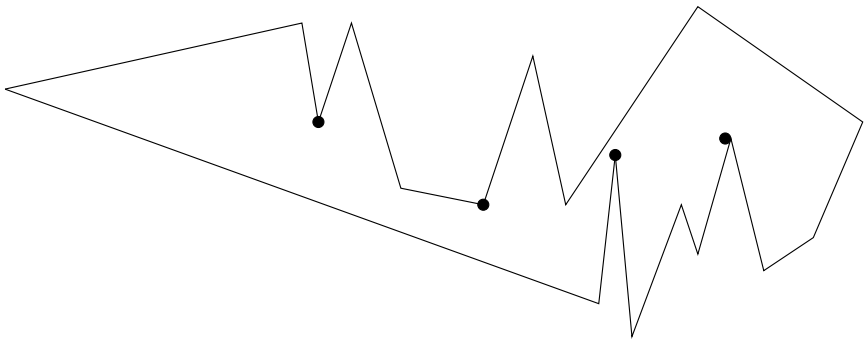


Figure 1.11: An example monotone polygon.

polygon.

The *point guarding* version of the problem has little restriction on guard placement. A guard may be placed anywhere on the boundary of P or on the interior of P. A guard may not be placed outside of P. The decision version of this problem accepts as input a vertex set V, a positive integer k and asks if there exists a guardset $G \subseteq P$ such that $|G| \leq k$ and every point $p \in P$ is seen by at least one guard in G. If such a guardset G exists, the output is YES. If no guardset exists, the output is NO. The minimization version of the problem tries to find the smallest such k for which the answer to the decision problem is YES.

As an example, consider Figure 1.12. To point guard this polygon we need to place at least 3 guards. Guards are shown as black dots. It can be easily verified that no set of 2 guards can guard this polygon.

Consider a vertex guarding solution X and a point guarding solution Y for some polygon P. We can say that $|X| \geq |Y|$. In this situation, a vertex guarding solution is a point guarding solution. At worst, our Y solution will choose guards

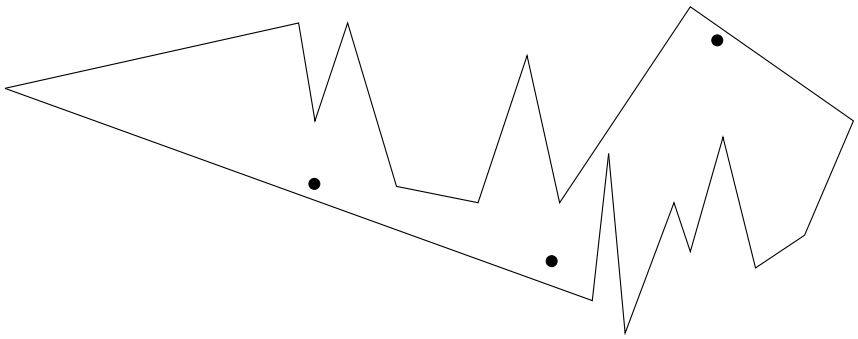


Figure 1.12: An example point guarded polygon.

consistent with X and therefore can only find better solutions.

1.4.1 Motivations

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

1.4.2 Art Gallery Results

The question of whether guarding simple polygons is NP-hard was settled by Aggarwal [2] and Lee and Lin [33] independently. They showed that the problem is NP-hard for both vertex guards and interior guards. Further results have shown that guarding a restricted subclass of polygons is still NP-hard [5, 39]. Along with being NP-complete, the art gallery problem was shown to be APX-hard in [16]. This means that there exists a constant $\epsilon > 0$ such that no polynomial time algorithm can guarantee an approximation ratio of $(1 + \epsilon)$ unless $P = NP$.

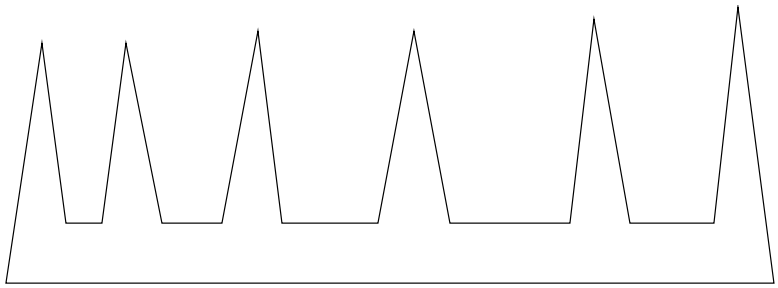


Figure 1.13: $\frac{n}{3}$ guards are necessary to guard this polygon.

The approximation complexity of guarding polygons has been studied by Eidenbenz and others. Eidenbenz [15] 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 [5, 16] independently prove that interior guarding simple polygons is APX-hard.

Tight bounds for the number of guards necessary and sufficient were found by Chvátal [9] and Fisk [20]. It is sometimes necessary to place $\frac{n}{3}$ guards to guard the entire polygon. To see an example of this, see Figure 1.13. In this example, the polygon is shaped like a comb with many spikes. Each spike requires a unique guard; in other words no one guard can see 2 of these spikes completely. It is easy to see that each of these spikes are loosely defined by 3 vertices.

Fisk provided a simple proof in [20] that broke up any polygon into a set of triangles and showed that this set of triangles can be 3-colored which implies that $\frac{n}{3}$ guards are sufficient for guarding a simple polygon.

Any polygon can be efficiently vertex guarded with a $O(log(n))$ approxima-

tion factor where n is the number of vertices in the polygon. The algorithm is a simple reduction to the combinatorial set cover problem. This reduction was shown by Ghosh in [22]. This result can be improved for simple polygons using randomization, giving an algorithm with expected running time $O(nOPT_v^2 \log^4 n)$ 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 [14]. Whether a constant factor approximation can be obtained for vertex guarding a simply polygon is a longstanding and well-known open problem. Deshpande *et al.* [13] present a pseudopolynomial randomized algorithm for finding a point guard cover with approximation factor $O(\log OPT)$. The point guarding problem seems to be much harder than the vertex guarding problem and precious little is known about it [13].

1.5 Contributions

This thesis shows original results obtained for both the terrain guarding problem and the art gallery problem.

1.5.1 Terrain Guarding Results

The sequence of approximation algorithms for the terrain guarding problem begs the question of whether the problem is NP-hard in the first place. The answer to this question is unclear at the outset because straightforward attempts to show NP-hardness run up against the following "Order Claim":

Claim 1. *Let a, b, c, d be four points on the terrain in increasing order of x-coordinates. If a sees c and b sees d, then a sees d.*

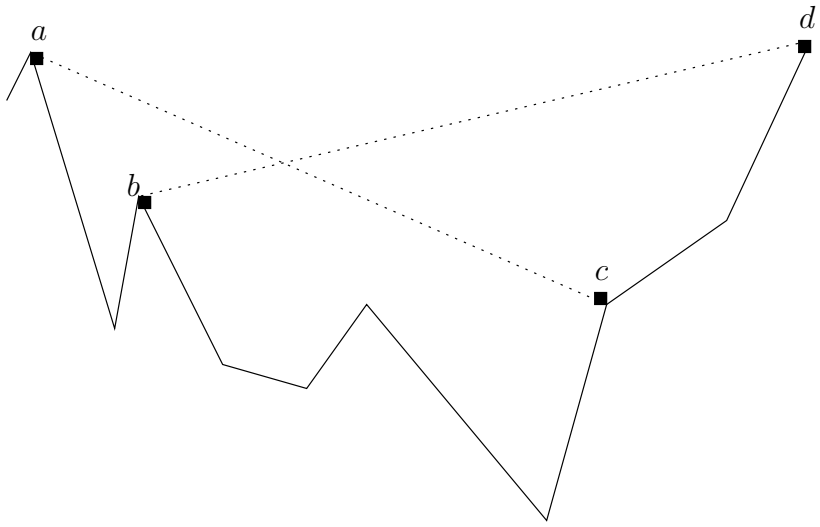


Figure 1.14: Example showing the order claim.

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 left-to-right ordering of a, b, c, and d, it is clear to see that no point can lie above the line segment $\overline{ad}$. Therefore *a sees d*. This claim is shown in Figure 1.14. □

Natural attempts at constructing NP-hardness reductions for the terrain guarding problem do not work because of the order claim. Lee and Lin provide a proof in [33] that showed the art gallery problem was NP-complete. According to Demaine and O'Rourke [12], the complexity of the terrain guarding problem was posted by Ben-Moshe. We quote from [12]:

> "What is the complexity of computing the guard set of minimum size for a given x-monotone chain in the plane? According to the poser, most tenured professors think the problem is NP-hard. This problem in fact goes back to 1995, when Chen *et al.* [8] claimed an NP-hardness result,

but 'the proof, whose details were omitted, was never completed successfully'" [29].

In this thesis, we were able to get around the order claim and prove NP-hardness for both the discrete terrain guarding problem and the continuous terrain guarding problem. The NP-hardness result will be shown in Chapter 2[1].

Given that the terrain guarding problem is NP-hard, this motivates finding the best possible approximation. A 4-approximation for the discrete terrain guarding problem is given in Chapter 3[2]. The main building block of the 4-approximation algorithm is an LP-rounding algorithm for one-sided guarding. Guided by these fractional solutions we can partition the set of points to be guarded into two sets, a set of points that is to be guarded from the left and a set of points that is to be guarded from the right. One-sided guarding is a version of the terrain guarding problem that requires points to be guarded to be seen by a guard to the left (or right).

Based on observations made from the 4-approximation, a $(1+\epsilon)$-approximation for the discrete terrain guarding problem was discovered. The $(1+\epsilon)$-approximation is given in Chapter 4[3]. Our $PTAS$ for the terrain guarding problem is based on local search. This work is inspired by the recent work of Mustafa and Ray [38] who showed how to obtain a $PTAS$ for other geometric set cover problems based on local search. To apply the methodology they develop in the terrain guarding context,

[1]Appears in [30].

[2]Appears in [17].

[3]Appears in [23].

we use observations made from the 4-approximation. Given the hardness result of Chapter 2 and the $PTAS$ in Chapter 4, this settles the computational complexity of the terrain guarding problem.

1.5.2 Art Gallery Results

We show that the point guarding version of the art gallery problem where the polygon is x-monotone has an $O(1)$-approximation in Chapter 5[4]. The key idea behind the $O(1)$-approximation is that we incrementally place guards starting from the left side of the polygon and moving right. We then compare our solution to the optimal solution and show that our solution can not be more than an $O(1)$-factor worse than the optimal solution.

Chapter 5 will cover an NP-hardness proof for vertex guarding monotone polygons [32]. The hardness result does not extend to point guarding. Our proof forces guards to be placed at the vertices of our polygon. This hardness result is much simpler than the hardness result for terrain guarding.

[4]Appears in [32].

CHAPTER 2
TERRAIN GUARDING:NP-HARDNESS

2.1 Introduction

In this chapter we show the discrete terrain guarding problem is NP-hard. This result extends to the continuous terrain guarding problem.

2.2 Reduction: Overview

In the discrete terrain guarding problem, the input is a set V of vertices, a set $G \subseteq V$ of candidate guards, and a set $X \subseteq V$ of points to be guarded. Section 2.3 explains the gadgets used in the reduction in detail. A full example of the reduction is given in Section 2.4.

We show a reduction from the planar 3SAT problem. This problem was shown to be NP-complete in [35]. Planar 3SAT is defined as follows: Let $\Phi = (X, C)$ be an instance of 3SAT, with variable set $X = \{x_1, \ldots, x_n\}$ and clauses $C = \{c_1, \ldots, c_m\}$ such that each clause consists of exactly three distinct literals. Define a *formula graph* $G_\Phi = (V, E)$ with vertex set $V = X \bigcup C$ and edges $E = E_1 \bigcup E_2$ where $E_1 = \{(x_i, x_{i+1}) | 1 \leq i \leq n\}$, and $E_2 = \{(x_i, c_j) \mid c_j \text{ contains } x_i \text{ or } \overline{x_i}\}$. A 3SAT formula Φ is called *planar* if the corresponding formula graph G_Φ is planar. The edge set E_1 defines a cycle on the vertices X, and thus divides the plane into exactly 2 faces. Each node $c_j \in C$ lies in exactly one of those two faces. We have to determine whether there exists an assignment of truth values to the variables in X that satisfies all the clauses in the C.

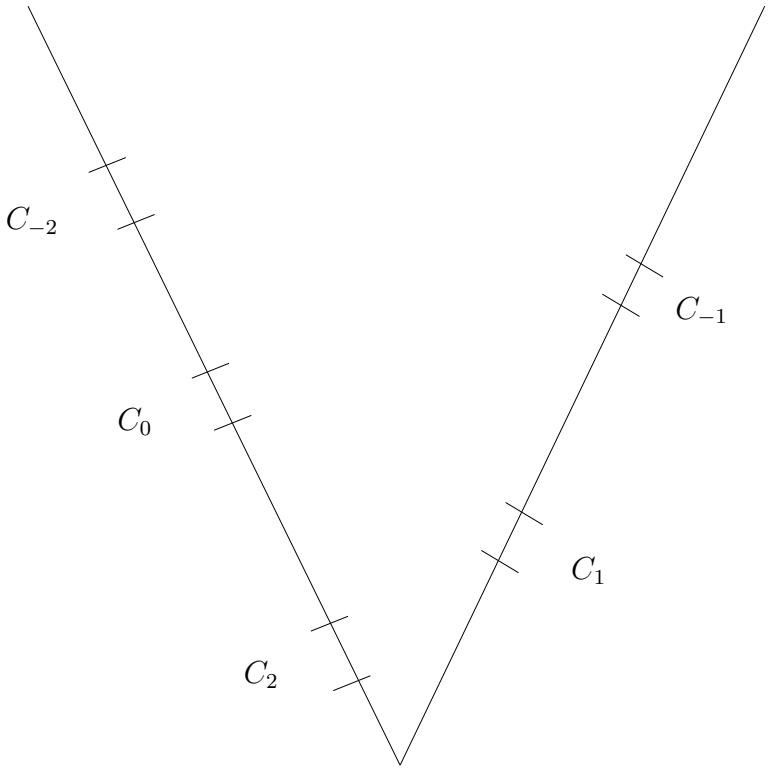


Figure 2.1: A coarse view of a terrain T constructed by our reduction showing chunks C_{-2}, C_{-1}, C_0, C_1 and C_2.

It is easy to see that the clauses inside the variable cycle can be generated by performing a sequence β of steps starting with $\sigma = \langle x_1, \ldots, x_n \rangle$ where at each step we do one of the following until σ becomes empty:

1. Delete a variable from sequence σ and call the resulting variable sequence σ.

2. Generate a clause using three consecutive variables in σ and delete the middle variable from σ. Call the resulting variable sequence σ.

Similarly there is a different sequence α of steps starting from $\sigma = \langle x_1, \ldots, x_n \rangle$ that generates all clauses outside the variable cycle.

The terrain T constructed by our reduction is shaped like a valley. A coarse view of the terrain can be seen in Figure 2.1. We identify disjoint pieces of the terrain called *chunks*. Even indexed chunks, $C_0, C_2, C_4, ..., C_{-2}, C_{-4}, ...$, are on the left side of the terrain and odd indexed chunks, $C_1, C_3, ..., C_{-1}, C_{-3}, ...$, are on the right side of the terrain. Chunks $C_0, C_1, C_2, ..., C_k$ are used to "implement" the sequence β. Chunks $C_0, C_{-1}, C_{-2}, ..., C_{-k'}$ are used to "implement" the sequence α.

Recall that we are considering the discrete terrain guarding problem; we have a finite set of guards. Chunks contain *distinguished points* which are the points to be guarded in our reduction. Chunks also contain a set of potential guard locations. Distinguished points and the set of potential guard locations will be defined in Section 2.3.

Corresponding to each even chunk, C_i is a subsequence λ_i of the sequence of variables $\langle x_1, x_2, ..., x_n \rangle$. There will be $2|\lambda_i|$ guard locations, one for each of the $2|\lambda_i|$ literals corresponding to the variables in λ_i[1]. We will refer to these guard locations by the corresponding literal names. Literal locations x and $\overline{x}$ corresponding to a variable x are consecutive on the chunk but either may be to the left or right of the other. The left to right ordering of literals in a chunk C_i corresponding to different variables is according to λ_i if i is even. If i is odd, the right to left ordering of the literals corresponding to different variables is according to λ_i.

Associated with each chunk C_i will be a number n_i. *In the reduction, n_i guards*

[1]Certain chunks are an exception, some chunks may have $2|\lambda_i| + 2$ literals. Most chunks have 2 literals corresponding to 1 variable; certain chunks may have 4 literals corresponding to 1 variable.

will be needed within chunk C_i to see distinguished points in C_i. If less than n_i guards are placed in chunk C_i, no matter how many guards are placed elsewhere, certain distinguished points in chunk C_i will go unseen. Without loss of generality, we assume that chunk C_0 is placed on the "left" side of the valley. For C_0, $\lambda_0 = \langle x_1, x_2, ..., x_n \rangle$, the literal locations in left-to-right order on the terrain are $x_1, \overline{x_1}, x_2, \overline{x_2}, ..., x_n, \overline{x_n}$ and $n_0 = n$. To guard C_0 using n_0 guards, we will have to place exactly n guards at exactly n of the literal locations, with one guard for each variable or its complement. Note that such a placement of guards specifies an assignment to the variables.

$C_0, C_1, ..., C_k$ are used to implement the sequence β, as we now describe. Suppose that we have added chunks $C_1, ..., C_i$ to implement steps $\beta_1, ..., \beta_j$ of β. Let $\sigma(j)$ refer to the sequence σ after step β_j. By construction, chunk C_i will have $\lambda_i = \sigma(j)$. Suppose β_{j+1} is a step in which we delete variable x from $\sigma(j)$. Chunk C_{i+1} will have $\lambda_{i+1} = \sigma(j) \setminus x$. We will have $n_i = |\lambda_i|$, and $n_{i+1} = |\lambda_{i+1}|$. The relationship between C_i and C_{i+1} will be what we call a *deletion*, which has the following property: to guard C_i and C_{i+1} using $n_i + n_{i+1}$ guards, it is necessary that we have:

1. exactly $n_{i+1} = |\lambda_{i+1}|$ guards at the literals within C_{i+1}, one for each variable so that this corresponds to an assignment to the variables in λ_{i+1};

2. exactly $n_i = |\lambda_i|$ guards at the literals within C_i, one for each variable so that this corresponds to an assignment to the variables in λ_i;

3. The location of the guards must be consistent for all variables except x: There is a guard at literal y in C_i if and only if there is a guard at literal y in C_{i+1}.

Suppose that β_{j+1} is a clause step involving the variables x, y and z. This requires up to two applications of an *inversion gadget* followed by a *clause gadget*. An inversion involving a variable x uses three chunks C_i, C_{i+1}, and C_{i+2}. Its purpose is to change the left to right ordering of literals x and $\overline{x}$ in C_{i+2} to be opposite of that in C_i. If the relationship between C_i, C_{i+1}, and C_{i+2} is an inversion corresponding to x, then $\lambda_{i+2} = \sigma(j)$ is the same as λ_i. We have $n_i = |\lambda_i|$, $n_{i+1} = |\lambda_i| + 1$, and $n_{i+2} = |\lambda_{i+2}|$. To guard C_i, C_{i+1}, and C_{i+2} using $n_i + n_{i+1} + n_{i+2}$ guards, it is necessary that we have:

1. $n_{i+2} = |\lambda_{i+2}|$ guards for C_{i+2}, one for each variable, as above;
2. n_{i+1} guards for C_{i+1};
3. $n_i = |\lambda_i|$ guards for C_i, one for each variable;
4. The location of the guards must be consistent for all variables: There is a guard at literal y in C_i if and only if there is a guard at literal y in C_{i+2}.

Suppose that β_{j+1} is a clause step involving $\overline{x} \vee y \vee \overline{z}$. The variables x, y and z must occur consecutively in either left to right or right to left order in C_i; it must also be the case that the literal $\overline{x}$, the two literals corresponding to y and the literal $\overline{z}$ occur consecutively in either left to right or right to left order in C_i. This can be seen in Figure 2.2. Recall that β_{j+1} also deletes the middle variable y. By construction, chunk C_i will have $\lambda_i = \sigma(j)$. Chunk C_{i+1} will have $\lambda_{i+1} = \sigma(j) \setminus y$. We have $n_i = |\lambda_i|$ and $n_{i+1} = |\lambda_{i+1}|$. The relationship between C_i and C_{i+1} will be what we call a *clause*

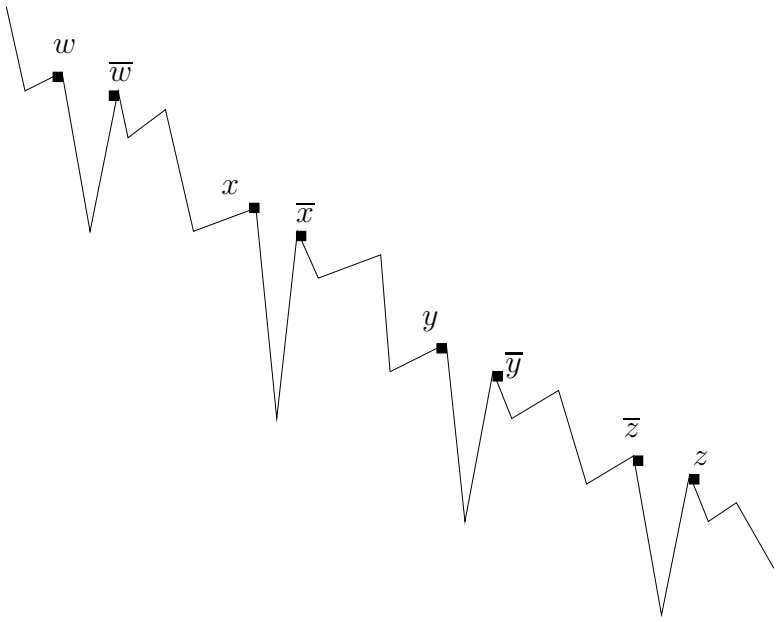


Figure 2.2: Partial chunk showing the ordering of the variables w, x, y, and z.

gadget, which has the following property: to guard C_i and C_{i+1} using $n_i + n_{i+1}$ guards, it is necessary that we have:

1. $n_{i+1} = |\lambda_{i+1}|$ guards for C_{i+1}, one for each variable;

2. $n_i = |\lambda_i|$ guards for C_i, one for each variable;

3. The location of the guards must must be consistent for all variables except y: There is a guard at literal a in C_i if and only if there is a guard at literal a in C_{i+1};

4. There is a guard in C_i at one of $\overline{x}$, y, or $\overline{z}$.

Similar actions are done to build the chunks $C_{-1}, C_{-2}, ..., C_{-k'}$ for the α sequence. Our discussion implies that chunks $C_{-k'}, ..., C_0, ..., C_k$ can be guarded with

$\sum_{-k'}^{k} n_i$ guards if and only if we have a satisfying assignment to the planar 3SAT formula Φ. The location of the guards in chunk C_0 will tell us the truth value for each variable. Our construction will be such that if Φ is satisfiable, then $\sum_{-k'}^{k} n_i$ guards are *sufficient* for seeing all distinguished points. This will establish NP-hardness.

2.3 Reduction: Gadgets

The following subsections describe the gadgets introduced in Section 2.2. In Section 2.3.1, we begin by describing the shape of a chunk and the location of the literals corresponding to a variable. In Section 2.3.2, we describe the basic gadget relating two chunks called the *mirror gadget*. Subsequently, we modify the variable gadget to obtain the deletion gadget, the inversion gadget, and the clause gadget. We will refer to the construction of chunks $C_1, C_2, \ldots, C_k$ as "going down" from C_0, and the construction of chunks $C_{-1}, C_{-2}, \ldots, C_{-k'}$ as "going up" from C_0. Take an arbitrary variable x in an even chunk C_i. Guard locations in C_i to the right of x will be considered "below" x and guard locations in C_i placed to the left of x will be considered "above" x. With odd chunks, guard locations to the left of x are considered below and guard locations to the right are considered above. For example, in Figure 2.2, w is above x; y is below x.

2.3.1 Variable Gadget

The first gadget we will describe is the *variable gadget*. An example of a variable gadget for x in chunk C_i is shown in Figure 2.3. The variable gadget has a *variable distinguished point*, d, *that can be seen from only two vertices: the literals* x

and $\overline{x}$ vertices. The following is what we will refer to as the *Uniqueness Claim*:

Uniqueness Claim. *No guard can see more than 1 variable distinguished point.*

Because of the *Uniqueness Claim*, the total number of variable distinguished points provides a lower bound on the number of guards that are necessary to guard all of the distinguished points.

To see how multiple variables are placed, assume a chunk C_i has $\lambda_i = \langle w, x, y, z \rangle$. Figure 2.2 shows how variable gadgets corresponding to each variable are placed within the chunk. Chunk C_0 has n such variable gadgets, 1 for each variable. In Figure 2.2, 4 guards are required to guard the 4 variable distinguished points because of the *Uniqueness Claim*.

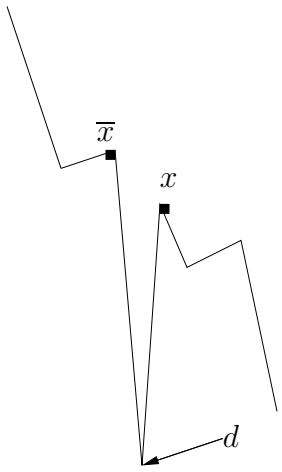


Figure 2.3: Variable Gadget.

Local Summary of Variable Gadgets: To guard the variable distinguished point d for a variable x in chunk C_i, at least 1 guard must be placed at the literal x or $\overline{x}$ guard location in C_i.

2.3.2 Mirroring

Going down, chunks C_i and C_{i+1} form what we call a mirror gadget. Here, we will have $n_i = |\lambda_i|$, $\lambda_{i+1} = \lambda_i$, and $n_{i+1} = |\lambda_{i+1}|$. The relationship between C_i and C_{i+1} will be what we call a *mirroring*, which has the following property: to guard C_i and C_{i+1} using $n_i + n_{i+1}$ guards, it is necessary that we have:

1. exactly $|\lambda_{i+1}|$ guards at the literals within C_{i+1}, one for each variable so that this corresponds to an assignment to the variables in λ_{i+1};

2. exactly $|\lambda_i|$ guards at the literals within C_i, one for each variable so that this corresponds to an assignment to the variables in λ_i;

3. The location of the guards must be consistent for all variables: There is a guard at literal y in C_i if and only if there is a guard at literal y in C_{i+1}.

To describe the mirroring for 1 variable, let us first focus on a variable gadget corresponding to a variable b in chunks C_i and C_{i+1}, see Figure 2.4. We introduce the notion of *mirrored distinguished points* corresponding to b in C_{i+1}. In Figure 2.4, mirrored distinguished points are p and q. b^i is the literal b in chunk C_i. $\overline{b^{i+1}}$ and b^i both see our mirrored distinguished point p but neither see q. b^{i+1} and $\overline{b^i}$ both see q but neither see p. A ray shot from $\overline{b^i}$ through b^i hits the terrain in C_{i+1} above p. This leads us to the following lemma:

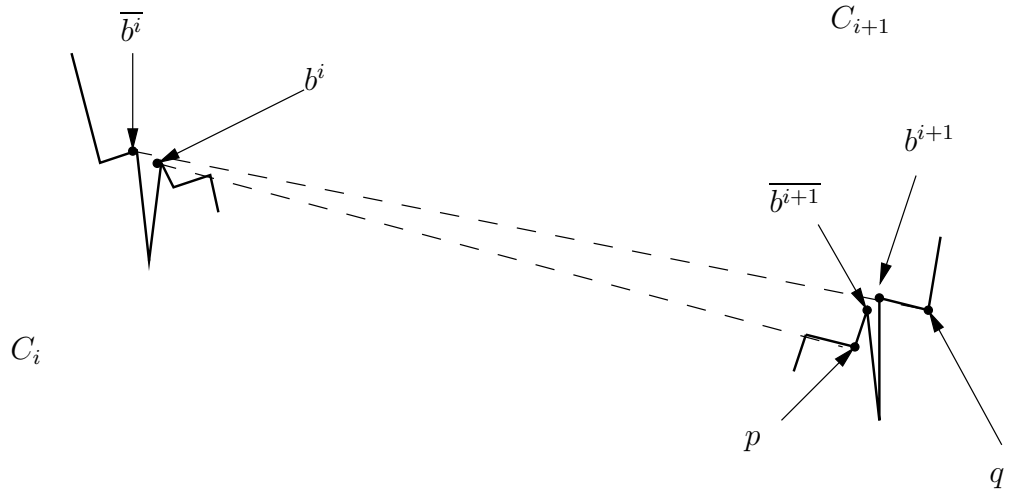


Figure 2.4: Mirroring one variable. Visibilities are as follows: $\overline{b^i}$ sees $\{q, b^{i+1}\}$. b^i sees $\{p, \overline{b^{i+1}}, b^{i+1}\}$. b^{i+1} sees $\{q, \overline{b^{i+1}}, \overline{b^i}, b^i\}$. $\overline{b^{i+1}}$ sees $\{p, b^{i+1}, b^i\}$.

Lemma 2. *For two guards to see the variable distinguished points in C_i and C_{i+1} corresponding to a variable b and the mirrored distinguished points corresponding to variable b in C_{i+1}, it is necessary and sufficient to place guards at the literal b locations in both chunks or guards at the literal $\overline{b}$ locations in both chunks.*

Proof. Since we have two variable gadgets for b, the *Uniqueness Claim* states that two guards are necessary to guard the variable distinguished points for b in C_i and C_{i+1}. We claim two guards are sufficient to guard the mirrored and variable distinguished points in C_{i+1} and variable distinguished points in C_i. We must choose one guard from $\{b^i, \overline{b^i}\}$ and one guard from $\{b^{i+1}, \overline{b^{i+1}}\}$. If we place a guard at b^i, q is not seen. Since $\overline{b^{i+1}}$ does not see q, we must place a guard at b^{i+1}. Similar arguments can be made if we choose $\overline{b^i}$ first. □

Mirroring up uses a similar proof. If a guard is placed at b^{i+1}, a guard must be placed at b^i so that p is seen. Similarly with $\overline{b^{i+1}}$ and $\overline{b^i}$.

We see in Figure 2.5 how variable gadgets are constructed to ensure a guard placed in one variable gadget does not see the mirrored distinguished points of a different variable gadget. Let a^i and b^i belong to chunk C_i. To ensure that guards placed at a literal guard location for one variable does not affect the mirroring of another variable, in other words q should be seen by only $\overline{a^{i+1}}$ and a^i and by no other guards in C_i and C_{i+1}, similarly q' should be seen by only a^{i+1} and $\overline{a^i}$ and by no other guards in C_i and C_{i+1}. The following are also true:

1. The line defined by a^i and m hits the terrain above point p'. m blocks a^i from seeing distinguished points below the variable gadget a in C_i and below the variable gadget for a in C_{i+1}. Recall that a ray shot from $\overline{a^i}$ through a^i hits the terrain above q so $\overline{a^i}$ does not see any distinguished points below the variable gadget for a in C_{i+1}. It is also easy to see that a^i and a^{i+1} do not see any distinguished points below a in C_i. In general, a guard placed at a literal for a variable h in some chunk C_i will not see any of the mirrored or variable distinguished points of different variables below (to the right of) the variable gadget for h in C_i or below (to the left of) the variable gadget for h in C_{i+1}.

2. Neither b^i nor $\overline{b^i}$ can see q or q'. This ensures that a guard placed at a literal location for the variable b in C_i does not see any of a's mirrored distinguished points in C_{i+1}. The ray shot from $\overline{b^i}$ through d hit the terrain above (to the right of) q. Similarly for a ray shot from b^i through d. A ray shot from $\overline{b^i}$ through

a^{i+1} hits the terrain above (to the right of) q'. Similarly for b^i. In general, no guard below (to the right of) the variable gadget for $a \in C_i$ can see q or q' nor can any guard below (to the left of) the variable gadget for $a \in C_{i+1}$ see q or q'. Note that the visibilities do not disrupt the order claim.

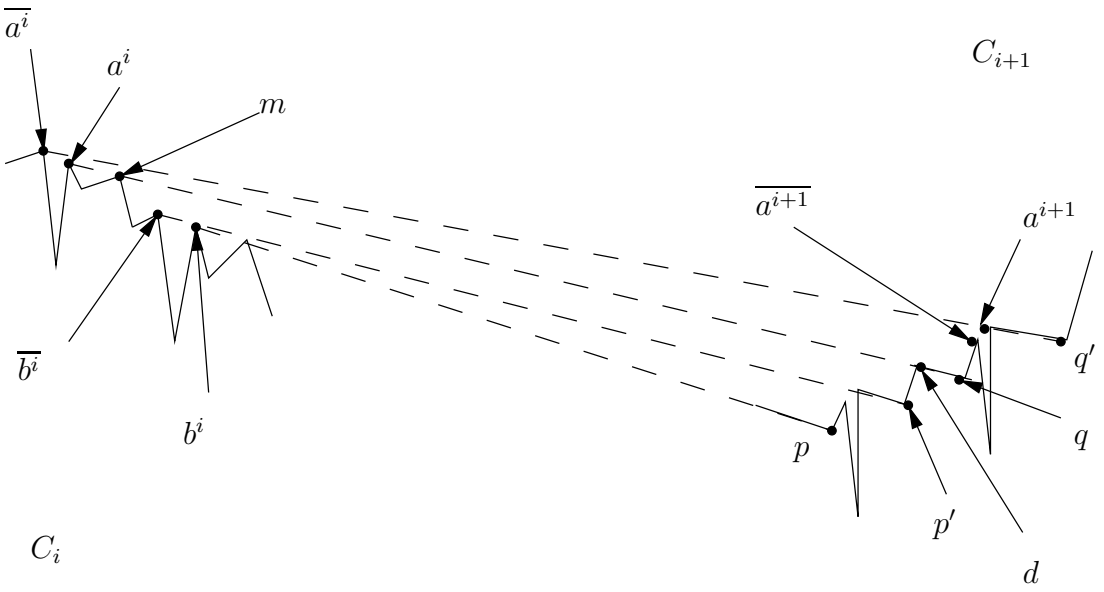


Figure 2.5: Variable gadgets do not interfere with each other. Important visibilities are as follows: $\overline{a^i}$ sees $\{q', a^{i+1}\}$. a^i sees $\{q, a^{i+1}, \overline{a^{i+1}}\}$. $\overline{b^i}$ sees $\{p', \overline{a^{i+1}}, a^{i+1}\}$. b^i sees $\{p, a^{i+1}, \overline{a^{i+1}}\}$. Note that the visibilities do not disrupt the order claim.

Local Summary of Mirroring Gadget C_i–C_{i+1} going down: To guard the variable distinguished points and mirrored distinguished points of C_{i+1} and the variable distinguished points of C_i with $n_i + n_{i+1}$ guards, it is necessary and sufficient to place n_i guards at literals in C_i and n_{i+1} guards at literals in C_{i+1} in a consistent way.

For mirroring up, the Figure is exactly the same as Figure 2.5. The visibilities

as described before do not change. Lemma 2 says that if we have a guard at a^{i+1}, the second guard is forced to be at a^i. Similarly for $\overline{a^{i+1}}$ and $\overline{a^i}$.

Local Summary of Mirroring Gadget C_{i+1}–C_i going up: To guard the variable distinguished points and mirrored distinguished points of C_{i+1} and the variable distinguished points of C_i with $n_i + n_{i+1}$ guards, it is necessary and sufficient to place n_i guards at literals in C_i and n_{i+1} guards at literals in C_{i+1} in a consistent way.

Subsequent subsections will describe how we can modify a variable gadget to place different gadgets, ie deletion, clause and inversion.

2.3.3 Deletion Gadget

A deletion of a variable x going down from chunk C_i to chunk C_{i+1} involves flattening out the terrain in chunk C_{i+1} where the variable gadget for x would have been placed.

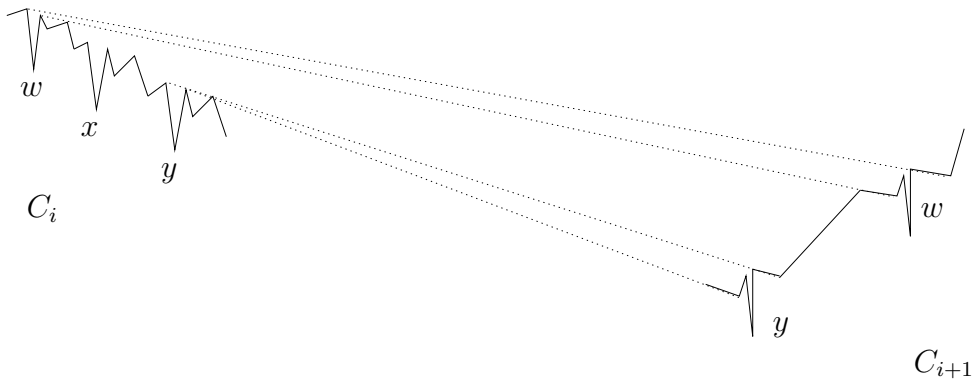


Figure 2.6: Deleting a variable when mirroring down.

Let us consider chunks C_i and C_{i+1} going down when a deletion gadget is being placed to delete variable x. The total number of guards needed will be $n_i + n_{i+1}$ where $n_i = |\lambda_i|$ and $n_{i+1} = |\lambda_{i+1}|$. The list of variables in C_{i+1}, $\lambda_{i+1} = \lambda_i \setminus x$. We replace the variable gadget for x in C_{i+1} with a flat surface as seen in Figure 2.6.

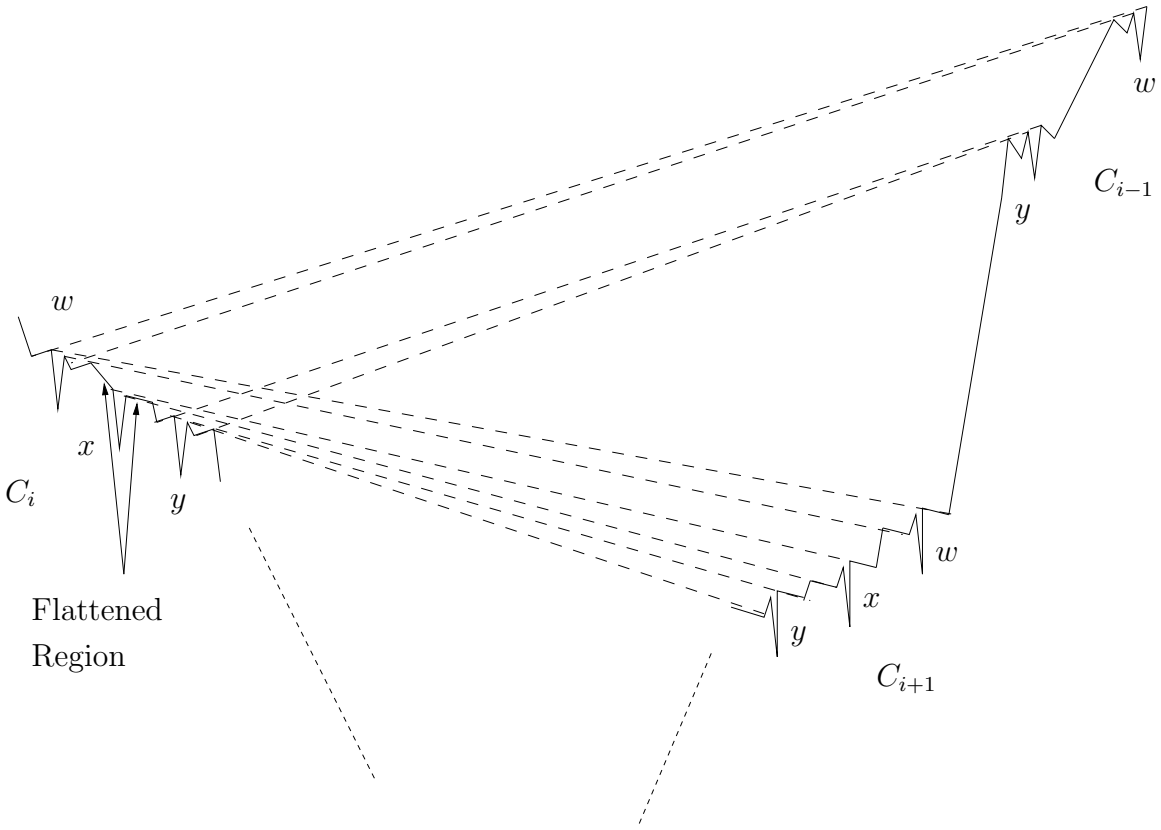


Figure 2.7: Deleting a variable when mirroring up. A more detailed view of variable x in chunk C_i is shown in Figure 2.8

Local Summary of Deletion Gadget C_i–C_{i+1} going down: To guard the variable distinguished points and mirrored distinguished points of C_{i+1} and the variable

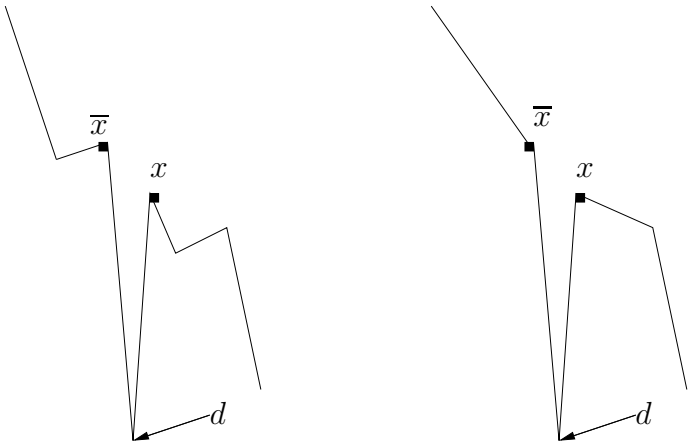


Figure 2.8: This Figure shows the region that is flattened out for a variable x when deleting up the terrain. The left picture is what the x variable gadget originally looked like. The right picture is what the variable gadget for x looks like after the mirrored distinguished points are removed/flattened.

distinguished points of C_i with $n_i + n_{i+1}$ guards, it is necessary and sufficient to place n_i guards at literals in C_i and n_{i+1} guards at literals in C_{i+1} in a consistent way.

Going up, we need three chunks C_{i+1}, C_i, and C_{i-1} to construct a deletion gadget for deleting variable x. We will have $\lambda_i = \lambda_{i+1}$, $\lambda_{i-1} = \lambda_{i+1} \setminus x$, $n_{i+1} = |\lambda_{i+1}|, n_i = |\lambda_i|$ and $n_{i-1} = |\lambda_{i-1}|$. The total number of guards needed will be $n_{i+1} + n_i + n_{i-1}$. We flatten out the mirrored distinguished points of variable gadget x in C_i as seen in Figure 2.7 and Figure 2.8. The mirrored distinguished points for x in C_i were originally there to help us mirror x from C_i to C_{i+1}. However, x is being deleted so the mirrored distinguished points can go away as shown.

Local Summary of Deletion Gadget C_{i+1}–C_i–C_{i-1} going up: To guard the variable distinguished points and mirrored distinguished points of C_{i+1} and C_i, and the variable distinguished points of C_{i-1} with $n_{i+1} + n_i + n_{i-1}$ guards, it is necessary

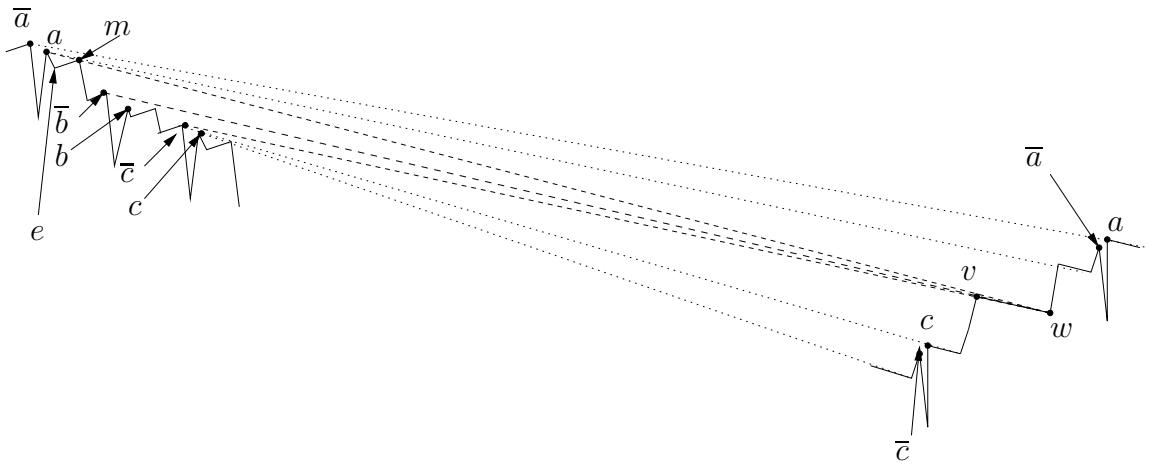


Figure 2.9: Clause going down, a detailed view of how the b variable gadget in C_i is modified is shown in Figure 2.10. A detailed view of how the a variable gadget in C_i is modified is shown in Figure 2.11.

and sufficient to place n_{i+1} guards at literals in C_{i+1} and n_i guards at literals in C_i, and n_{i-1} guards at literals in C_{i-1} in a consistent way.

2.3.4 Downward Clause Gadget

Let us say the clause we are constructing is $Cl_i = (a \vee \overline{b} \vee \overline{c})$, see Figure 2.9. We will have $\lambda_{i+1} = \lambda_i \setminus b$, $n_i = |\lambda_i|, n_{i+1} = |\lambda_{i+1}|$. The total number of guards needed within C_i and C_{i+1} will be $n_i + n_{i+1}$. We will replace the middle variable gadget b in C_{i+1} with our clause gadget. In chunk C_i, the left to right ordering of literals if i is even (right to left if i is odd) must be exactly $a, \overline{b}, b, \overline{c}$. We will assume the ordering is correct when placing a clause gadget[2]. In Figure 2.9, w is our *clause distinguished point*. We can manipulate the b variable gadget in C_i so that b is blocked from seeing w, see Figure 2.10. Two small changes are made to this variable gadget:

[2]Section 2.3.6 will show how to make a change if the ordering is incorrect.

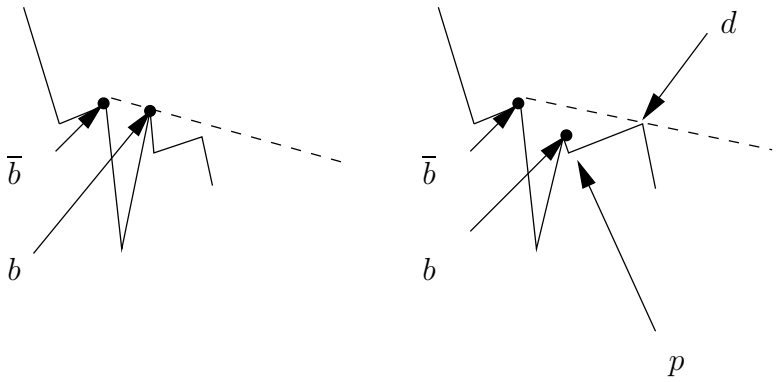


Figure 2.10: The left picture shows the original b in C_i. The right picture shows 2 small changes. The first being the vertical lowering of literal guard location b. The second is the extension of $\overline{pd}$ line segment.

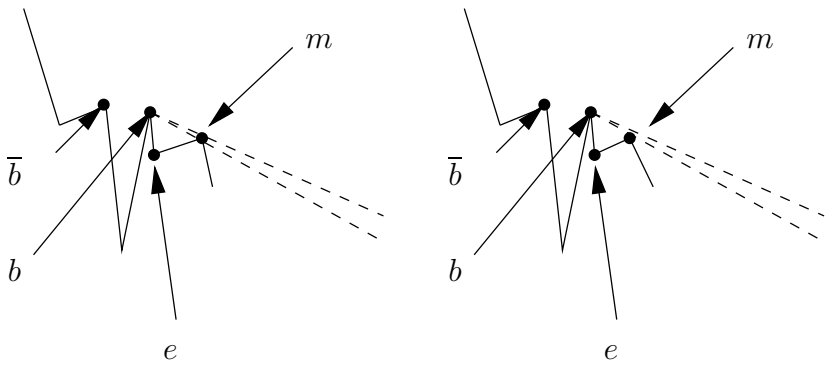


Figure 2.11: The left picture shows the original a in C_i. The right picture shows the small change. The top dotted line is the original visibility of the literal guard a. The bottom dotted line is the new visibility of the literal guard a. To create this new visibility, the m point is moved towards the e point.

1. The b literal is moved downward vertically so it does not see w. This does not affect previous downward mirrorings.

2. d is extended further on the $\overline{pd}$ line so that the visibility of $\overline{b}$ to the right is unchanged. The angle defined by bpd is unchanged. Initially, $\overline{b}$ was blocked from seeing mirrored distinguished points for c in C_{i+1} by b in C_i. Since b moved, the d point also needed to move so that $\overline{b}$ does not see into the c variable gadget in C_{i+1}.

Referring back to Figure 2.9, the original use of the m point was to block a potential guard placed at the $a \in C_i$ guard location from seeing mirrored distinguished points below (to the left of) the a variable gadget in C_{i+1}. In this case however, we want $a \in C_i$ to see w. We move our m point towards e so that the guard location for $a \in C_i$ sees w, see Figure 2.11. It should be noted that our mirroring of a is not disrupted with this modification. This modification allows $a \in C_i$ to see w. The visibilities do not disrupt the order claim. It is important to note that $\overline{a} \in C_i$ does not see w. Finally, we adjust the $\overline{vw}$ line segment by moving w slightly upwards so that w sees $\overline{c} \in C_i$. A ray shot from w through v will hit the terrain in chunk C_i at point $\overline{c}$ so that c in C_i does not see w. It should be noted that the mirroring down of a and c are still intact; we are still able to mirror the values of a and c down the terrain. If neither $a \in C_i$ nor $\overline{b} \in C_i$ nor $\overline{c} \in C_i$ is chosen as a guard location, we require an extra guard to see w. However if one of these literals is chosen to be a guard, our clause distinguished point w is guarded and no extra guard is needed.

We also note that b will no longer be used in any future clauses going downward. The reduction from planar 3SAT allows us to order the clauses in a certain way to ensure that the middle variable will no longer be used in future clauses going down the terrain. A detailed explanation on how the clauses are ordered, see Section 2.4. Because of this ordering, we can safely replace the b variable gadget in C_{i+1} with a clause gadget.

Local Summary of Clause Gadget C_i–C_{i+1} going down: To guard the variable distinguished points, clause distinguished point and mirrored distinguished points of C_{i+1} and the variable distinguished points of C_i with $n_i + n_{i+1}$ guards, it is necessary and sufficient to place n_i guards at literals in C_i and n_{i+1} guards at literals in C_{i+1} in a consistent way. Note that if a guard is placed at literal locations a or $\overline{b}$ or $\overline{c}$ in chunk C_i, our clause distinguished point is seen and no additional guard is required.

2.3.5 Upward Clause Gadget

The clause gadget going up is done similarly to mirroring variables upward with a few small changes. We will have $\lambda_{i-1} = \lambda_i, \lambda_{i-2} = \lambda_i \setminus b$, $n_i = |\lambda_i|, n_{i-1} = |\lambda_{i-1}|$ and $n_{i-2} = |\lambda_{i-2}|$. The total number of guards needed will be $n_i + n_{i-1} + n_{i-2}$. We replace the highest (in this case leftmost) mirroring distinguished point of $b \in C_{i-1}$ with a clause distinguished point w. We flatten out the other mirroring distinguished point for $b \in C_{i-1}$ similar to the deletion gadget, see Figure 2.12 for a broad view. See Figure 2.13 for a detailed view of the modification of variable gadget b in chunk C_{i-1}.

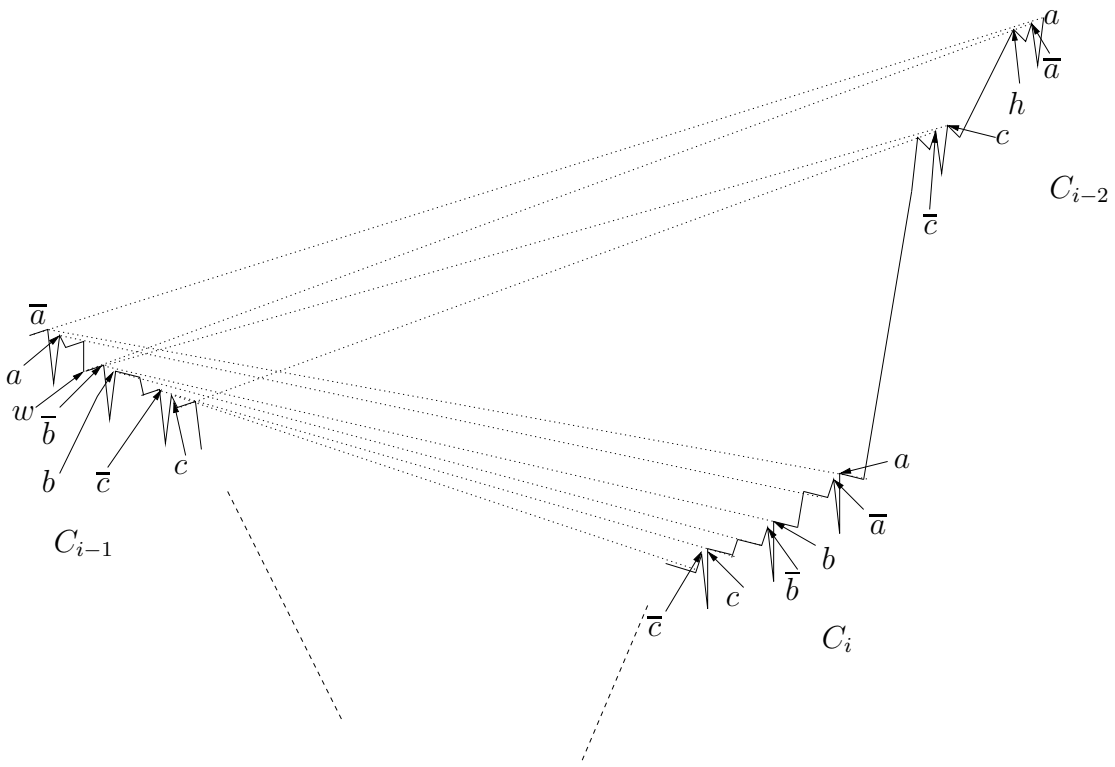


Figure 2.12: Clause going up.

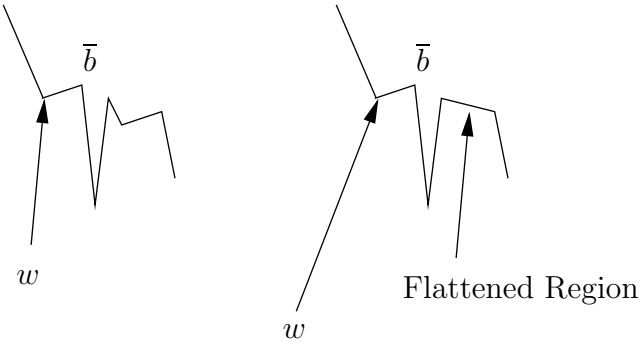


Figure 2.13: The left picture shows what the b variable gadget originally looked like in C_{i-1} in Figure 2.12. The right picture shows the modified variable gadget to contain the clause distinguished point.

We make the following small modifications to our terrain, see Figure 2.12:

1. h is adjusted so that $\overline{a}$ in C_{i-2} sees w. The original purpose of our h point was to ensure that $\overline{a} \in C_{i-2}$ did not see mirrored distinguished points below (to the right of) $a \in C_{i-1}$. However, we want $\overline{a} \in C_{i-2}$ to see w. It should be noted that $a \in C_{i-2}$ does not see w.

2. w is adjusted accordingly so that a ray shot from w through $\overline{b}$ in C_{i-1} sees the literal guard location for $c \in C_{i-2}$. This allows c in C_{i-2} to see w.

We now have only three literal guard locations that can see w: $\{\overline{b} \in C_{i-1}$, $\overline{a} \in C_{i-2}$, $c \in C_{i-2}\}$. We note that the b variable can safely disappear as it will not be needed in any other clauses going upwards because of the ordering of the clauses. Section 2.4 explains in further detail why this is the case.

Local Summary of Clause Gadget C_i–C_{i-1}–C_{i-2} going up: To guard the variable distinguished points, clause distinguished point and mirrored distinguished points of C_{i-1}, the variable distinguished points of C_{i-2} and the variable and mirrored distinguished points of C_i with $n_i + n_{i-1} + n_{i-2}$ guards, it is necessary and sufficient to place n_i guards at literals in C_i and n_{i-1} guards at literals in C_{i-1} and n_{i-2} guards at literals in C_{i-2} in a consistent way. Note that if a guard is placed at a in C_{i-2} or $\overline{b}$ in C_{i-1} or $\overline{c}$ in C_{i-2}, our clause distinguished point is seen and no additional guard is required.

2.3.6 Inversion Gadget

The left to right (right to left) ordering of literals becomes important when placing a clause gadget and it is possible that the literals are "out of order." In a normal mirroring of variable a, the left to right order of a and $\overline{a}$ will be the same in all even chunks, similarly with all odd chunks. To switch the order of the literals, we make use of an inversion gadget.

Let us consider chunks C_i, C_{i-1}, C_{i-2} when an inversion gadget is being placed to invert a variable, see Figure 2.14. We will have $\lambda_{i-1} = \lambda_i$, $\lambda_{i-2} = \lambda_i$, $n_i = |\lambda_i|, n_{i-1} = |\lambda_{i-1}| + 1$ and $n_{i-2} = |\lambda_{i-2}|$. The total number of guards needed will be $n_i + n_{i-1} + n_{i-2}$. The a literal in C_{i-2} is to the right of $\overline{a}$ in C_{i-2}. Using the inversion gadget in C_{i-1}, we can swap the left to right ordering of the a and $\overline{a}$ literal so that a in C_i is to the left of $\overline{a}$ in C_i.

2.3.6.1 Inverting Down

In Figure 2.14, the variable gadget for a in chunk C_{i-1} is replaced with an inversion gadget. The inversion gadget adds two literal locations for variable a in C_{i-1}, namely a' and $\overline{a'}$. The variable and mirrored distinguished points of $a \in C_{i-1}$ are being replaced with five *inversion distinguished points*. These five *inversion distinguished points* are: $\{x, z_1, y_1, y_2, z_2\}$. y_1 and y_2 should be thought of as "mirrored distinguished points" since they are seen by guards inside chunk C_{i-1} and by the a and $\overline{a}$ literal guard locations in chunk C_{i-2}. z_1 and z_2 are the "variable distinguished points." They are variable distinguished points in the sense that no guard outside of the inversion gadget

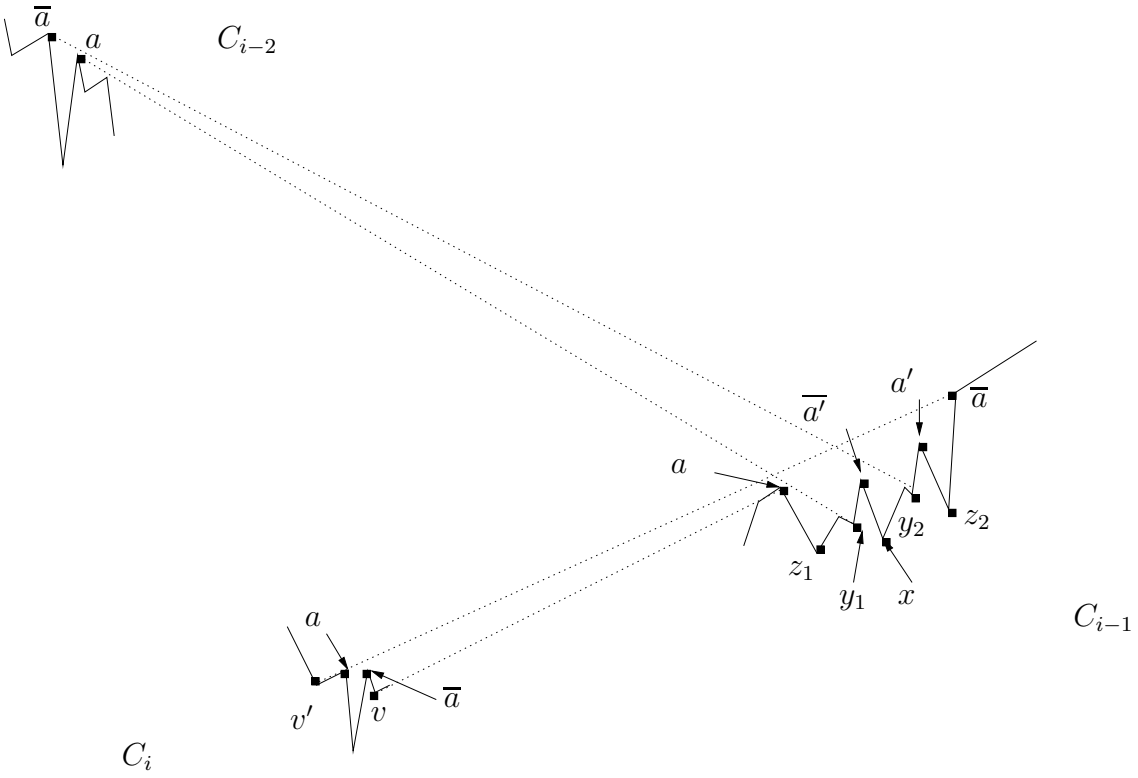


Figure 2.14: Inverting one variable.

for a in C_{i-1} can see them. More importantly, z_1 and z_2 are considered replacement "variable distinguished points" because they obey the *Uniqueness Claim*. z_1 is only seen by $\{a \in C_{i-1}, \overline{a'} \in C_{i-1}\}$. z_2 is only seen by $\{\overline{a} \in C_{i-1}, a' \in C_{i-1}\}$. The x point is the special inversion distinguished point that allows the inversion to take place. The important visibilities of Figure 2.14 are given here:

1. $a \in C_{i-1}$ sees $\{z_1, v\}$.

2. $\overline{a'} \in C_{i-1}$ sees $\{z_1, y_1, x\}$.

3. $a' \in C_{i-1}$ sees $\{x, y_2, z_2\}$.

4. $\overline{a} \in C_{i-1}$ sees $\{z_2, v'\}$.

5. $a \in C_{i-2}$ sees $\{y_1\}$.

6. $\overline{a} \in C_{i-2}$ sees $\{y_2\}$.

Although not entirely obvious from the Figure, it's important to note a ray shot from $\overline{a} \in C_{i-1}$ through $a \in C_{i-1}$ hits the terrain to the left of $v \in C_i$. These visibilities do not disrupt the order claim.

Because of the *Uniqueness Claim*, it is necessary that we place 4 guards to see the variable distinguished points of C_i, C_{i-1} and C_{i-2}. If we place the 4 guards in a consistent manner, the mirrored distinguished points of C_i and the inversion distinguished points of C_{i-1} will also be seen. The *Uniqueness Claim* states that we must choose a guard from each of the following sets:

1. $\{a \in C_i, \overline{a} \in C_i\}$ so that we see the variable distinguished point for a in C_i.

2. $\{a \in C_{i-2}, \overline{a} \in C_{i-2}\}$ so that we see the variable distinguished point for a in C_{i-2}.

3. $\{a \in C_{i-1}, \overline{a'} \in C_{i-1}\}$ so that we see z_1.

4. $\{a' \in C_{i-1}, \overline{a} \in C_{i-1}\}$ so that we see z_2.

Consistent placement of the guards ensures that the following distinguished points are also seen: $\{v', v, y_1, x, y_2\}$. The only 2 solutions are $\{a \in C_i, a \in C_{i-2}, a \in C_{i-1}, a' \in C_{i-1}\}$ and $\{\overline{a} \in C_i, \overline{a} \in C_{i-2}, \overline{a} \in C_{i-1}, \overline{a'} \in C_{i-1}\}$. Any other combination of guards means at least 1 distinguished point is unseen. A detailed explanation of this is explained in the next few paragraphs.

We will only concern ourselves with the inversion of a and ignore the other variables being mirrored. The other variables are being mirrored without consequence. We will assume we already have a guard at $a \in C_{i-2}$ or $\overline{a} \in C_{i-2}$. Because of the *Uniqueness Claim*, it is necessary that we place 3 guards to see the remaining "variable distinguished points" of z_1, z_2 and also the variable distinguished point of $a \in C_i$. If we place the remaining 3 guards in a consistent manner, the remaining distinguished points of y_1, y_2, x, v and v' will also be seen.

Using the example in Figure 2.14, let us say $a \in C_{i-2}$ was chosen to be a guard. We know at least 1 guard must be placed at $a \in C_i$ or $\overline{a} \in C_i$ to see the variable distinguished point of a in C_i leaving 2 guards to see the unguarded inversion distinguished points: y_2, x, z_1 and z_2. Let us first consider who can guard y_2. The only 2 guards that see y_2 are $a' \in C_{i-1}$ and $\overline{a} \in C_{i-2}$. If we place a guard at $\overline{a} \in C_{i-2}$, one of the "variable distinguished points" of z_1 or z_2 will go unseen. Therefore we must choose to place our guard at $a' \in C_{i-1}$.

We have one guard left to place in the inversion gadget that must see z_1. In order to see both v and v', we must place our guard at $a \in C_{i-1}$. The only other choice is $\overline{a'} \in C_{i-1}$ but this guard does not see v or v'. Placing a guard at $a \in C_{i-1}$ leaves v' and the variable distinguished point of $a \in C_i$ unguarded. $a \in C_i$ is chosen to be a guard and the inversion is complete. Similar arguments are made showing that if $\overline{a} \in C_{i-2}$ is chosen, then $\overline{a'} \in C_{i-1}$, $\overline{a} \in C_{i-1}$ and $\overline{a} \in C_{i-1}$ must be chosen.

Local Summary of Inversion Gadget C_{i-2}–C_{i-1}–C_i going down: To guard the variable distinguished points and mirrored distinguished points of C_i, the variable, mirrored and inversion distinguished points of C_{i-1} and the variable distinguished points of C_{i-2} with $n_i + n_{i-1} + n_{i-2}$ guards, it is necessary and sufficient to place n_i guards at literals in C_i, n_{i-1} guards in C_{i-1}, and n_{i-2} guards at literals in C_{i-2} in a consistent way. If variable a is being inverted, the left to right ordering of the literals a and $\overline{a}$ in C_{i-2} are opposite of that in C_i.

2.3.6.2 Inverting Up

Similar arguments are used to show the inversion going up. The same visibilities hold and the same set of distinguished points must be seen. We will have $\lambda_{i-1} = \lambda_{i-2}$, $\lambda_i = \lambda_{i-2}$, $n_i = |\lambda_i|, n_{i-1} = |\lambda_{i-1}| + 1$ and $n_{i-2} = |\lambda_{i-2}|$. The total number of guards needed will be $n_i + n_{i-1} + n_{i-2}$.

Local Summary of Inversion Gadget C_i–C_{i-1}–C_{i-2} going down: To guard the variable distinguished points and mirrored distinguished points of C_i, the variable, mirrored and inversion distinguished points of C_{i-1} and the variable distinguished points of C_{i-2} with $n_i + n_{i-1} + n_{i-2}$ guards, it is necessary and sufficient to place n_i guards at literals in C_i, n_{i-1} guards in C_{i-1}, and n_{i-2} guards at literals in C_{i-2} in a consistent way. If variable a is being inverted, the left to right ordering of the literals a and $\overline{a}$ in C_i are opposite of that in C_{i-2}.

2.3.7 Local vs Global View of Gadgets

Having completed the construction, we see that for every chunk C_i we need n_i points placed within the chunk just to guard the variable distinguished points within the chunk. This is given by the *Uniqueness Claim*. We now observe that the local summary of any gadget holds good in a global sense, that is, it is independent of how guards are placed in chunks outside this gadget. We illustrate this by summarizing a mirror gadget using chunks C_i and C_{i+1} going down. The reader may find it useful to compare with the local summary in Section 2.3.2.

Global Summary of Mirroring Gadget C_i–C_{i+1} going down: To guard the variable distinguished points and mirrored distinguished points of C_{i+1} and the variable distinguished points of C_i with n_i guards in C_i and n_{i+1} guards in C_{i+1}, it is necessary to place n_i guards at literals in C_i and n_{i+1} guards at literals in C_{i+1} in a consistent way. This necessity holds good for *any* placement of guards in locations outside C_i and C_{i+1}. The local sufficiency condition obviously holds good for any placement of guards outside C_i and C_{i+1}.

What this stronger condition means, in the context of Figure 2.5, is that $\overline{a^{i+1}}$ and a^i are the only guard locations that see q among *all possible* guard locations on the terrain. Similarly for points q', p and p'. The argument for why this holds is the same as the one made for guard locations within C_i and C_{i+1}. Having completed the entire construction, we are only now in a position to state this global property. The "necessary" parts of each of the gadgets are similarly modified to hold in a global

sense.

2.3.8 Putting it all Together

Each chunk C_i in our construction needs n_i guards within it. Because of the *Uniqueness Claim*, the terrain we construct needs at least $\sum_{-k'}^{k} n_i$ guards just to see all of the variable distinguished points[3]. Our construction ensures that if the distinguished points can be seen by $\sum_{-k'}^{k} n_i$ guards, then the input formula must be satisfiable. In particular, the assignment for the variables chosen by the n_0 points in chunk C_0 must be consistently mirrored to all chunks and the clause distinguished points must be seen. If the input formula is satisfiable, picking a satisfying assignment and propagating it through our gadgets in the natural way results in a set of $\sum_{-k'}^{k} n_i$ guards that see all of the distinguished points. Thus the proof of NP-hardness is thus completed.

Theorem 3. *Discrete terrain guarding is NP-hard.*

2.3.9 Continuous Version

Using the same construction, it can be shown that the continuous version of the terrain guarding problem is also NP-hard. We argue that the entire terrain can be seen by $\sum_{-k'}^{k} n_i$ guards if and only if the input formula is satisfiable. The *Uniqueness Claim* holds true despite guards being able to be placed anywhere on the terrain; since there are n_i variable distinguished points in chunk i, it follows that $\sum_{-k'}^{k} n_i$ are necessary for seeing the entire terrain. We now argue that if $\sum_{-k'}^{k} n_i$ see the entire

[3]Plus any possible inversion distinguished points.

terrain, they can be assumed to be in guard locations from the earlier reduction. From this, it follows that the formula is satisfiable.

Referring to Figure 2.3, the only potential guards that see d are points on a line segment $\overline{\overline{a}d}$ and points on a line segment $\overline{da}$. Let's say we pick a guard g on the line segment $\overline{\overline{a}d}$. $\overline{a}$ will see every point that g does. If we choose g as our guard, we can simply move our guard to $\overline{a}$ without any loss of visibility. Similar arguments can be said about a and the line segment $\overline{da}$. Therefore, we assume that any guard placed in the sub-terrain $[\overline{a}da]$ is either at a or $\overline{a}$. In particular, the only potential guards for d are a and $\overline{a}$. Similar arguments are made for the "variable distinguished points" in the inversion gadget. Therefore the *Uniqueness Claim* holds true in the continuous version; in other words the lower bound on the number of guards necessary to guard the variable distinguished points is the same in the continuous version as in the discrete version.

If the formula is satisfiable, $\sum_{-k'}^{k} n_i$ guards will see the entire terrain if the guards are placed in satisfying locations. Clearly the distinguished points are all seen. It can be shown that the terrain within the chunks is seen. The "empty space" outside of the chunks is also seen. For any 2 chunks C_i and C_{i-2} where $i = k, k-1, k-2, ..., 0, -1, ..., -k'+3, -k'+2$, any guard in chunk C_{i-1} will see the "empty space" between C_i and C_{i-2} because of the order claim. The "empty space" above chunk $C_{-k'+1}$ is seen by the guard placed at the literal for the last deleted variable in chunk $C_{k'}$. The "top" of the terrain is drawn in such a way that a guard placed for the last variable being deleted while "going up" will see the highest part

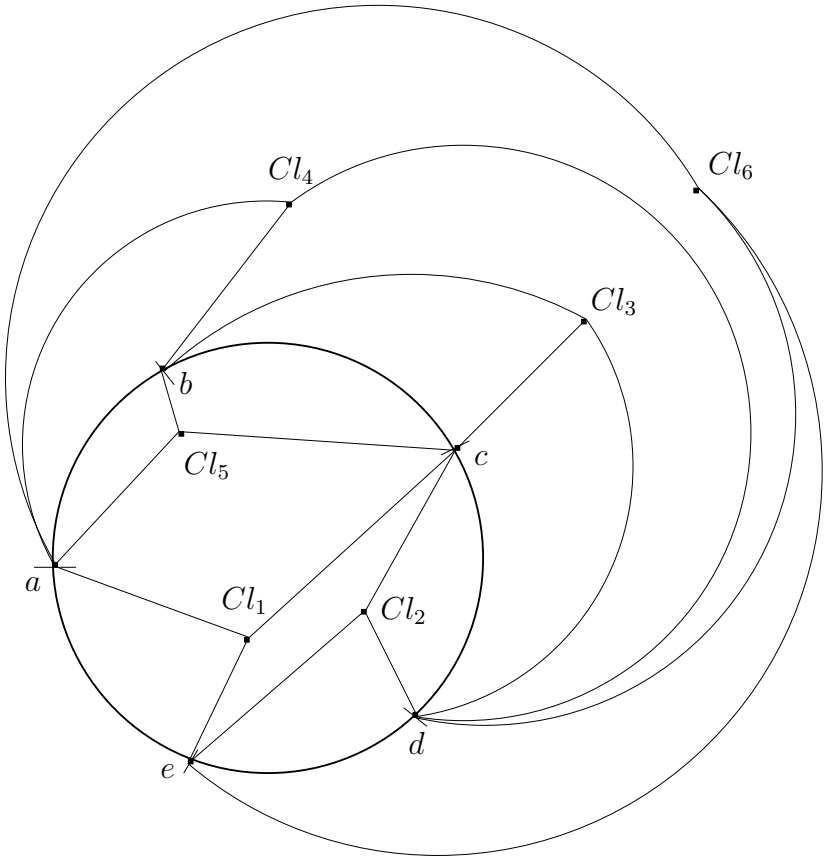


Figure 2.15: Planar 3SAT Example.

of the terrain in chunk $C_{k'}$. As for the "bottom" of the terrain, the terrain can be slightly modified between chunk C_{k-1} to C_k so that the terrain connecting those two chunks is seen by the only two remaining literals in chunk C_{k-1}. The entire terrain is thus seen.

2.4 Reduction Example

The reduction from planar 3SAT is done in the following way. The reduction given in this section combines several steps when visibility is not affected in an effort to minimize the number of figures needed. For example, several variables might be

deleted in one chunk where the reduction calls for only one deletion per chunk.

In Figure 2.15 we have an instance of planar 3SAT. There are clauses outside the variable cycle and clauses inside the variable cycle. A clause Cl_i is connected to 3 distinct variables. For example, $Cl_2 = (\overline{c} \vee \overline{d} \vee e)$. We arbitrarily pick a variable to be the lowest indexed variable and work clockwise around the variable cycle in increasing order. In the example in Figure 2.15 we choose a to be the lowest index variable. Our ordering of variables is $a < b < c < d < e$. This indexing of variables also gives us the ordering of variable gadgets in chunks. In even chunks, a will be the leftmost variable gadget, b will be the next leftmost, followed by c and so on. In odd chunks, a will be the rightmost variable gadget, b will be the next rightmost, followed by c and so on. Chunk C_0 is shown in Figure 2.18. The *interval* of a clause Cl_i is denoted $I(Cl_i)$. The interval of a clause is defined as the span from the lowest index variable in Cl_i to the highest indexed variable in Cl_i. From the example, $I(Cl_3) = (b, d)$.

We will focus on clauses outside of the variable cycle. We assume that each clause has 3 distinct variables. Because of this and because our graph is planar, every clause outside the variable cycle has a unique interval. Take two clauses outside the cycle Cl_i and Cl_j. We know the intervals are distinct. Because of planarity, either the intervals $I(Cl_i)$ and $I(Cl_j)$ have disjoint interiors or one of the intervals is properly contained in the other. If $I(Cl_i)$ is properly contained within $I(Cl_j)$, we say that clause $Cl_i < Cl_j$. We therefore have a partial ordering of the clauses. With this partial ordering we construct a valid total ordering of all of the clauses both inside the variable cycle and outside the variable cycle. The ordering of clauses outside the

cycle are used when placing clauses going up the terrain. Similarly, intervals $I(Cl_i)$ and $I(Cl_j)$ for clauses inside the variable ring either have disjoint interiors or $I(Cl_i)$ is properly contained within $I(Cl_j)$.

Let us consider the ordering of clauses outside of the cycle, call this ordering Γ. It is because of this ordering that we can delete the "middle" variable from the terrain when we place our clause gadget. The middle variable is defined as the variable that is not an endpoint of the interval. For example, if $Cl_i = (c \vee h \vee r)$ and $I(Cl_i) = (c, r)$, our middle variable is h. Let us take the first clause in Γ, call this clause Cl_i. We know that for every other clause $Cl_j \in \Gamma, Cl_j \not< Cl_i$. Since we are placing the smallest $I(Cl_i)$ first, based on our partial ordering, we know there are no clauses less than Cl_i. Since intervals do not overlap because of planarity, no other clause $Cl_j \in \Gamma$ will use the middle variable of Cl_i. Before we can place a clause gadget for Cl_i, there may be unused variables in the span of $I(Cl_i)$. We must first delete these unused variables before generating a clause gadget for Cl_i.

As an example, consider Figure 2.15. We have three clauses on the outside of the cycle and three clauses on the inside of the cycle. The following clauses are:

1. $Cl_1 = (a \vee \overline{c} \vee \overline{e})$
2. $Cl_2 = (\overline{c} \vee \overline{d} \vee e)$
3. $Cl_3 = (b \vee c \vee d)$
4. $Cl_4 = (a \vee b \vee d)$
5. $Cl_5 = (a \vee b \vee c)$

6. $Cl_6 = (\overline{a} \vee d \vee e)$

The intervals of each of the clauses are:

1. $I(Cl_1) = (a, e)$
2. $I(Cl_2) = (c, e)$
3. $I(Cl_3) = (b, d)$
4. $I(Cl_4) = (a, d)$
5. $I(Cl_5) = (a, c)$
6. $I(Cl_6) = (a, e)$

A partial ordering of the clauses outside the cycle is $Cl_3 < Cl_4$ and $Cl_4 < Cl_6$ and $Cl_3 < Cl_6$. A possible total ordering for clauses outside the ring is then $\langle Cl_3, Cl_4, Cl_6 \rangle$. A partial ordering of clauses inside the cycle is $Cl_2 < Cl_1$ and $Cl_5 < Cl_1$. A possible total ordering for clauses inside the ring is $\langle Cl_2, Cl_5, Cl_1 \rangle$ or $\langle Cl_5, Cl_2, Cl_1 \rangle$.

The remainder of the example will use the planar 3SAT example shown in Figure 2.16. The clauses for the example are defined as $Cl_1 = (a \vee \overline{c} \vee \overline{d}), Cl_2 = (\overline{a} \vee \overline{d} \vee e), Cl_3 = (b \vee c \vee d), Cl_4 = (a \vee b \vee d)$.

An overview of the entire terrain is shown in Figure 2.17. In each subsequent figure, we will show the specific part of the terrain we are describing along with a smaller version of the entire terrain to give reference to where we are on the terrain. Each figure places small rectangles for literals chosen as guard locations.

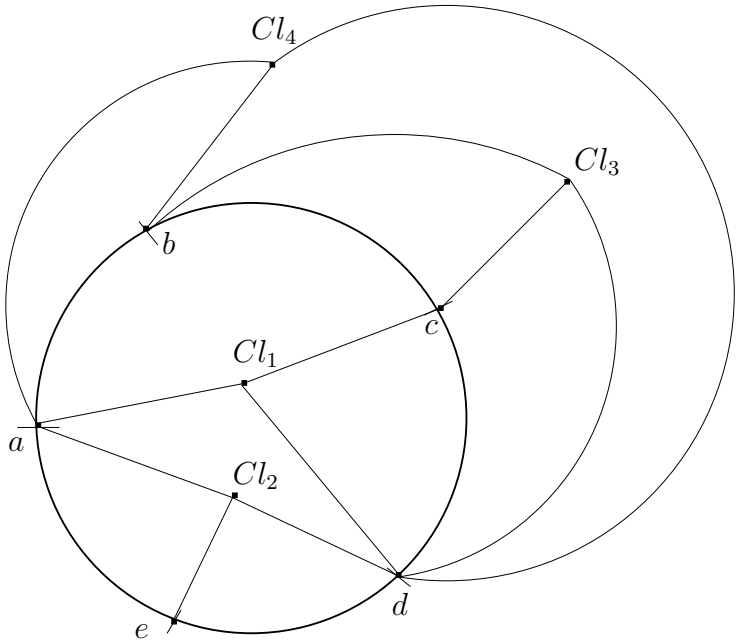


Figure 2.16: Planar 3SAT Example.

Figure 2.18 shows the details of chunk C_0. Starting at chunk C_0 we will work our way downward. In our example, we have chosen to place guards at the literal locations $a, \overline{b}, c, \overline{d}$, and e. We choose clause Cl_1 as the first clause placed on the terrain going downward from our total ordering obtained before.

Before we can place a gadget for clause Cl_1 on the terrain, we must delete the b variable since $I(Cl_1) = (a, d)$ and we only use the a, c and d variables. This deletion is done in chunk C_1 as shown in Figure 2.19. In this figure, the location where the b variable gadget would be in C_1 is replaced by a flat surface.

We are now ready to place a clause gadget and this is shown in Figure 2.20. In this figure, the c variable gadget, as seen in the left in Figure 2.20, is replaced with a clause gadget for clause Cl_1, as seen in the right in Figure 2.20. If we were

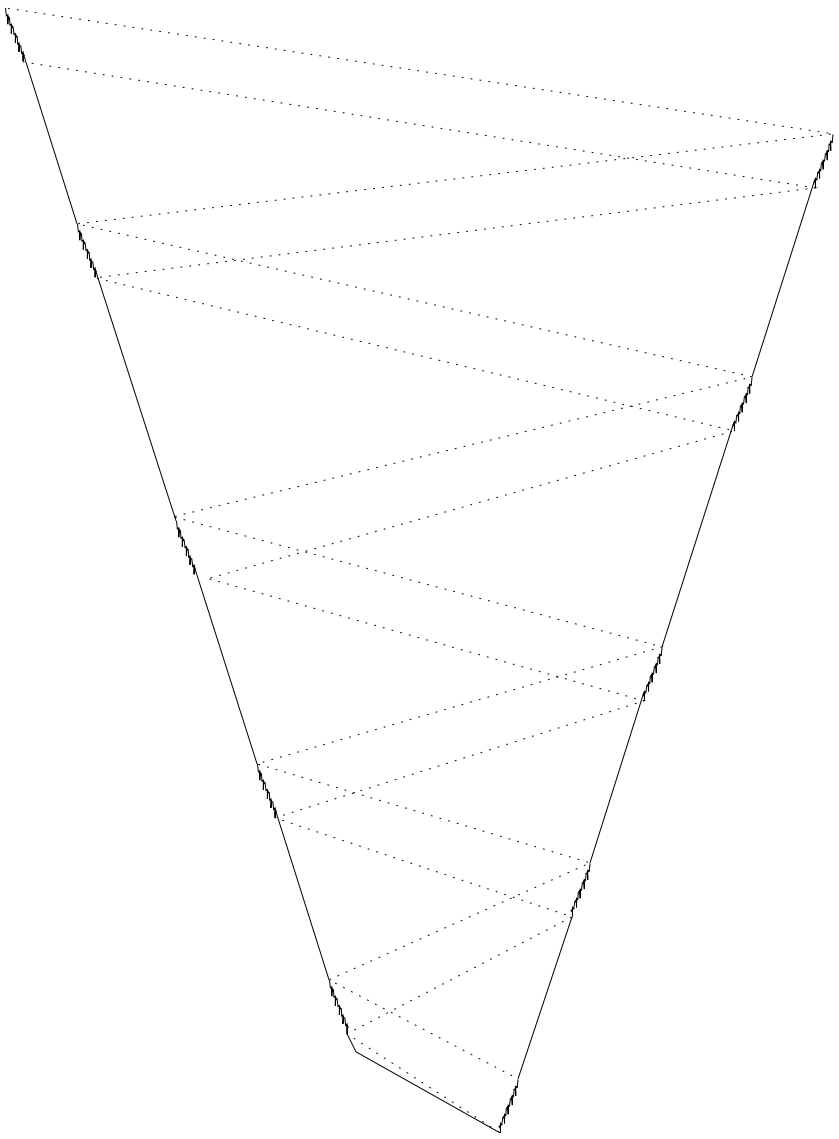


Figure 2.17: The entire terrain.

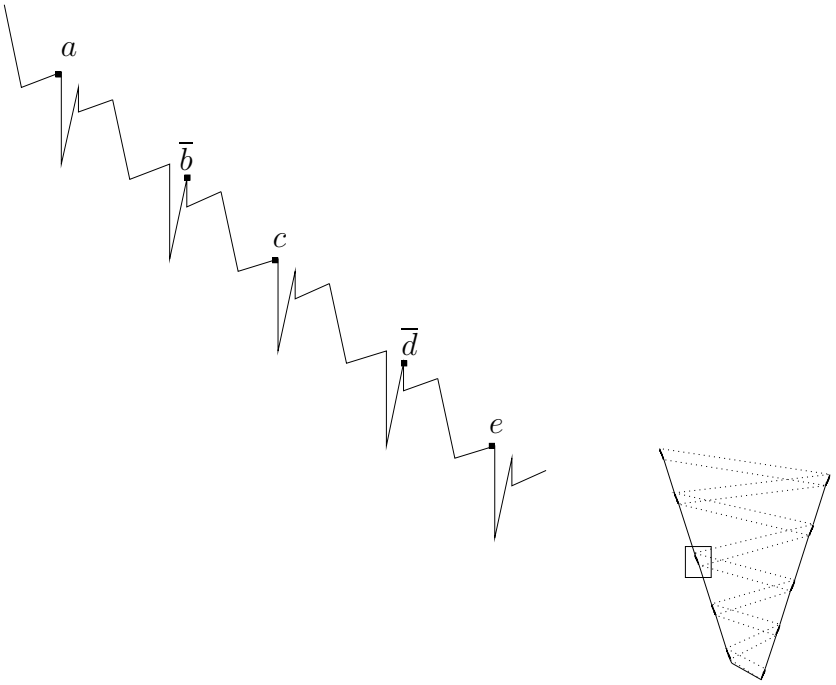


Figure 2.18: Chunk C_0 which contains variable gadgets for all variables on the variable cycle.

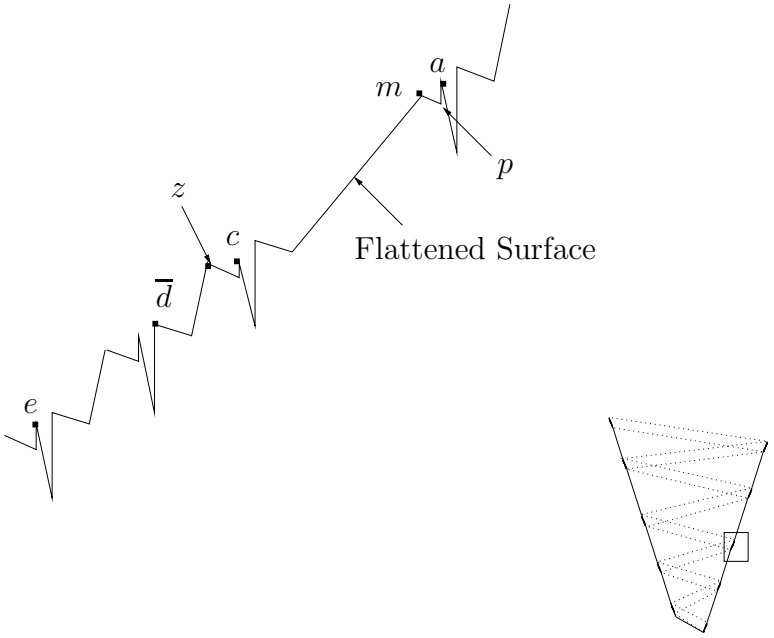


Figure 2.19: Chunk C_1 which places a deletion gadget for variable b.

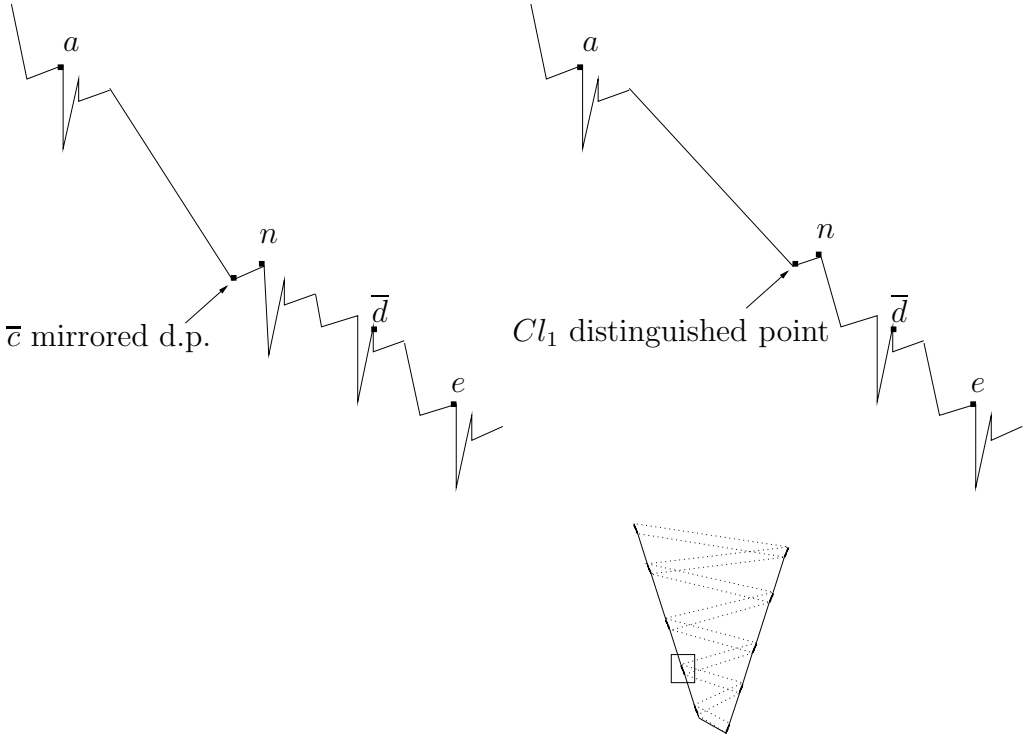


Figure 2.20: Chunk C_2 which places a clause gadget for clause Cl_1.

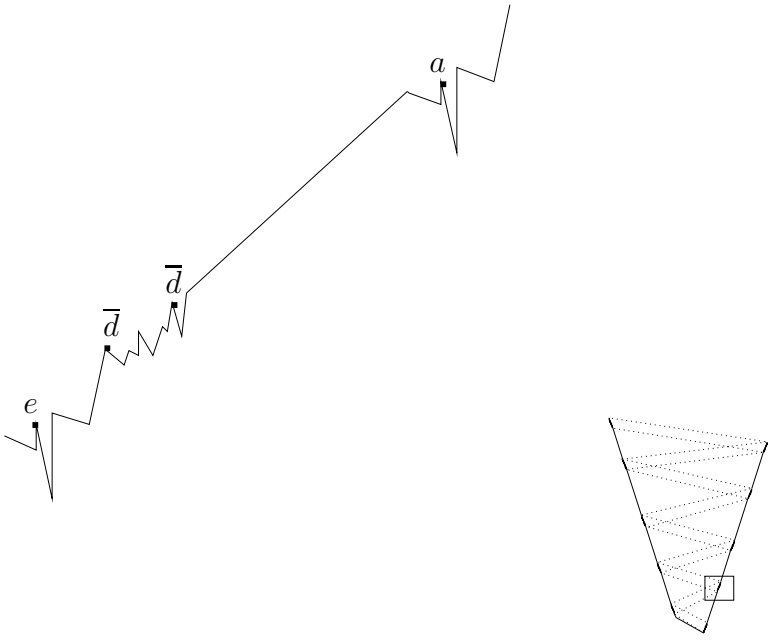


Figure 2.21: Chunk C_3 which places an inversion gadget for variable d.

only doing a mirroring, a ray shot from the $\overline{c}$ mirrored distinguished point in chunk C_2 through n would hit the $\overline{c}$ literal location in chunk C_1. However, since we are replacing the c variable gadget with a clause gadget, we want two other literals to see into c's variable region, which now contains the Cl_1 distinguished point. Those literal locations being $\overline{d}$ in C_1 and a in C_1. The Cl_1 distinguished point is adjusted upward accordingly so that a ray shot from Cl_1's distinguished point through n would hit the guard location for $\overline{d} \in C_1$. Referring back to Figure 2.19, we move the m towards p so that $a \in C_1$ sees Cl_1's distinguished point in chunk C_2. Recall that m was originally there to keep a from seeing into variable gadgets to the right of a in chunk C_2. We also move $c \in C_1$ down so that it does not see Cl_1's distinguished point. We adjust z so that $\overline{c}$ doesn't see variable gadgets below the c variable gadget in C_2. There are now 3 literal locations that can see Cl_1's distinguished point, namely $a \in C_1, \overline{c} \in C_1$, and $\overline{d} \in C_1$. If any of these literals have a guard at their location, Cl_1's distinguished point is seen, in other words, Cl_1 is satisfied. It is also important to note that none of these modifications affect the mirroring of variables a and d or any other variable.

The next clause that is placed is $Cl_2 = (\overline{a} \vee \overline{d} \vee e)$. Literals $\overline{a}$ and e are in the correct location in chunk C_2 but $\overline{d}$ is not. Therefore, the d variable must be inverted before we can place the next clause gadget. We invert d in chunk C_3 as shown in Figure 2.21. After the inversion gadget is placed in chunk C_3, the ordering of the d and $\overline{d}$ literals in chunk C_4 is correct. This is shown in Figure 2.22.

Once the literals are in the correct order, we can place our clause gadget for Cl_2. This is done in chunk C_5 as shown in Figure 2.23. In this Figure, the d variable

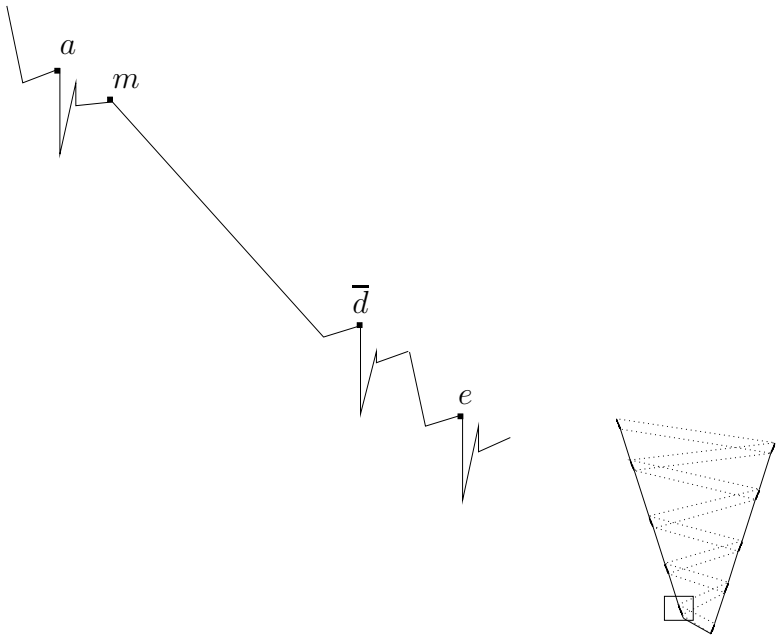


Figure 2.22: Chunk C_4.

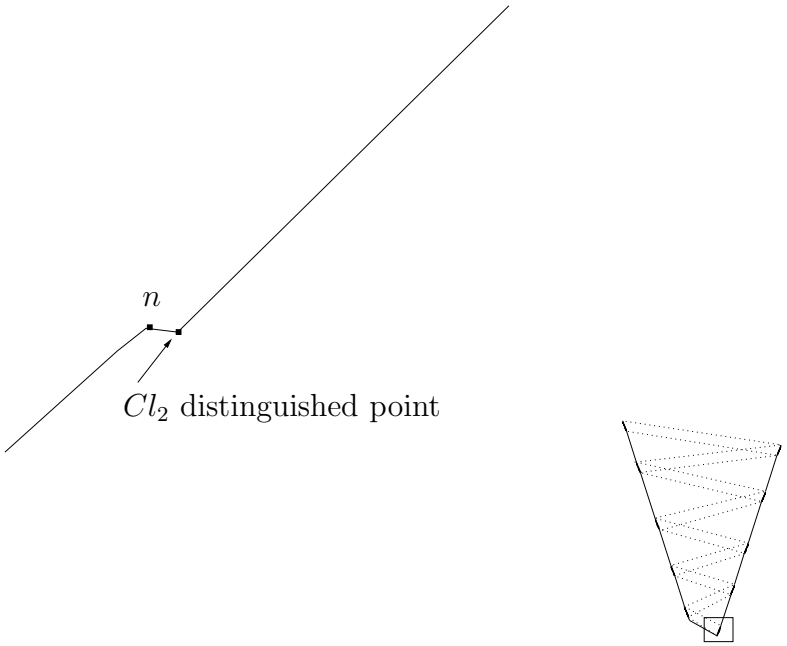


Figure 2.23: Chunk C_5 which places a clause gadget for clause Cl_2 and a deletion gadget for variable a and e.

gadget is replaced with a clause gadget for clause Cl_2. The purpose of the n point in Figure 2.23 was to ensure variable gadgets below d in chunk C_4 did not see into d's variable gadget. In this case, we want 1 other literal to see into d's variable region, which is now our Cl_2 distinguished point, that point being $e \in C_4$. The Cl_2 distinguished point is adjusted up accordingly. The literal $d \in C_4$ is also adjusted down accordingly. Referring back to Figure 2.22, we adjust the m point so that $\overline{a} \in C_4$ sees Cl_2's distinguished point. Recall that m was originally there to keep a from seeing into variable gadgets below a in chunk C_5. There are now 3 literal locations that can see Cl_2's distinguished point, namely $\overline{a} \in C_4, \overline{d} \in C_4$, and $e \in C_4$. If any of these literals have a guard at their location, Cl_2's distinguished point is seen and the clause is satisfied.

As said in the beginning of this section, we are combining several deletions in chunk C_5 to save space. We show the deletion of the a and e variable in Figure 2.23. The variable gadgets are simply replaced with flat surfaces.

This ends our reduction going downward and we count how many guards are necessary going downward. Because of the uniqueness claim, each variable gadget requires a guard to be placed at one of its literal guard locations. No other point on the terrain sees these variable distinguished points so a guard is required to be placed at a literal location for each variable gadget. We also add 1 extra guard for each inversion gadget that was placed. Recall that an inversion requires 1 extra guard because the inversion gadget has 2 "variable distinguished points". We count the number of variable gadgets in each chunk C_0, C_1, C_2, C_3, C_4, and C_5 and end up with

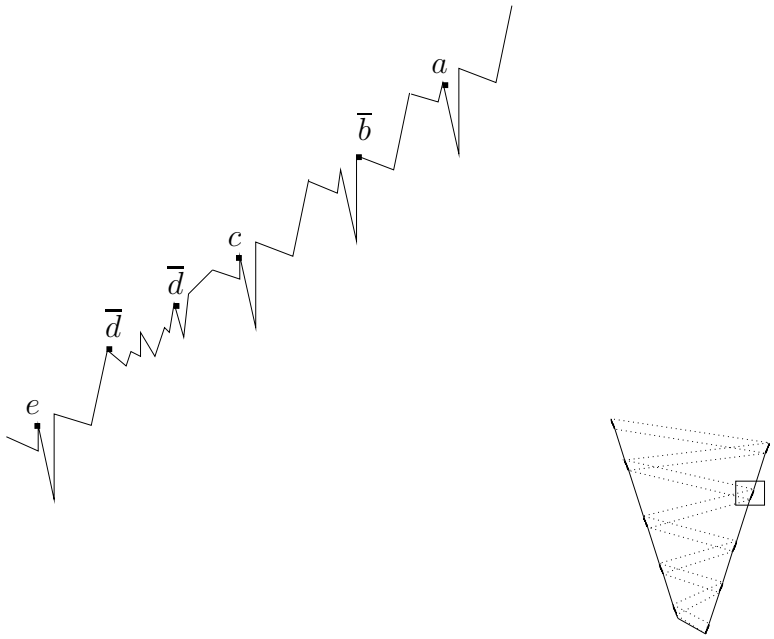


Figure 2.24: Chunk C_{-1} which places an inversion gadget for variable d.

$5 + 4 + 3 + (3 + 1) + 3 + 0 = 19$.

We now consider placing gadgets going "up" the terrain. We wish to place a gadget for clause Cl_3 on our terrain but we must invert variable d first. Figure 2.24 shows chunk C_{-1} and the inversion of d.

We can now place a clause gadget for $Cl_3 = (b \vee c \vee d)$. We place this gadget in chunk C_{-2} as shown in Figure 2.25. The mirrored distinguished point for c is replaced with Cl_3's distinguished point. Since we are deleting c in this chunk, we flatten out the other mirrored distinguished point for c. We move Cl_3's distinguished point accordingly so that it sees the d guard location in chunk C_{-3}.

The changes for the clause gadget are continued in chunk C_{-3} as shown in Figure 2.26. The change we must make is adjusting the m point as seen before. The

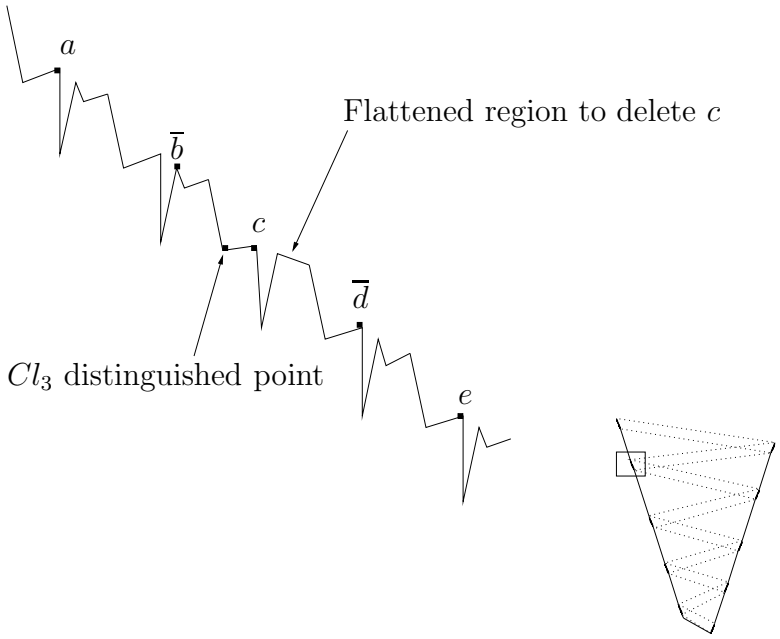


Figure 2.25: Chunk C_{-2} which places a clause gadget for clause Cl_3 and also deletes variable c.

original purpose of m was to ensure $b \in C_{-3}$ did not see other variables mirrored distinguished points to the right of $b \in C_{-2}$. However, this changes because we want $b \in C_{-3}$ to see Cl_3's distinguished point. The only guard locations that see Cl_3's distinguished point are $c \in C_{-2}$, $b \in C_{-3}$ and $d \in C_{-3}$. We also delete the e variable in chunk C_{-3} to save space. To delete e, the mirrored distinguished points for $e \in C_{-3}$ are flattened out.

Chunk C_{-4} is shown in Figure 2.27. In this chunk we place the clause gadget for Cl_4. We replace variable gadget $b \in C_{-4}$ with a clause gadget for Cl_4. Cl_4's distinguished point is adjusted accordingly so that it sees the d literal guard location in chunk C_{-5}. Since this is the last clause to be placed and to save space, the a and d variables are deleted in chunk C_{-5} as shown in Figure 2.28. Adjustments to the

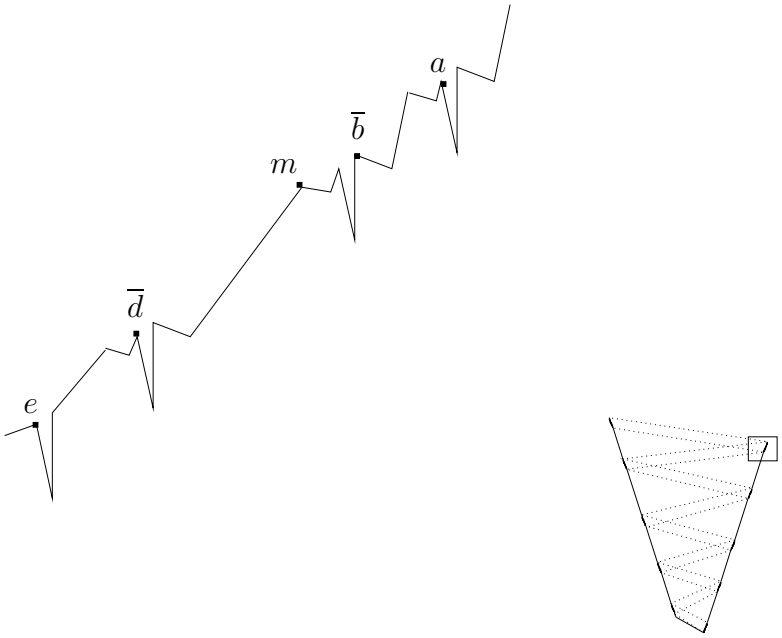


Figure 2.26: Chunk C_{-3} which places a deletion gadget for variable e.

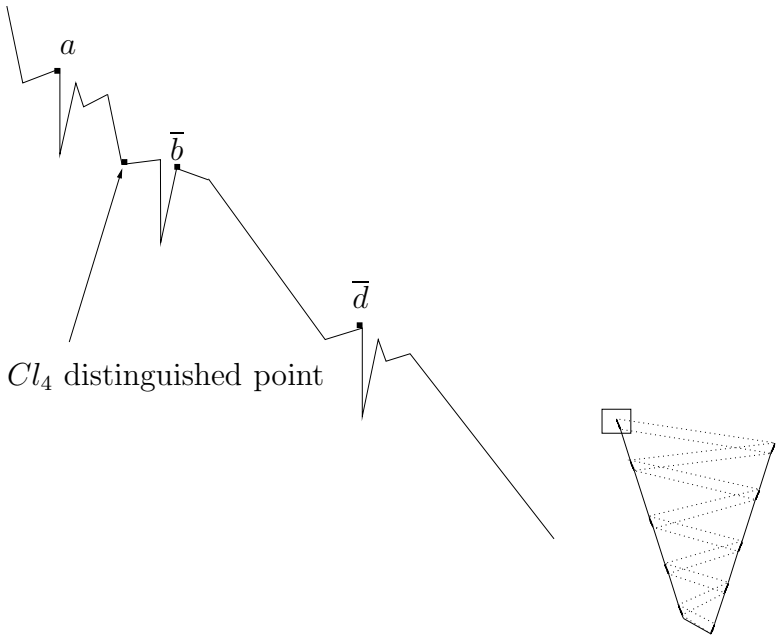


Figure 2.27: Chunk C_{-4} which places a clause gadget for clause Cl_4 and also deletes variable b.

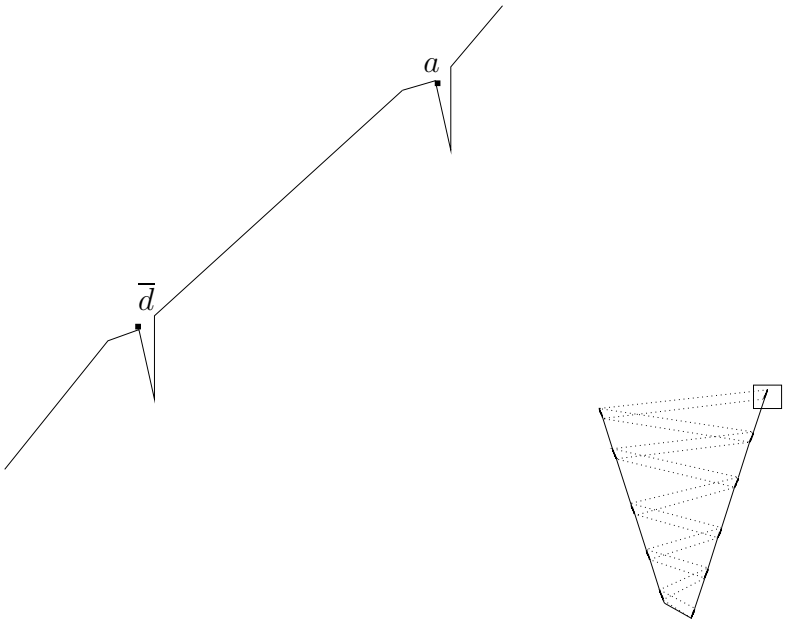


Figure 2.28: Chunk C_{-5} which places a deletion gadget for variables a and d.

terrain in chunk C_{-5} are made similarly as before so that a sees Cl_4's distinguished point. The only guard locations that see Cl_4's distinguished point are $b \in C_{-4}$, $a \in C_{-5}$ and $d \in C_{-5}$.

This ends our reduction going upward and we count how many guards are necessary going upward. We note that each variable gadget requires a guard to be placed at one of the literal points for that particular variable gadget because of the *Uniqueness Claim*. No other point on the terrain sees these variable distinguished points so a guard is required to be placed at a literal location for each variable gadget. We also add 1 extra guard for each inversion gadget that was placed. Recall that an inversion requires 1 extra guard. We count the number of variable gadgets in each chunk $C_{-1}, C_{-2}, C_{-3}, C_{-4}$, and C_{-5} and end up with $(5+1)+5+4+3+2 = 20$.

The entire terrain needs at least 39 guards. However 39 guards are sufficient

if the planar 3SAT instance is satisfiable. Assuming consistent choices were made in mirroring, no extra guards are required to see the mirrored distinguished points. If the planar 3SAT instance is satisfiable, the entire terrain can be guarded with 39 guards because the clause distinguished points will also be seen. If more than 39 guards are required to see the entire terrain, the planar 3SAT instance is not satisfiable.

CHAPTER 3
TERRAIN GUARDING:4-APPROXIMATION ALGORITHM

In this chapter we present a 4-approximation for the discrete version of the terrain guarding problem. It is also worth noting that other algorithms are much more complex than our algorithm. Our algorithm is a 4-approximation that runs in polynomial time.

3.1 Exact Algorithm For One Sided Guarding

In this section, we provide an exact algorithm for one sided guarding. Recall that the input for the discrete version of the terrain guarding problem is a set of vertices $P = \{v_1, v_2, ..., v_n\}$, a set of candidate guards G and a set of points to be seen C. For left sided guarding, any point $c \in C$ must see a guard that is to the left of c. In other words, for each $c \in C$, there must exist a guard $g \in G$ such that $g \leq c$ and g sees c. We say that a point $p < q$ if the x-coordinate of p is less than the x-coordinate of q. The leftmost vertex of our terrain T will always be a guard. The leftmost guard that sees v_i is denoted $L(v_i)$. We use the following lemma to show that it is possible to select the optimal set of guards for one sided guarding in polynomial time.

Lemma 4. *Let T' be any nonempty set of vertices on the terrain that must be guarded. 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 repeat. We note 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 $L(p) \leq L(r) < r < p$. □

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. $T'' \leftarrow T' \setminus \bigcup t \in T'$ such that $x_{L(p)} < x_t$ and $L(p)$ sees t.
4. Return leftSideTG(T'') $\cup$ $L(p)$.

3.2 Proof of Algorithm

We prove by induction 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 need to be guarded. 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. This leads us to the following lemma:

Lemma 5. *$L(p)$ sees any point in T' that g sees.*

Proof. 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 that is seen by p but not by $L(p)$ and $p < q$. q must then lie below the ray shot from $L(p)$ through 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) works similarly and is shown below.

rightSideTG(T')

1. If T' is empty, return $\emptyset$.

2. 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 $p < r < R(r) \leq R(p)$.

3. $T'' \leftarrow T' \setminus \bigcup t \in T'$ such that $t < R(p)$ and $R(p)$ sees t.

4. Return rightSideTG(T'') $\cup$ $R(p)$.

3.3 Algorithm Example

Consider Figure 3.1 to help explain Lemma 4 and the algorithm leftSideTG. Initially $T' = \{a, b, c, d\}$. 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. $T' \leftarrow T' \setminus \{a, b\}$, therefore $T' = \{c, d\}$. In the next iteration of leftGuardTG, c is initially chosen 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, d becomes our new p. We now place a guard at $L(d)$ and continue on. These are the two cases in picture form.

Theorem 6. *There exists a polynomial time algorithm that optimally guards a terrain from the left (resp. right).*

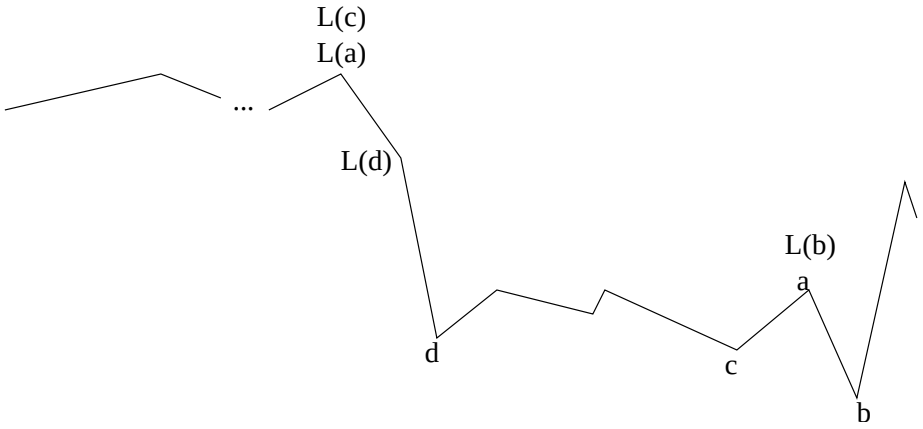


Figure 3.1: Sample terrain.

3.4 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:

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

Each guard i may be thought of as a fractional guard whose fraction is w_i. The constraint above states that each vertex $y \in T'$ sees a set of fractional guards on the terrain; the sum of those fractional guards must be greater than or equal to 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 or 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 $x \in L$ 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 $x \in R$ from the right. We return the union of these two sets of guards. The 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. $L'' \leftarrow L' \setminus \bigcup t \in L'$ such that $t < R(p)$ and $R(p)$ sees t.

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 $L(p) \cup \text{leftGuard}(L'', l'')$.

Lemma 7. *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. The proof is 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, $\sum_{i=1}^{n} l'' \leq \sum_{i=1}^{n} l'_i - 1$. Using Lemma 5, 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 8. *There exists a polynomial time algorithm that provides a 4-approximation to the vertex terrain guarding problem.*

CHAPTER 4
TERRAIN GUARDING:$(1+\epsilon)$-APPROXIMATION ALGORITHM

This chapter considers the discrete terrain guarding problem and shows a polynomial time algorithm that returns a guard cover whose cardinality is at most $(1+\epsilon)\cdot OPT$ for any $\epsilon > 0$. Here, OPT denotes the cardinality of an optimal guard cover.

The inspiration for our work comes from the recent results of Chan and Har-Peled [7] and Mustafa and Ray [38]. Chan and Har-Peled show that a local search algorithm actually yields a PTAS for the maximum independent set problem given a collection of disks. Unlike a previous PTAS for the problem [19], their analysis does not use packing arguments and thus also applies to "pseudo-disks". Mustafa and Ray consider several geometric hitting set and set cover problems and describe local search algorithms that yield PTASs. For instance, in a rather surprising result they obtain a PTAS for the problem of covering a set of points by the smallest number of a given set of disks. Both papers use separator theorems for planar graphs. In particular, they show that there exists a planar graph that relates the locally optimal solution returned by the local search and the global optimal solution. The separator theorem is then used to show that the locally optimal solution is not too much worse than the global optimum.

Our PTAS for the terrain guarding problem is also based on local search. Our key contribution is to show the existence of an appropriate planar graph even for the terrain guarding context. Having shown this, the rest of the analysis is very similar

to that of Mustafa and Ray [38].

4.1 Guarding Terrains via Local Search

Recall that our input is a polygonal terrain, a set X of points on the terrain that need to be guarded, a set G of possible guard locations, and a parameter $0 < \epsilon < 1$. For purposes of exposition, we will assume that $X \cap G = \emptyset$. We describe a polynomial time algorithm that returns a subset $Q \subseteq G$ that sees X, so that $|Q|$ is at most a factor $(1+\epsilon)$ times the size of the smallest subset of G that sees X. Let n denote the input size – the number of vertices in the terrain, plus $|G|$, plus $|X|$.

We say that a subset of G that sees X is b-locally optimal if one cannot obtain a smaller set of guards that sees X by deleting at most b guards from it and inserting at most $b-1$ guards.

Our algorithm simply returns a b-locally optimal solution for $b = \frac{\alpha}{\epsilon^2}$, where α is a suitably large constant, by performing local search. We start with some arbitrary $Q \subseteq G$ that covers X. For every subset $S \subseteq Q$ of size at most b, we see if there exists a subset $T \subseteq G \setminus Q$ of size at most $|S| - 1$ such that $(Q \setminus S) \cup T$ guards X. If so, we set $Q \leftarrow (Q \setminus S) \cup T$. Every such exchange decreases the size of Q by at least one, and as such can happen at most n times. Since there are $\binom{n}{b}$ subsets S to consider, the running time is bounded by $n^{O(b)}$.

4.1.1 Approximation Analysis

Let R' denote the optimal cover for X, and B' the set of guards output by our local search algorithm on termination. We show that $|B' \setminus R'| \leq (1+\epsilon)|R' \setminus B'|$, and

thus $|B'| \leq (1+\epsilon)|R'|$. Let $R \equiv R' \setminus B'$, $B \equiv B' \setminus R'$, and abusing notation, let X denote the set after removing all points seen by $R' \cap B'$. So now both R and B cover X and we wish to show that $|B| \leq (1+\epsilon)|R|$. We will refer to points in B as blue points and points in R as red points.

The following lemma is our main contribution; it shows that the *locality condition* of Mustafa and Ray [38] is satisfied.

Lemma 9. *There exists a planar graph $\mathcal{G} = (V \equiv R \cup B, E)$ with the property that for each $x \in X$, there is an edge (r, b) in $\mathcal{G}$ between guards $r \in R$ and $b \in B$ that both see x.*

Before giving the proof, we show how the lemma implies that $|B| \leq (1+\epsilon)|R|$; this is similar to [7, 38]. We need the following partition theorem on planar graphs due to Frederickson [21]. For $U \subseteq V$, let $\Gamma(U)$ denote the set of neighbors in $\mathcal{G}$ of vertices in U with U excluded. Let $\mu = |V|$.

Lemma 10. *For any parameter $1 \leq r \leq \mu$, we can find a set $S \subseteq V$ of size at most $c_1\mu/\sqrt{r}$ and a partition of $V \setminus S$ into μ/r sets $V_1, V_2, \ldots, V_{\mu/r}$, satisfying (i) $|V_i| \leq c_2 r$, (ii) $|\Gamma(V_i)| \leq c_3\sqrt{r}$, and (iii) $(V_i \cup \Gamma(V_i)) \cap V_j = \emptyset$ for $i \neq j$. Here, c_1, c_2, and c_3 are absolute positive constants.*

Let us apply the lemma with $r \equiv b/(c_2 + c_3)$. We have $|V_i \cup \Gamma(V_i)| \leq c_2 r + c_3\sqrt{r} \leq b$. Thus, letting $R_i = R \cap V_i$ and $B_i = B \cap V_i$, we must have $|B_i| \leq |R_i| + |\Gamma(V_i)|$. For otherwise, the local search can replace B_i by $R_i \cup \Gamma(V_i)$ and obtain a smaller set that still covers X (Lemma 9), a contradiction.

Thus

$$\begin{aligned} |B| &\leq |S| + \sum_i |B_i| \leq |S| + \sum_i |R_i| + \sum_i |\Gamma(V_i)| \leq |R| + c\frac{\mu}{\sqrt{r}} \\ &\leq |R| + c'\frac{|R|+|B|}{\sqrt{b}}, \end{aligned}$$

where c and c' are positive constants. With b a large enough constant times $1/\epsilon^2$, this implies that $|B| \leq (1+\epsilon)|R|$.

4.1.2 Proof of Lemma 9

We begin with some notation. For points a and b on the terrain, we say $a \leq b$ to mean that the x-coordinate $a.x$ of a is at most $b.x$. We use the notation of intervals that this implies – for instance, $[a, b]$ denotes all points c on the terrain so that $a \leq c \leq b$.

We now prove Lemma 9. Let us first construct the planar graph $\mathcal{G}$. For each $x \in X$, let $\lambda(x)$ denote the leftmost point that sees x among points in $R \cup B$ to the left of x, assuming such a point does exist. Similarly, let $\rho(x)$ denote the rightmost point that sees x among points in $R \cup B$ to the right of x, assuming such a point does exist. Note that at least one of $\lambda(x)$ or $\rho(x)$ does exist.

Let A_1 denote the set of segments $\lambda(x)x$, for $x \in X$. Because of the order claim, these segments do not cross. For each $v \in R \cup B$, shoot a vertical ray up from v; if this ray hits some segment in A_1, let $\lambda(y)y$ denote the first such segment hit; we add the edge $(v, \lambda(y))$ to a set E_1 if v and $\lambda(y)$ are of opposite colors.

Now, the edges in $A_1 \cup E_1$ can be embedded above the terrain in a non-crossing way. To see this, let A_1 be embedded as the original straight line segments. To embed

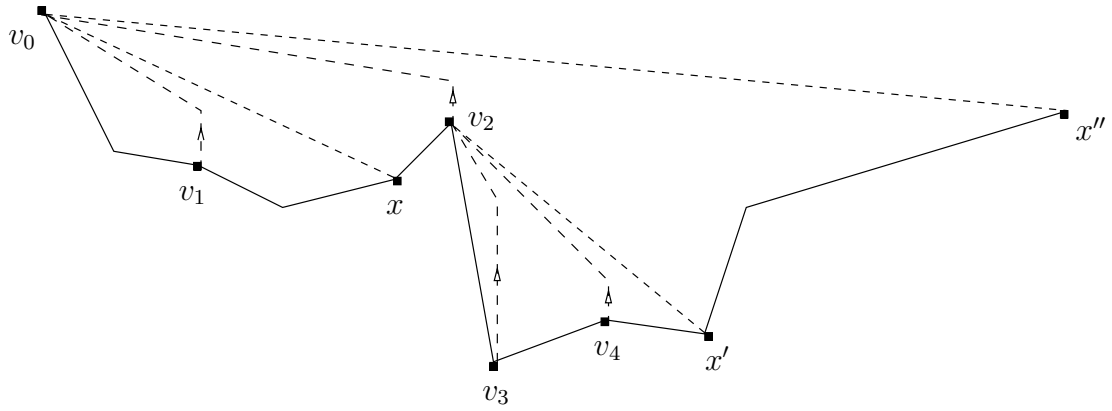


Figure 4.1: The embedding of $A_1 \cup E_1$, with $X = \{x, x', x''\}$, and $R \cup B = \{v_0, v_1, v_2, v_3, v_4\}$. Segments in A_1 are shown in dashed lines, and the edges in E_1 are embedded as dashed curves with arrows. Note that $v_0 = \lambda(x) = \lambda(x'')$, and $v_2 = \lambda(x')$.

an edge of the form $(v, \lambda(y)) \in E_1$ as above, we travel straight up from v till we hit $\lambda(y)y$, and then slide along the segment $\lambda(y)y$ to reach $\lambda(y)$. See Figure 4.1.

Let A_2 denote the set of segments $x\rho(x)$, for $x \in X$. Again, these segments do not cross. For each $v \in R \cup B$, shoot a vertical ray up from v; if this ray hits some segment in A_2, let $y\rho(y)$ denote the first such segment hit; we add the edge $(v, \rho(y))$ to a set E_2 if v and $\rho(y)$ are of opposite colors.

The edges in $A_2 \cup E_2$ can also be embedded above the terrain in a non-crossing way. We "flip" the embedding of $A_1 \cup E_1$ to obtain a non-crossing embedding below the terrain; see Figure 4.2. This gives us a planar embedding of $A_1 \cup E_1 \cup A_2 \cup E_2$.

Finally, for each $x \in X$, we add the edge $(\lambda(x), \rho(x))$ to a set E_3 if $\lambda(x)$ and $\rho(x)$ are of opposite colors. Our graph $\mathcal{G}$ consists of the edge set $E_1 \cup E_2 \cup E_3$. This is a planar graph; just embed E_1 and E_2 as above, and for each $(\lambda(x), \rho(x)) \in E_3$, embed it using the embedding of the segments $\lambda(x)x$ and $x\rho(x)$.

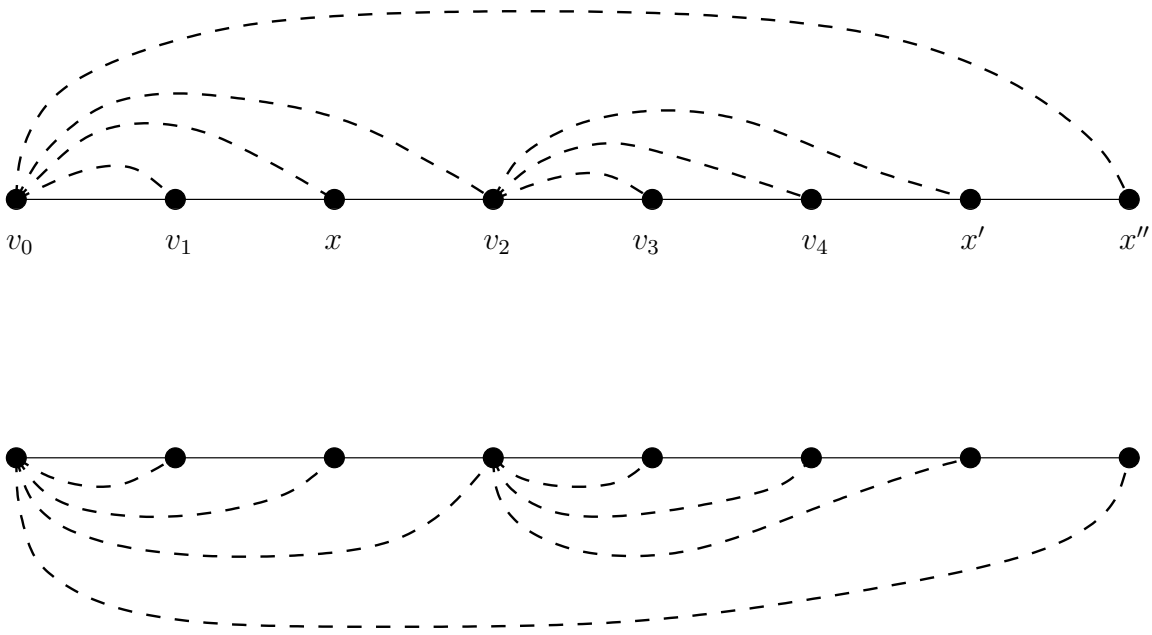


Figure 4.2: A combinatorial embedding of $A_1 \cup E_1$ from Figure 4.1, and flipping it so that $A_1 \cup E_1$ is now embedded below the terrain. Note that only the edges in $A_1 \cup E_1$ are being flipped to make room for $A_2 \cup E_2$; the vertex set $R \cup B \cup X$ retains its embedding.

Now we need to show that for each $x \in X$, there are points $r \in R$ and $b \in B$ that see x, and $(r, b) \in E_1 \cup E_2 \cup E_3$. Fix an $x \in X$. If $\lambda(x)$ and $\rho(x)$ are of opposite colors, then $(\lambda(x), \rho(x)) \in E_3$, and we are done. Otherwise, it must be the case that there are red and blue points to the left of x that see x, or that there are red and blue points to the right of x that see x.

Let us assume that the first case holds (there are red and blue points to the left of x that see x), and that $\lambda(x)$ is red. The other situations are symmetric. Let b be the leftmost blue point that sees x; it must be that $b \in (\lambda(x), x)$. Thus the ray shot up from b hits $\lambda(x)x$; let $\lambda(y)y$ be the first segment in A_1 that it hits. Because segments in A_1 don't cross, it must be that $\lambda(y) \in [\lambda(x), b)$ and $y \in (b, x)$. The order

claim (applied to $\lambda(y)$, b, y, and x) implies that $\lambda(y)$ sees x. Now $\lambda(y)$ cannot be blue, otherwise b is not the leftmost blue point that sees x. Thus $\lambda(y) \in R$, $b \in B$, both $\lambda(y)$ and b see x and $(b, \lambda(y)) \in E_1$. This completes the proof.

CHAPTER 5
APPROXIMATE GUARDING OF MONOTONE POLYGONS

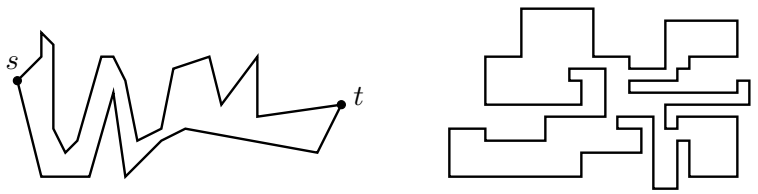
In this chapter, we show in Section 5.2 that the problem of finding the smallest number of vertex guards to guard a monotone polygon is NP-hard. We also give a constant factor approximation to the art gallery problem using monotone polygons in Section 5.3.

5.1 Definitions

An instance of the art gallery problem contains a polygon P. The polygon is defined by a set of points $V = \{v_1, v_2, ..., v_n\}$. There are also edges connecting (v_i, v_{i+1}) where $i = 1, 2, ..., n-1$. There is also an edge connecting (v_n, v_1). These edges give us two disjoint regions: inside and polygon and outside the polygon. For any two points $p, q \in P$, we say that p sees q if the line segment $\overline{ab}$ does not go outside of P. We wish to find a set of vertices $G \subseteq P$ such that every point $p \in P$ is seen by a guard in G. We call this set G a guarding set. The optimization problem is thus defined as finding the smallest such G.

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 5.1.

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



monotone rectilinear

Figure 5.1: Illustrating the polygon classes.

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 all vertices, the interior angle between segments are either 90 or 270 degrees; see Figure 5.1.

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. Let q be a point in $\mathbf{VP}(p)$ that lies to the right of p. We denote by $\mathbf{VP}_R(p, q)$ the part of $\mathbf{VP}(p)$ that

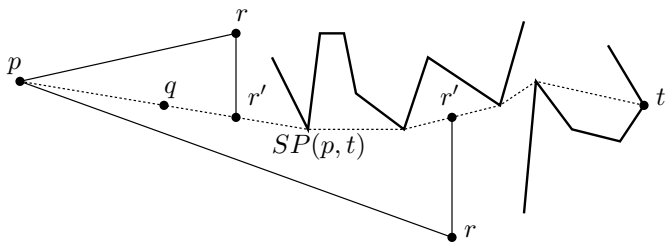


Figure 5.2: Illustrating the proof of Lemma 11.

lies to the right of q. Also, $\mathbf{VP}_R(p) = \mathbf{VP}_R(p, p)$.

A pocket in a monotone polygon $\mathbf{P}$ is an *upper pocket* if it is adjacent to the upper boundary U of $\mathbf{P}$, a *lower pocket* if it is adjacent to the lower boundary D but not U and a *middle pocket* if it is adjacent to neither U nor D.

Let $SP(p, q)$ denote the shortest (Euclidean) path between points p and q inside $\mathbf{P}$.

Lemma 11. *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 5.2. □

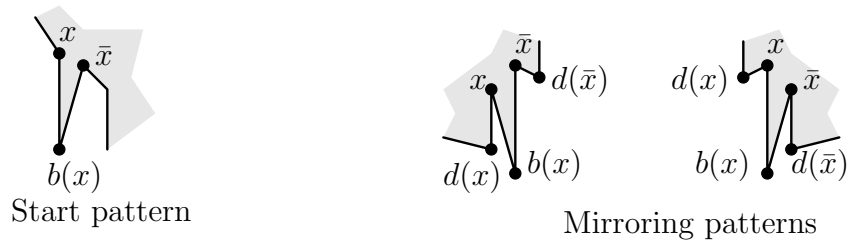


Figure 5.3: The two types of variable patterns.

5.2 Vertex Guarding Monotone Polygons:NP-Hardness

In this section we will show that vertex guarding a monotone polygon is NP-complete. The reduction is from 3SAT. A 3SAT instance $(\mathcal{X}, \mathcal{C})$ contains 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, $c_i = l_{i,1} \vee l_{i,2} \vee l_{i,3}$ so that $l_{i,j} = x_k$ or $l_{i,j} = \bar{x}_k$. A 3SAT instance is satisfiable if a satisfying truth assignment for $\mathcal{C}$ exists such that all clauses c_i are true.

We will show that any 3SAT instance is polynomially transformable to an instance of vertex guarding a monotone polygon. We will construct a monotone polygon P from the 3SAT instance such that P is guardable by K or fewer guards if and only if the 3SAT instance is satisfiable. We will first present some basic constructs to show how the polygon will be constructed. We will then connect them to create a polygon. Let $K = n\,(m+1)$, n is the number of variables and m the number of clauses of the 3SAT instance.

Starting Variable Pattern: The starting variable pattern is shown to the left in Figure 5.3. The bottom of the downward spike $b(x)$ of the figure is the distinguished

vertex of the pattern. This area is only seen by x and $\bar{x}$ and must be guarded by one of these two vertices. This pattern will appear along the lower boundary of the monotone polygon a total of n times, one corresponding to each variable.

Mirroring Variable Pattern: This variable pattern is shown to the right in Figure 5.3. Once again, the bottom of the spike at $b(x)$ must be guarded by either x or $\bar{x}$. We also note that vertices $d(x)$ and $d(\bar{x})$ must both be seen and this is what forces our choice of guard at x or $\bar{x}$. In the final polygon mn of these patterns will be present also along the lower boundary.

Figure 5.4 shows how the starting variable patterns are connected to mirroring variable patterns. If we choose x_j in the starting variable pattern, we are forced to continuing to choose x_j in each of subsequent mirroring variable patterns. If we at some mirroring pattern would choose $\bar{x}_j$ instead of x_j, the distinguished points $d(x_j)$ 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 mirroring variable patterns.

Clauses: For each clause c in the boolean formula, there is a sequence of mirroring variable patterns $x_1, \bar{x}_1, x_2, \bar{x}_2, \ldots, x_n, \bar{x}_n$ along the lower boundary and a clause pattern along the upper boundary of the polygon. The clause pattern consists of three vertices in an upward spike such that the top vertex of the spike is only seen by the mirroring variable patterns corresponding to the literals in the clause; see Figure 5.5. We denote the top vertex of the spike by c to correspond

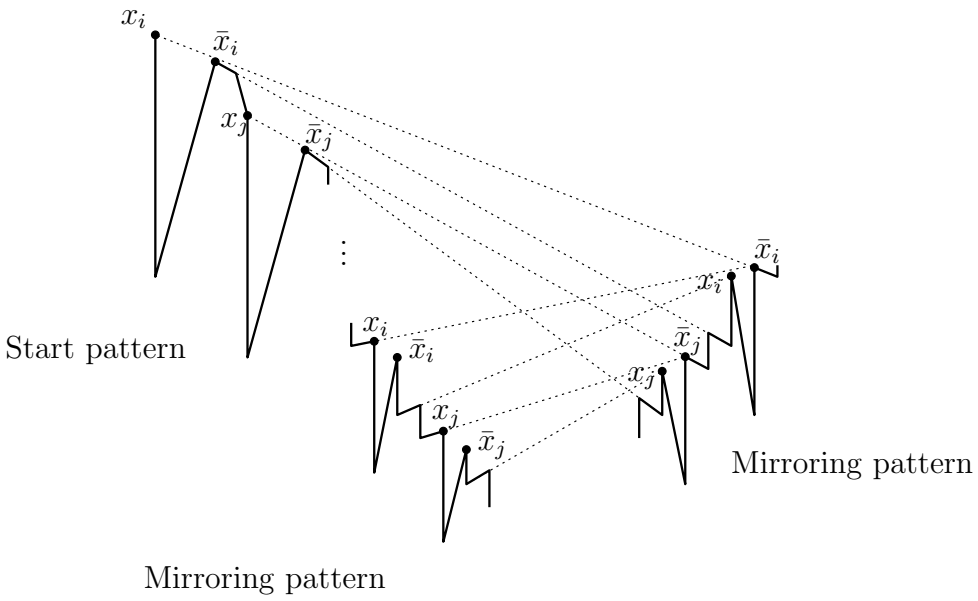


Figure 5.4: Variable patterns transferring logical values.

to the clause.

We choose our truth value for each variable in the starting variable patterns. The subsequent mirroring variable patterns transfer that decision along to all the other mirroring variable patterns on the lower boundary of the polygon; see Figure 5.4.

In the example of Figure 5.5 the 3SAT clause corresponds to $c = x_1 \vee \bar{x}_3 \vee x_5$. Hence, a vertex guard placement that corresponds to a truth assignment that makes c true, will have at least one guard on x_1, $\bar{x}_3$ or x_5 and can therefore see vertex c without additional guards.

Note that 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 simply there to mirror their truth values along in case these variables are needed in later clauses.

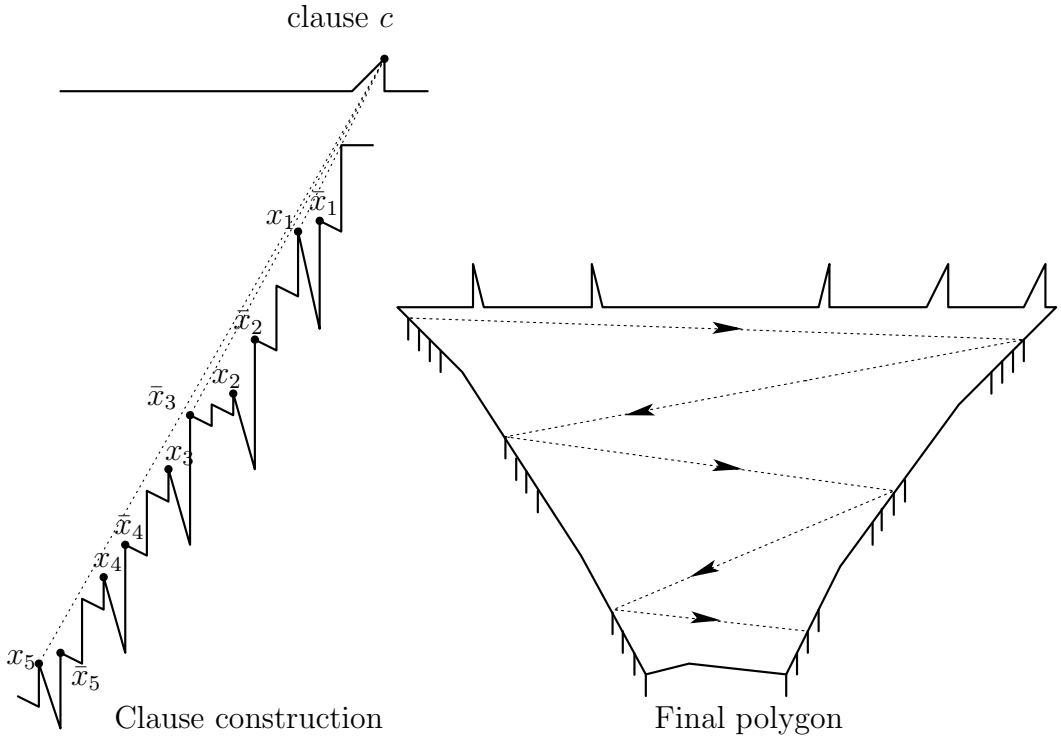


Figure 5.5: A mirroring variable pattern sequence with its clause spike.

The monotone polygon we construct consists of $4n + (6n + 3)m + 1$ vertices. Each starting variable pattern having four vertices, each mirroring pattern six vertices, the clause spike consists of three vertices and a final vertex connecting the distinguished edges of the two lowermost mirroring patterns on the lower boundary.

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 3SAT 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.

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. This 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.

It is easy to see that the construction of the polygon can be done in polynomial time so we have proved the following theorem.

Theorem 12. *Finding the smallest vertex guard cover for a monotone polygon is NP-hard.*

Note that our proof does not immediately generalize to interior guards. Proving NP-hardness for interior guards in monotone polygons seems to require completely different techniques. In the next section, we show how to approximate the minimum number of interior guards in a monotone polygon.

5.3 Interior Guarding Monotone Polygons

In this chapter we describe a polynomial time algorithm for point guarding a monotone polygon. Our algorithm will return a solution that is no worse than $O(1)$*OPT where OPT is the number of guards in the optimal solution.

Our algorithm for guarding a monotone polygon **P** will incrementally guard **P** starting from the left and moving right. Hence, we are interested in the structure of the pockets that occur when guarding is done in this way. We first define *kernel*

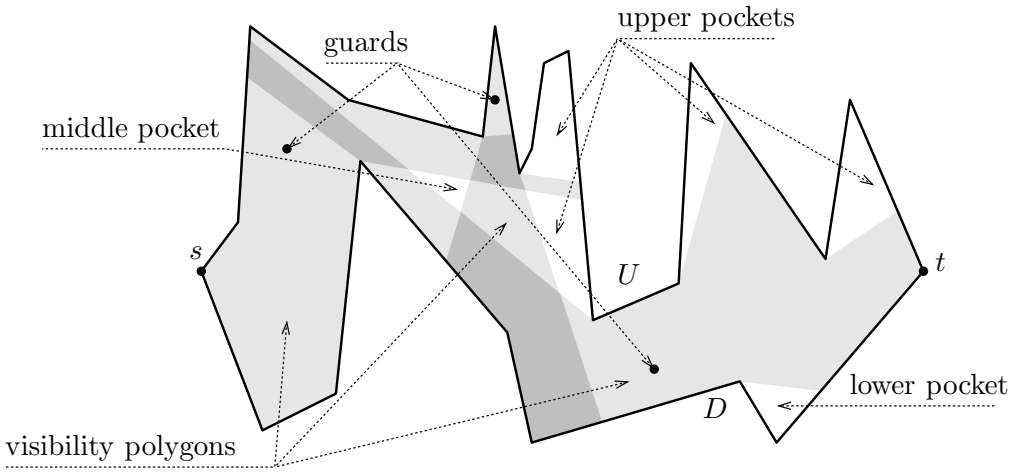


Figure 5.6: Illustrating pockets.

expansions of the pockets given a partial guard cover $\mathcal{G}^p$, and then taking maximal nonempty intersection of these we produce the main region that we will be interested in. This region is called a *spear* and with this we can define a well behaved guard cover $\mathcal{G}^*$ that has small size. We finally prove that our incremental algorithm produces a guard cover at most a constant times larger than $\mathcal{G}^*$.

Assume that we have a partial guard cover $\mathcal{G}^p$ in $\mathbf{P}$. Consider the upper pockets resulting from this guard cover and enumerate them $\mathbf{p}_1^U, \ldots$ from left to right. The lower pockets are enumerated from left to right $\mathbf{p}_1^D, \ldots$ in the same way; see Figure 5.6. We disregard any middle pockets for now and return to them later.

Let $\mathbf{p}^U$ be an upper pocket. The *kernel expansion* of $\mathbf{p}^U$, denoted $\mathbf{ke}(\mathbf{p}^U)$, consists of all the points in $\mathbf{P}$ that see everything in $\mathbf{p}^U$ to the left of themselves; see Figure 5.7. For the lower pockets we define the kernel expansion symmetrically. The

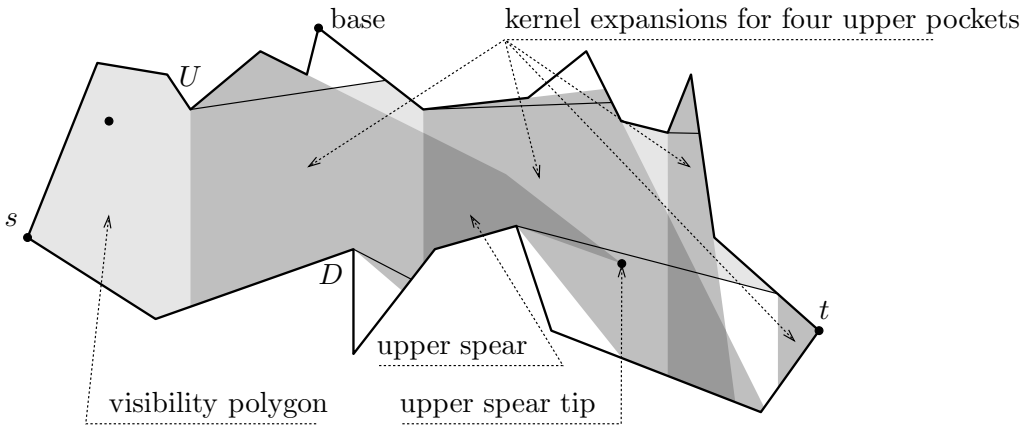


Figure 5.7: Illustrating the kernel expansions.

definition of kernel expansion is valid also when no guards have as yet been placed in the polygon. In this case, we take all of the polygon $\mathbf{P}$ to be a single upper pocket.

Let k be the largest index so that $\bigcap_{i=1}^{k} \mathbf{ke}(\mathbf{p}_i^U)$ is nonempty. This nonempty intersection of kernel expansions is called the *upper spear*, also denoted $\mathbf{sp}^U$; see Figure 5.7. We can in the same way define the *lower spear* $\mathbf{sp}^D$ as the maximal nonempty intersection of the kernel expansions for the lower pockets.

Given the partial guard cover $\mathcal{G}^p$ the upper spear $\mathbf{sp}^U$ can be computed in linear time as follows. For an upper pocket $\mathbf{p}_i^U$, we let r_i^U be the leftmost upper boundary point and q_i^U be the rightmost upper boundary point; see Figure 5.8. Traverse the edges of the upper boundary in the upper pockets $\mathbf{p}_1^U, \mathbf{p}_2^U, \ldots$ in order from s to t and for each boundary edge e having some part in a pocket issue a half line directed in the same direction as the traversal. When the last edge of a pocket $\mathbf{p}_i^U$ is reached we add the half line issuing from q_i^U towards the vertex w_i that maximizes the interior

angle without intersecting the exterior of $\mathbf{P}$ above U; see Figure 5.8. The point w_i is established as we traverse the U from q_i^U to r_{i+1}^U. This gives us a sequence of half lines

$$\{h_{e_{1,1}}, h_{e_{1,2}}, \ldots h_{e_{1,l_1}}, h_{[q_1^U, v_1]}, h_{e_{2,1}}, \ldots h_{e_{2,l_2}}, h_{[q_2^U, v_2]}, \ldots\}$$

where h_e is the half line issuing from edge e, $e_{i,j}$ is the j^{th} edge of the traversal inside pocket $\mathbf{p}_i^U$ and l_i is the number of upper boundary edges in $\mathbf{p}_i^U$.

Using these half lines in the order they were computed we incrementally find their right half plane intersection in the same way as is done to compute the kernel of a polygon [25, 34]. This gives us the upper boundary of the spear.

To compute the lower boundary of the spear we follow the lower boundary D of $\mathbf{P}$ from the point having the same x-coordinate as r_1^U toward t and maintain the half lines issuing from r_i^U, with $1 \leq i \leq k$, having maximal interior angle to U and such that they do not intersect the exterior of $\mathbf{P}$ below D; see Figure 5.8. Again k denotes the largest integer such that the union of the k first kernel expansions is nonempty. The intersection point between the upper and lower boundary of the spear is called the upper *spear tip* and we denote it u^U; see Figures 5.7 and 5.8.

In a similar manner we can compute the lower spear $\mathbf{sp}^D$ and its lower spear tip u^D.

To every spear $\mathbf{sp}$ we also associate a point called the *base* of the spear, denoted v^U and v^D depending on whether $\mathbf{sp}$ is an upper or lower spear. If $\mathbf{sp}$ is an upper spear, consider the spear tip u^U. It is the intersection point of the upper boundary of $\mathbf{sp}$ with its lower boundary. The last edge of the upper boundary of $\mathbf{sp}$ intersects the

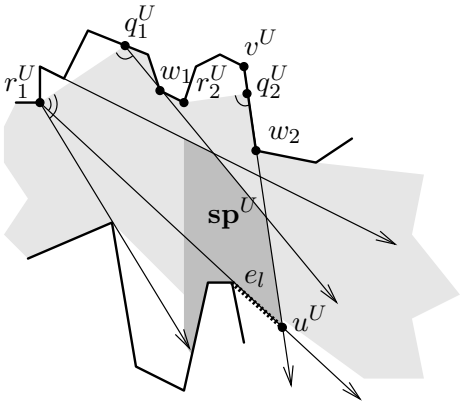


Figure 5.8: Computing the upper spear and the base.

lower boundary at u^U and this edge is a subset of a half line h issuing from an upper boundary point of the polygon as described above. This upper boundary point is the base v^U of the spear; see Figures 5.7 and 5.8. We can define the base v^D of a lower spear in a similar manner.

The upper and lower 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, $\mathbf{sp}^U(\mathcal{G}^p)$ and $\mathbf{sp}^D(\mathcal{G}^p)$ we denote the upper spear tip $u^U(\mathcal{G}^p)$, the lower spear tip $u^D(\mathcal{G}^p)$, the upper base $v^U(\mathcal{G}^p)$, and the lower base $v^D(\mathcal{G}^p)$. If $\mathcal{G}^p = \emptyset$, the upper spear $\mathbf{sp}^U(\emptyset)$, the upper spear tip $u^U(\emptyset)$, and the upper base $v^U(\emptyset)$ are well defined.

For an upper spear $\mathbf{sp}^U(\mathcal{G}^p)$, define the *shadow* of $\mathbf{sp}^U(\mathcal{G}^p)$, denoted $\mathbf{shd}^U(\mathcal{G}^p)$, to be the part of the visibility polygon of $v^U(\mathcal{G}^p)$ strictly to the right of $u^U(\mathcal{G}^p)$. Hence, $\mathbf{shd}^U(\mathcal{G}^p) = \mathbf{VP}_R(v^U(\mathcal{G}^p), u^U(\mathcal{G}^p))$. Similarly, $\mathbf{shd}^D(\mathcal{G}^p) = \mathbf{VP}_R(v^D(\mathcal{G}^p), u^D(\mathcal{G}^p))$ is the

shadow of a lower spear $\mathbf{sp}^D(\mathcal{G}^p)$.

We prove the following lemma.

Lemma 13. *Let $\mathcal{G}^p_-$ and $\mathcal{G}^p_+$ be two partial guard covers of $\mathbf{P}$ such that $\mathbf{sp}^U(\mathcal{G}^p_-)$ and $\mathbf{sp}^U(\mathcal{G}^p_+)$ do not intersect, then $\mathbf{shd}^U(\mathcal{G}^p_-) \cap \mathbf{shd}^U(\mathcal{G}^p_+) = \emptyset$.*

Proof. We make a proof by contradiction and assume that the two visibility polygons intersect. Assume further that $\mathbf{sp}^U(\mathcal{G}^p_-)$ lies to the left of $\mathbf{sp}^U(\mathcal{G}^p_+)$. Let p be a point in the intersection of the two visibility polygons. We can connect p to $u^U(\mathcal{G}^p_-)$ with a line segment and then follow the line segment from $u^U(\mathcal{G}^p_-)$ back to the base $v^U(\mathcal{G}^p_-)$ associated to $\mathbf{sp}^U(\mathcal{G}^p_-)$. From $v^U(\mathcal{G}^p_-)$ we follow the upper boundary of $\mathbf{P}$ to the base $v^U(\mathcal{G}^p_+)$ associated to $\mathbf{sp}^U(\mathcal{G}^p_+)$, from this point on to $u^U(\mathcal{G}^p_+)$, and then back to p. This traversal bounds a polygon interior to $\mathbf{P}$ that completely contains the lower boundary segment e_l of $\mathbf{sp}^U(\mathcal{G}^p_+)$ that intersects at $u^U(\mathcal{G}^p_+)$; see Figure 5.8. However, this is not possible because by construction this segment must intersect the lower boundary D of $\mathbf{P}$, giving us a contradiction; see Figure 5.9. □

At this point, it is important to note that a single guard placed in a spear will guard everything to the left of the guard in the pockets associated to the spear. However, the same region can also be guarded by placing at least two guards after the spear tip. It could therefore be possible for a guard cover to jump a sequence of spears and not have any guards in these. However, any guard cover that does this will have to pay by using more guards further to the right. To formalize this concept we bound from below the number of guards needed to the right of a sequence of spears

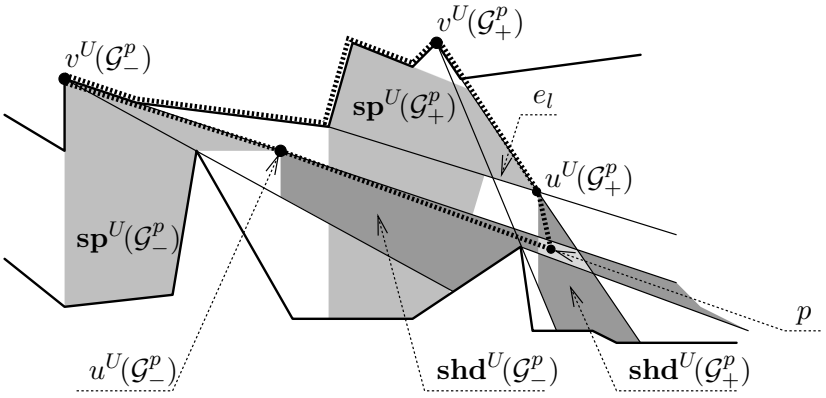


Figure 5.9: Illustrating the proof of Lemma 13.

in the case that no guard is in any of them.

Let $\mathcal{G}$ be any guard cover for $\mathbf{P}$ and let p be a point in $\mathbf{P}$. We partition $\mathcal{G}$ into two sets $\mathcal{G}^p$ and $\mathcal{G}^f$. $\mathcal{G}^p$ contains all the guards in $\mathcal{G}$ to the left of p, and $\mathcal{G}^f = \mathcal{G} \setminus \mathcal{G}^p$ contains the guards to the right of p. If $\mathcal{G}^p$ is nonempty assume that it has g as its rightmost guard and assume furthermore that all of $\mathbf{P}$ to the left of g is guarded by $\mathcal{G}^p$. We define the following sets recursively.

$$\begin{aligned} \mathcal{G}^p_0 &= \mathcal{G}^p \\ \mathcal{G}^p_i &= \mathcal{G}^p_{i-1} \cup \{u^U(\mathcal{G}^p_{i-1})\} \qquad \text{for } i > 0 \end{aligned}$$

Lemma 14. *Let $\mathcal{G}^p_0, \ldots, \mathcal{G}^p_k$ and $\mathcal{G}^f$ be sets as defined above. If all guards of $\mathcal{G}^f$ lie to the right of $u^U(\mathcal{G}^p_k)$, then $\mathcal{G}^f$ contains at least $k+1$ guards.*

Proof. By the construction of the sets $\mathcal{G}^p_i$, we know that their corresponding spears do not intersect, and hence, from Lemma 13 the corresponding shadows do not intersect either. Let g be the rightmost guard of $\mathcal{G}^p = \mathcal{G}^p_0$.

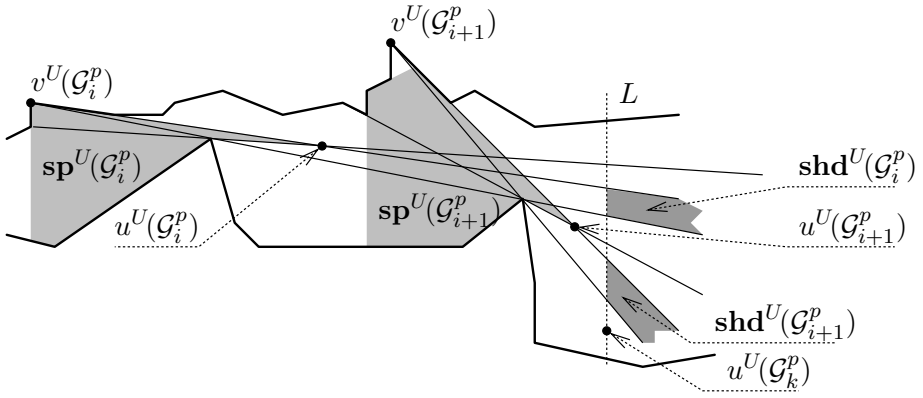


Figure 5.10: Illustrating the proof of Lemma 14.

Consider the vertical line L through $u^U(\mathcal{G}^p_k)$. If there is a shadow $\mathbf{shd}^U(\mathcal{G}^p_i)$ that does not intersect this vertical line then at least one guard is needed in the polygon between g and $u^U(\mathcal{G}^p_k)$, contradicting that $\mathcal{G}$ is a guard cover for $\mathbf{P}$. Hence, all shadows intersect this vertical line.

Since the $k+1$ shadows intersect L, they subdivide the part of $\mathbf{P}$ to the right of L into $k+1$ non-intersecting regions. Because the shadows are the only regions that see the bases $v^U(\mathcal{G}^p_0) \ldots v^U(\mathcal{G}^p_k)$ and since they do not intersect, at least one guard is required in each shadow $\mathbf{shd}^U(\mathcal{G}^p_i)$, giving us at least $k+1$ guards in $\mathcal{G}^f$; see Figure 5.10. □

We can, of course, prove similar results as those in Lemmas 13 and 14 for the shadows of lower spears.

Let us define the concept of a *serial* guard cover $\mathcal{G}^s$. First, we let, for each guard g in $\mathcal{G}^s$, the set $\mathcal{G}^s(g)$ be the guards of $\mathcal{G}^s$ with smaller or equal x-coordinate

than g, and we let $\mathcal{G}^s_{\_}(g)$ be the guards of $\mathcal{G}^s$ with strictly smaller x-coordinate than g. Hence, the guards in $\mathcal{G}^s(g)$ lie to the left of g and the guards in $\mathcal{G}^s_{\_}(g)$ lie properly to the left of g. A guard cover $\mathcal{G}^s$ is serial if the following two invariant conditions hold for all guards g in $\mathcal{G}^s$:

1. The guards of $\mathcal{G}^s(g)$ see everything of $\mathbf{P}$ to the left of g.

2. There is a guard in either $\mathbf{sp}^U(\mathcal{G}^s_{\_}(g))$ or in $\mathbf{sp}^D(\mathcal{G}^s_{\_}(g))$. Note that this guard is necessarily g but can be another guard with the same x-coordinate.

The next lemma shows that there is a serial guard cover of small size.

Lemma 15. *If $\mathcal{G}$ is a guard cover for the monotone polygon $\mathbf{P}$, then there is a serial guard cover $\mathcal{G}^s$ for $\mathbf{P}$ such that $|\mathcal{G}^s| \leq 5|\mathcal{G}|$.*

Proof. Given a guard cover $\mathcal{G}$ we transform it to be serial as follows. Order the guards of $\mathcal{G} = \{g_1, \ldots, g_m\}$ from left to right. The transformation places guards into three sets $\mathcal{G}^U$, $\mathcal{G}^D$, and $\mathcal{G}^M$ ensuring that $\mathcal{G}^s = \mathcal{G}^U \cup \mathcal{G}^D \cup \mathcal{G}^M$ is serial.

To make the constructed guard set serial we employ a plane sweep approach moving from left to right. As soon as the sweep line reaches a guard in $\mathcal{G}$, the guard is attached to the sweep line and moves along it following the shortest path to t. By Lemma 11 this does not decrease visibility to the right. As the sweep proceeds we will attach new ones as they are reached and release guards when necessary. The release of a guard g from the sweep line gives us the position of the corresponding guard g^s in $\mathcal{G}^M$. We place additional guards as necessary in the sets $\mathcal{G}^U$ and $\mathcal{G}^D$.

Consider the structural changes that occur when the sweep line moves from left to right. Let $\mathcal{G}^s_-$ be the current set of guards released from the sweep line. Two things can happen:

1. A guard $g \in \mathcal{G}$ becomes the last guard to leave a spear with respect to the previously released guards in $\mathcal{G}^s_-$, $\mathbf{sp}(\mathcal{G}^s_-)$, and it is then released from the sweep line, otherwise not all points in $\mathbf{P}$ to the left of g are guarded. A guard g^s positioned at the release point of g is added to $\mathcal{G}^M$.

2. The sweep line reaches the spear tip $u(\mathcal{G}^s_-)$ without having released a guard. In this case, the spear $\mathbf{sp}(\mathcal{G}^s_-)$ is empty of guards. We add a guard g^s_- at the position of the spear tip $u(\mathcal{G}^s_-)$ to one of $\mathcal{G}^U$ and $\mathcal{G}^D$, such that g^s_- is in $\mathcal{G}^U$, if $\mathbf{sp}(\mathcal{G}^s_-)$ is an upper spear and in $\mathcal{G}^D$, if $\mathbf{sp}(\mathcal{G}^s_-)$ is a lower spear. In addition, we release an arbitrary guard g from the sweep line, if there is one attached to it, and add a guard g^s to $\mathcal{G}^M$ positioned at g.

When the sweep line reaches t, those guards still attached to it are removed (except for possibly one) giving us the serial guard cover $\mathcal{G}^s$.

To count the number of guards placed by this process we have immediately that $|\mathcal{G}^M| \leq |\mathcal{G}|$. Furthermore, from Lemma 14, we can associate each guard in $\mathcal{G}^U$ and $\mathcal{G}^D$ either with the one released at the same x-coordinate or one inside the shadow of the associated spear. Since no two shadows intersect we have that a guard in $\mathcal{G}$ is associated to at most two in $\mathcal{G}^U$ and two in $\mathcal{G}^D$. Hence, $|\mathcal{G}^U| \leq 2|\mathcal{G}|$ and $|\mathcal{G}^D| \leq 2|\mathcal{G}|$ giving us that $|\mathcal{G}^s| \leq |\mathcal{G}| + |\mathcal{G}^U| + |\mathcal{G}^D| \leq 5|\mathcal{G}|$. □

Algorithm *Guard-Monotone-Polygons*
Input: A monotone polygon $\mathbf{P}$
Output: A guard cover for $\mathbf{P}$
1 $\mathcal{G} := \emptyset$
2 **while** not all upper pockets are guarded **do**
2.1 Compute $\mathbf{sp}^U(\mathcal{G})$ and $u^U(\mathcal{G})$
2.2 Place a guard g at $u^U(\mathcal{G})$; $\mathcal{G} := \mathcal{G} \cup \{g\}$
2.3 Compute $\bigcup_{g \in \mathcal{G}} \mathbf{VP}(g)$, let $\mathbf{p}^U$ be the first upper pocket, and let L be the vertical line segment through g
2.4 Place a guard g' on L so that $u^U(\mathcal{G} \cup \{g'\})$ lies as far to the right as possible; $\mathcal{G} := \mathcal{G} \cup \{g'\}$
endwhile
3 Repeat Step 2 for the lower pockets to guard these
4 Guard the middle pockets if any such remain
return $\mathcal{G}$
End *Guard-Monotone-Polygons*

We can now give the details of our guarding algorithm.

Each iteration of Step 2 involves computing the upper spear and the upper spear tip, which we can do in linear time as we have shown. Step 2.4 can be done efficiently as follows: Let $\mathbf{VP}(\mathcal{G})$ be the part of the polygon seen so far. We begin by placing g' at the top of the line segment L and compute the upper spear with g' in the guard set. Then, we slide g' along L continuously updating the point $u^U(\mathcal{G} \cup \{g'\})$ as we go along. The structural changes of $\mathbf{sp}^U(\mathcal{G} \cup \{g'\})$ occur at certain key points on L. These are:

1. when the convex vertex of $\mathbf{VP}(\mathcal{G}) \cup \mathbf{VP}(g')$ on an edge adjacent to an upper pocket becomes incident to a vertex of the polygon boundary U.

2. when an edge of the boundary of $\mathbf{sp}^U(\mathcal{G} \cup \{g'\})$ becomes incident to two vertices

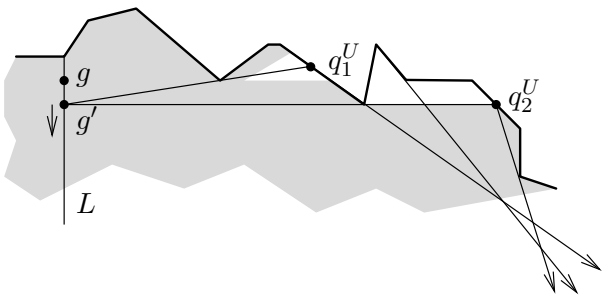


Figure 5.11: Computing the rightmost spear tip.

of the upper boundary U.

3. when three consecutive half lines issuing from pockets intersect at the same point.

The key points occur at an at most cubic number of discrete points on L. (The maximum number of possible common intersection points between three lines among n lines.) Moving g' in between the key points will make $u^U(\mathcal{G} \cup \{g'\})$ move monotonically to the right or to the left. Hence, by computing the key points, which can be done incrementally in at most linear time per step, we can find the point on L where $u^U(\mathcal{G} \cup \{g'\})$ lies as far to the right as possible; see Figure 5.11. Step 3 is performed in a similar manner and within the same complexity bounds as Step 2.

It remains to show how to guard the middle pockets, if any such remain after Step 3 of the algorithm. To show that middle pockets can indeed occur consider the monotone polygon of Figure 5.12 after Step 3 of our algorithm. The algorithm places guards $g_1, g_1', g_2, g_2', g_3$, and g_3' during Step 2. These guards also see the lower

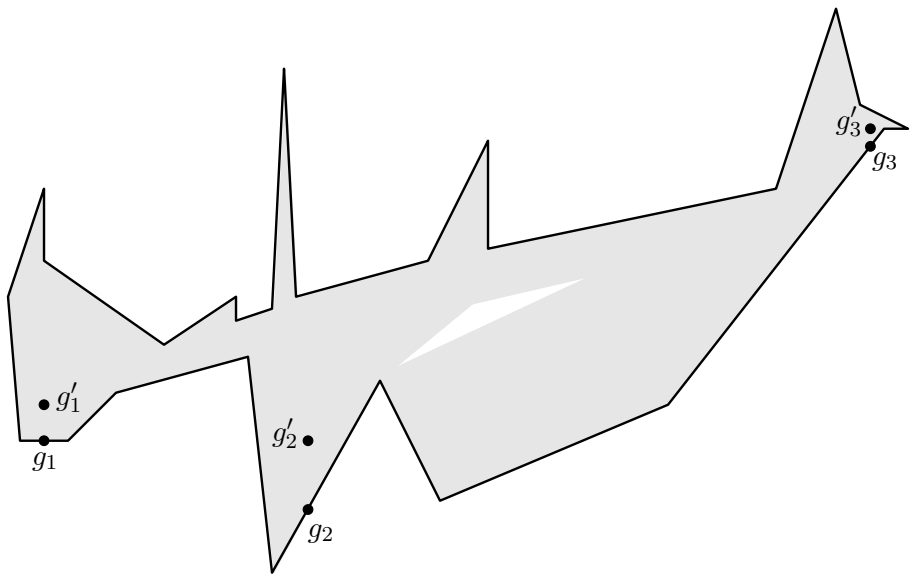


Figure 5.12: Example of a middle pocket.

boundary so no additional guards are placed during Step 3.

We prove that any middle pocket is contained in a polygon having at most two reflex vertices and can therefore be guarded by at most two guards.

Lemma 16. *Let $\mathcal{G}$ be the guard set of monotone polygon $\mathbf{P}$ obtained after Step 3 of our algorithm and let p be a point in a middle pocket of $\mathcal{G}$. If g_l and g_r are the two guards in $\mathcal{G}$ immediately to the left and right of p, respectively, then there is a connected polygonal region having at most six vertices of which at most two are reflex that contains all unseen points between g_l and g_r.*

Proof. Let L be a vertical line slightly to the right of g_l so that no unseen points lie between g_l and L. Similarly, let R be a vertical line slightly to the left of g_r so that no unseen points lie between g_r and R. Let g_r^U be the first guard placed at an upper

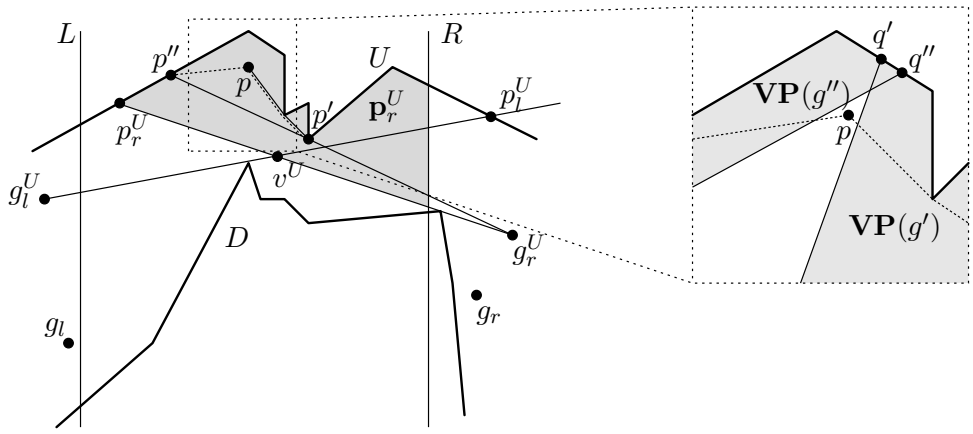


Figure 5.13: Illustrating the proof of Lemma 16.

spear tip after R by Step 2 of our algorithm. Let p_r^U be the leftmost point on U, the upper boundary, seen by g_r^U and let $\mathcal{G}_l$ be the guards of $\mathcal{G}$ that lie to the left of L.

Consider the polygonal region $\mathbf{p}_r^U$ between L and R, and above the line from g_r^U to p_r^U; see Figure 5.13. We claim that no unseen point can lie in $\mathbf{p}_r^U$. Assume to the contrary that there is a point p in $\mathbf{p}_r^U$, not on the boundary, that is not seen by g_r^U. Construct the shortest path $SP(g_r^U, p)$ inside $\mathbf{P}$ from g_r^U to p. Let the first segment of $SP(g_r^U, p)$ have the end points g_r^U and p'. Extend this segment until it hits the upper boundary at p''. The upper boundary from p'' to p' is not seen from g_r^U so by construction this region must be seen from guards in $\mathcal{G}_l$. Let $\mathcal{G}_l'$ be the guards of $\mathcal{G}_l$ that see points of $SP(p, p')$ and let $\mathcal{G}_l''$ be the guards of $\mathcal{G}_l$ that see points of $SP(p, p'')$. No guard can be in both sets since it would then see p.

Let q' be the leftmost point on the upper boundary that any guard in $\mathcal{G}_l'$ can see and let g' a guard in this set that sees q'. Similarly let q'' be the leftmost point

on the upper boundary that any guard in $\mathcal{G}_l''$ can see and let g'' be a guard in this set that sees q''. The point q' cannot lie to the right of q'', since then there are upper boundary points between p'' and p' not seen by $\mathcal{G}_l$. Hence, the line segment from p to g' must intersect the upper boundary and the line segment from p to g'' must intersect the lower boundary. This is impossible if both g' and g'' are to see boundary points between p'' and p'. Therefore all points in $\mathbf{p}_r^U$ are seen by $\mathcal{G}$.

If we let g_l^U be the guard that sees the rightmost upper boundary point p_l^U among all points in $\mathcal{G}_l$ and let $\mathbf{p}_l^U$ be the part between L and R above the line from g_l^U to p_l^U; see Figure 5.13. Assuming that there is an interior point p in $\mathbf{p}_l^U$ not seen by g_r^U we can prove a contradiction in the same way as before, and hence, all points in $\mathbf{p}_l^U$ are seen by guards from $\mathcal{G}$.

Similarly defining $\mathbf{p}_r^D$ and $\mathbf{p}_l^D$ as the regions below the corresponding lines $[g_r^D, p_r^D]$ and $[g_l^D, p_l^D]$ for the lower boundary, we can prove that all points in these regions must be seen by guards in $\mathcal{G}$.

Let $\mathbf{P}_M$ be the region of $\mathbf{P}$ between L and R. Define the region $\mathbf{M}$ to be

$$\mathbf{M} = \mathbf{P}_M \setminus (\mathbf{p}_r^U \cup \mathbf{p}_l^U \cup \mathbf{p}_r^D \cup \mathbf{p}_l^D);$$

see Figure 5.14. The region $\mathbf{M}$ contains all the unseen points in $\mathbf{P}_M$ and the boundary of $\mathbf{M}$ has at most six vertices with at most two of them being reflex. □

To implement Step 4 of our algorithm we perform a third sweep from left to right and incrementally compute the union of the visibility polygons of the guards placed to the left of the sweep line. As soon as we encounter a middle pocket to

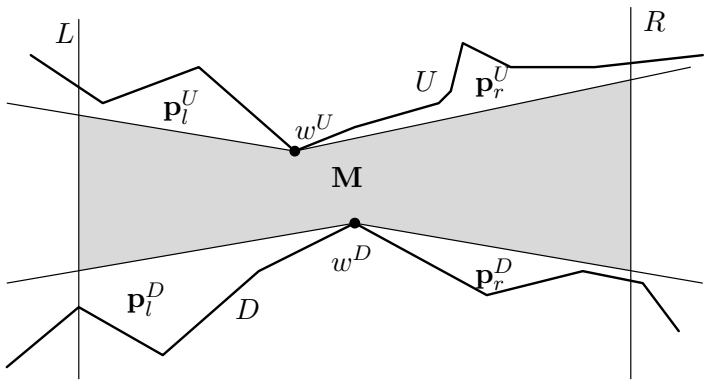


Figure 5.14: Worst case example illustrating the region **M** of Lemma 16.

the left of the sweep line, we can apply Lemma 16 and establish the region **M**, more specifically, the intersection point w^U between the lines $[g_r^U, p_r^U]$ and $[g_l^U, p_l^U]$, and the intersection point w^D between the lines $[g_r^D, p_r^D]$ and $[g_l^D, p_l^D]$ where we place guards that together will see all of **M**; see Figures 5.13 and 5.14.

Next, we establish the approximation factor of the algorithm.

Lemma 17. *Our algorithm places $O(OPT)$ guards in* **P**, *where OPT is the size of the smallest guard cover for* **P**.

Proof. To bound the total number of guards, we establish the number of guards placed in each of the Steps 2–4. To do so we compare the number of guards placed in each step with the size of a serial guard cover $\mathcal{G}^s$.

For each guard $g^s \in \mathcal{G}^s$ we denote by $q(g^s)$, the point of U straight above g^s and we denote by $r(g^s)$, the rightmost upper boundary point seen continuously along U from $q(g^s)$, i.e., g^s sees all points on U between $q(g^s)$ and $r(g^s)$ but no further.

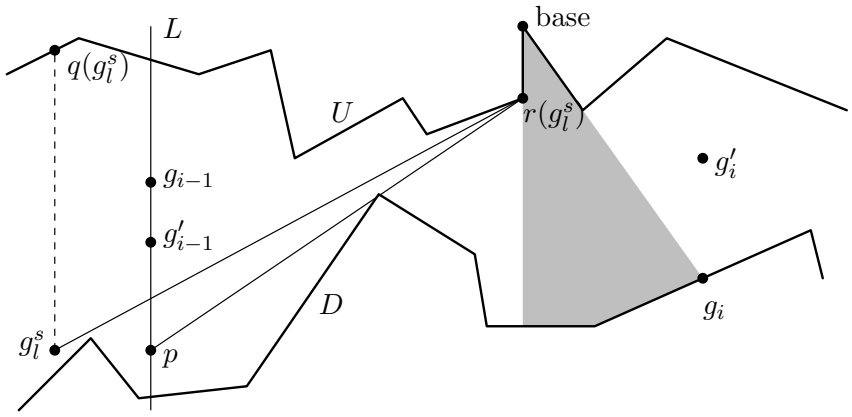


Figure 5.15: Illustrating the proof of Lemma 17.

Consider first Step 2. Let $\mathcal{G}_2$ be the guard set placed by our algorithm in Step 2. In the i^{th} iteration of the loop our algorithm places guards g_i and g'_i having the same x-coordinate.

If $\mathcal{G}^s$ has a guard g^s in the interval between g_{i-1} and g_i, we can associate the guards g_i and g'_i to g^s. We call such an association an I-association.

If $\mathcal{G}^s$ has no guard in the interval between g_{i-1} and g_i, then let $\mathcal{G}_l^s$ be the subset of guards in $\mathcal{G}^s$ that lie to the left of g_{i-1}. In addition, let $\mathcal{G}_l'^s$ be the subset of guards g^s in $\mathcal{G}_l^s$ for which $r(g^s)$ is to the right of g_{i-1}. Note that $\mathcal{G}_l'^s$ cannot be empty, as then $\mathcal{G}^s$ has no guard in the current upper spear $\mathbf{sp}^U(\mathcal{G}_l^s)$, thus contradicting that $\mathcal{G}^s$ is serial; see Figure 5.15.

Now, let $g_l^s \in \mathcal{G}_l'^s$ be the guard for which $r(g_l^s)$ is leftmost among all the guards in $\mathcal{G}_l'^s$. We argue that $r(g_l^s)$ lies on U between g_{i-1} and g_i. Assume for the contradiction that it does not. It then lies strictly to the right of g_i. Let L be the

vertical line through g_{i-1} and g'_{i-1} and let p be the lowest point of L that $r(g_l^s)$ sees; see Figure 5.15. The point p sees all of U from $q(g_l^s)$ to $r(g_l^s)$ since any point in this interval is seen by g_l^s. In this case, by placing g'_{i-1} on p, our algorithm would have been able to move g_i further to the right, a contradiction, since in Step 2.4 of our algorithm, we place g'_{i-1} on L so that g_i is as far to the right as possible. Thus, we know that we can associate g_i and g'_i to g_l^s and we call this type of association, an X-association. No other guards g_j and g'_j, $j \neq i$, placed by our algorithm in Step 2 can be X-associated to g_l^s, since $r(g_l^s)$ is not on U between g_{j-1} and g_j

Hence, any guard g^s in $\mathcal{G}^s$ is associated to at most four guards from $\mathcal{G}_2$, two of which are I-associated to g^s and two of which are X-associated. We have that

$$|\mathcal{G}_2| \leq 4|\mathcal{G}^s| \in O(OPT),$$

since we can choose $\mathcal{G}^s$ to be a smallest serial guard cover which by Lemma 15 is of size $O(OPT)$.

Similarly, let $\mathcal{G}_3$ be the guard set placed by our algorithm in Step 3. We can prove that

$$|\mathcal{G}_3| \leq 4|\mathcal{G}^s| \in O(OPT)$$

in the same way as above.

Finally, denote by $\mathcal{G}_4$, the guards placed by our algorithm in Step 4. By Lemma 16 we can simply associate the two guards placed at w^U and w^D to guard the middle pockets between two guards g_l and g_r, placed in the previous steps, with one

of them, say g_l. Hence,

$$|\mathcal{G}_4| \leq 2(|\mathcal{G}_2| + |\mathcal{G}_3|) \in O(OPT);$$

see also Addario-Berry *et al* [1]. □

We have proved the following theorem.

Theorem 18. *The algorithm Guard-Monotone-Polygons computes a guard cover for a monotone polygon* $\mathbf{P}$ *of size* $O(OPT)$ *in polynomial time, where* OPT *is the size of the smallest guard cover for* $\mathbf{P}$.

CHAPTER 6
CONCLUSION

6.1 Summary

This document has given a summary of my work in the areas of guarding terrains and guarding art galleries. My contributions have included a 4-approximation for the terrain guarding problem. Using some the observations from the 4-approximation and developments made elsewhere in other geometric set covering problems, the $(1+\epsilon)$-approximation was discovered. This document also provides an NP-hardness proof for the terrain guarding problem. This coupled with the $(1+\epsilon)$-approximation settles the computational complexity problem for this problem. A minor question that remains regarding the complexity of terrain guarding is whether or not it is fixed-parameter tractable.

This document also highlights an NP-hardness proof for vertex guarding monotone polygons. There is also a $O(1)$-approximation result for point guarding a monotone polygon.

The 4-approximation shown in Chapter 3 was shown by Elbassioni *et al.* in [17]. The NP-hardness result from Chapter 2 is shown in [31]. The $(1+\epsilon)$-approximation from Chapter 4 is shown in [23]. Both of the art gallery results from Chapter 5 are shown in [32].

6.2 Future Work

Future work may involve working on different and more difficult geometric set cover problems. Problems might include looking into better approximation algorithms for the original art gallery problem. Improving the approximation algorithm for monotone polygons and extending to vertex guarding are areas of interest. I believe some of the observations made with the terrain algorithms could translate into better algorithms for guarding art galleries. Little is known about point guarding a simple polygon and researching this problem is also of interest. Another potential problem involves the terrain guarding problem with limited visibility. The terrain guarding problem studied in this thesis allows guards to see forever unless blocked by the terrain. The problem changes with the added restriction of limited visibility.

I am also interested in developing some heuristics for each of these problems. I believe some of the characteristics of terrain guarding that have been uncovered would lead to fast heuristics that could outperform the many combinatorial set cover heuristics already available. Some heuristics might lead to improved approximation algorithms for the art gallery problems if new characteristics are discovered.

REFERENCES

[1] Louigi Addario-Berry, Omid Amini, Jean-Sébastien Sereni, and Stéphan Thomassé. Guarding art galleries: The extra cost for sculptures is linear. In *SWAT '08: Proceedings of the 11th Scandinavian workshop on Algorithm Theory*, pages 41–52, Berlin, Heidelberg, 2008. Springer-Verlag.

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

[3] Noga Alon, Dana Moshkovitz, and Shmuel Safra. Algorithmic construction of sets for k-restrictions. *ACM Trans. Algorithms*, 2(2):153–177, 2006.

[4] Boaz Ben-Moshe, Matthew J. Katz, and Joseph S. B. Mitchell. A constant-factor approximation algorithm for optimal 1.5d terrain guarding. *SIAM Journal on Computing*, 36(6):1631–1647, 2007.

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

[6] H Bronnimann and M Goodrich. Almost optimal set covers in finite vc-dimension. *Discrete Comput. Geom*, pages 14–463, 1995.

[7] Timothy M. Chan and Sariel Har-Peled. Approximation algorithms for maximum independent set of pseudo-disks. In *SCG '09: Proceedings of the 25th annual symposium on Computational geometry*, pages 333–340, New York, NY, USA, 2009. ACM.

[8] 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 133–138, 1995.

[9] V. Chvatal. A combinatorial theorem in plane geometry. *Journal of Combinatorial Theory Series B*, 18:39–41, 1975.

[10] V. Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4(3):233–235, 1979.

[11] Kenneth L. Clarkson and Kasturi R. Varadarajan. Improved approximation algorithms for geometric set cover. In *Proceedings of the 20th Symposium on Computational Geometry*, pages 135–141, 2005.

[12] Erik D. Demaine and Joseph O'Rourke. Open problems: Open problems from cccg 2005. In *Proceedings of the 18th Canadian Conference on Computational Geometry*, pages 75–80, 2006.

[13] Ajay Deshpande, Taejung Kim, Erik D. Demaine, and Sanjay E. Sarma. A pseudopolynomial time o(log n)-approximation algorithm for art gallery problems. In Frank K. H. A. Dehne, Jörg-Rüdiger Sack, and Norbert Zeh, editors, *WADS*, volume 4619 of *Lecture Notes in Computer Science*, pages 163–174. Springer, 2007.

[14] Alon Efrat and Sariel Har-Peled. Guarding galleries and terrains. *Inf. Process. Lett.*, 100(6):238–245, 2006.

[15] S. Eidenbenz. *Inapproximability of Visibility Problems on Polygons and Terrains.* PhD thesis, 2000.

[16] Stephan Eidenbenz. Inapproximability results for guarding polygons without holes. In *Lecture Notes in Computer Science*, pages 427–436. Springer, 1998.

[17] Khaled Elbassioni, Erik Krohn, Domagoj Matijevic, Julian Mestre, and Domagoj Severdija. Improved approximations for guarding 1.5-dimensional terrains. In Susanne Albers and Jean-Yves Marion, editors, *26th International Symposium on Theoretical Aspects of Computer Science (STACS 2009)*, Dagstuhl, Germany, 2009. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany.

[18] Khaled Elbassioni, Domagoj Matijevic, Julian Mestre, and Domagoj Severdija. Improved approximations for guarding 1.5-dimensional terrains. CoRR, abs/0809.0159v1, 2008.

[19] Thomas Erlebach, Klaus Jansen, and Eike Seidel. Polynomial-time approximation schemes for geometric intersection graphs. *SIAM Journal on Computing*, 34(6):1302–1323, 2005.

[20] S. Fisk. A short proof of chvatal's watchman theorem. *Journal of Combinatorial Theory Series B*, 24:374+, 1978.

[21] Greg N. Frederickson. Fast algorithms for shortest paths in planar graphs, with applications. *SIAM J. Comput.*, 16(6):1004–1022, 1987.

[22] S. Ghosh. Approximation algorithms for art gallery problems. In *Proc. Canadian Information Processing Society Congress*, 1987.

[23] Matt Gibson, Gaurav Kanade, Erik Krohn, and Kasturi R. Varadarajan. An approximation scheme for terrain guarding. In Irit Dinur, Klaus Jansen, Joseph Naor, and José D. P. Rolim, editors, *APPROX-RANDOM*, volume 5687 of *Lecture Notes in Computer Science*, pages 140–148. Springer, 2009.

[24] Frank Hoffmann, Michael Kaufmann, and Klaus Kriegel. The art gallery theorem for polygons with holes. In *SFCS '91: Proceedings of the 32nd annual symposium on Foundations of computer science*, pages 39–48, Washington, DC, USA, 1991. IEEE Computer Society.

[25] B. Joe and R. B. Simpson. Corrections to lee's visibility polygon algorithm. *BIT*, 27(4):458–473, 1987.

[26] David S. Johnson. Approximation algorithms for combinatorial problems. In *STOC '73: Proceedings of the fifth annual ACM symposium on Theory of computing*, pages 38–49, New York, NY, USA, 1973. ACM.

[27] R.M. Karp. Reducibility among combinatorial problems. In R.E. Miller and J.W. Thatcher, editors, *COMPLEXITY OF COMPUTER COMPUTATIONS*, New York, 1972. Plenum Press.

[28] James King. Errata on "A 4-Approximation Algorithm for Guarding 1.5-Dimensional Terrains". http://www.cs.mcgill.ca/ jking/.

[29] James King. A 4-approximation algorithm for guarding 1.5-dimensional terrains. In *Proceedings of the 13th Latin American Symposium on Theoretical Informatics*, pages 629–640, 2006.

[30] James King and Erik Krohn. The complexity of guarding terrains. In *SODA*, 2010. To Appear.

[31] Erik Krohn. The complexity of guarding terrains, *Manuscript*, 2009.

[32] Erik Krohn and Bengt J. Nilsson. Approximate guarding of monotone and rectilinear polygons, 2009. Submitted to *Algorithmica*.

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

[34] D.T. Lee. Visibility of a simple polygon. *Computer Vision, Graphics, and Image Processing*, pages 22:207–221, 1983.

[35] David Lichtenstein. Planar formulae and their uses. *SIAM Journal on Computing*, 11(2):329–343, 1982.

[36] L. Lovász. On the ratio of optimal integral and fractional covers. In *Discrete Mathematics*, number 13, pages 383–390, 1975.

[37] C Lund and M Yannakakis. On the hardness of approximating minimization problems. In *Proceedings of the Twenty Fifth Annual Symposium on the Theory of Computing, ACM*, 1993.

[38] Nabil Hassan Mustafa and Saurabh Ray. Ptas for geometric hitting set problems via local search. In *SCG '09: Proceedings of the 25th annual symposium on Computational geometry*, pages 17–22, New York, NY, USA, 2009. ACM.

[39] B.J. Nilsson. *Guarding Art Galleries — Methods for Mobile Guards*. PhD thesis, 1995.

[40] R Raz and S Safra. A sub-constant error-probability low-degree-test and a sub-constant error-probability. In *PCP Characterization of NP.InProceedings of the 29th Annual ACM Symposium on Theory of Computing.ACM*, pages 475–484. Press, 1997.

[41] Micha Sharir and Pankaj K. Agarwal. Davenport-schinzel sequences and their geometric applications. *Cambridge University Press*, 1995.