A 4-Approximation Algorithm for Guarding 1.5-Dimensional Terrains

James King

School of Computer Science,
McGill University,
Montreal, H3A 2A7, Canada

Abstract. In the 1.5-dimensional terrain guarding problem we are given as input an x-monotone chain (the terrain) and asked for the minimum set of guards (points on the terrain) such that every point on the terrain is seen by at least one guard. It has recently been shown that the 1.5-dimensional terrain guarding problem is approximable to within a constant factor [3, 7], though no attempt has been made to minimize the approximation factor. We give a 4-approximation algorithm for the 1.5D terrain guarding problem that runs in quadratic time. Our algorithm is faster, simpler, and has a better worst-case approximation factor than previous algorithms.

1 Introduction

1.1 Problem Statement

In the 1.5-dimensional terrain guarding problem we are given as input a terrain T that is an x-monotone polygonal chain. An x-monotone polygonal chain is a polygonal chain that intersects any vertical line at most once. It can be thought of as an array of n vertices in 2-dimensional space sorted in ascending order of x-coordinate, where edges 'connect the dots' from left to right. Note that the x-monotonicity requires x-coordinates to be distinct.

We say that a point on the terrain sees another point on the terrain if there is a line of sight between them, *i.e.* the line segment connecting them is never strictly below T. A guard is simply a point on the terrain that we add to a 'guarding set'. Given a terrain T, we are asked for the smallest possible guarding set, *i.e.* the smallest set G of points on T such that every point on T is seen by some point in G.

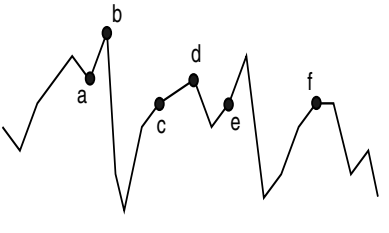

Fig. 1. An example of a 1.5D terrain. d can see b, c, and e but not a or f.

It is natural to consider two different versions of the terrain guarding problem: the discrete version and the continuous version. In the discrete version guards must be at vertices and only the vertices of the terrain need to be guarded. In the

J.R. Correa, A. Hevia, and M. Kiwi (Eds.): LATIN 2006, LNCS 3887, pp. 629–640, 2006.

continuous version guards may be anywhere on the terrain and every point on the terrain must be guarded. The discrete version is simpler but the continuous version is more natural to consider in a geometric context. For the rest of this paper we will use TG to denote the discrete version of the problem and TG-C to denote the continuous version.

Every instance of TG is an instance of Set Cover, but we know that Set Cover is NP-complete (see, *e.g.*, [13]) and no sub-logarithmic approximation factor can be obtained unless NP $\subseteq$ DTIME($n^{\log \log n}$) [12]. In general it is not particularly difficult to modify a TG algorithm to solve instances of TG-C, though this often involves some polynomial increase in time complexity.

1.2 Related Work

The 1.5D terrain guarding problem is very similar to the *art gallery problem* in which one must guard the interior of a simple polygon. The art gallery problem and its variants are well studied [1, 6, 11, 16, 17].

It is unknown whether or not TG is NP-hard. In 1995 Chen *et al.* [5] proposed an NP-hardness proof obtainable via a modification of Lee and Lin's proof that the art gallery problem is NP-complete [17]. However, the proof, whose details were omitted, was never completed successfully. Since then, attempts to find a polynomial-time algorithm for TG and attempts to prove that it is NP-hard have both been unsuccessful.

The first constant-factor approximation algorithm for the 1.5D terrain guarding problem was given by Ben-Moshe *et al.* [3]. Their algorithm works by first placing guards to divide the terrain into independent subterrains. Each subterrain has the property that it does not require internal guards, *i.e.* every unguarded vertex can be seen from outside the subterrain. For each such subterrain that is not completely guarded they then proceed with steps that either reduce the subterrain or split it into multiple independent subterrains. They made no attempt to minimize their algorithm's approximation factor; as such it is very large (at least 48). It could be brought down possibly as low as 6 with some minor modifications and careful accounting, but due to the inevitable cost incurred by repeatedly dividing the terrain it does not seem that it could be brought any lower than 6. Their algorithm runs in $O(n^2)$ time for the discrete version. They also provide a reduction from TG-C to TG that allows them to solve TG-C in $O(n^4)$ time, though the approximation factor can double in this case.

Another constant-factor approximation algorithm is given by Clarkson and Varadarajan [7]. Consider a partition of a 1.5D terrain into maximal intervals such that, for any two points p and p' in a given interval, the leftmost point that sees p and the leftmost point that sees p' are the same. If we label each interval with the leftmost point that sees it and read the labels from leftmost interval to rightmost interval, Clarkson and Varadarajan note that we end up with an $(n, 2)$ Davenport-Schinzel sequence [18]. Such a sequence must have length at most $2n$. This characterization of the lack of complexity in 1.5D terrains allows them to efficiently find appropriate ϵ-nets [14] for instances of TG. They then apply the Set Cover method of Brönnimann and Goodrich [4] to solve the problem using

these ϵ-nets. The end result is a constant-factor approximation algorithm that runs in polynomial time. Using efficient derandomization their algorithm could probably be made to run deterministically in $O(n^2 \log n)$ time.

The 1.5D terrain guarding problem becomes easy if, instead of being placed on the terrain, all guards 'float' above the terrain at a fixed altitude that is above the highest vertex. Eidenbenz [8] gives a linear-time algorithm for finding an optimal set of guards in this case. The problem also becomes easy if guards can only look rightwards. Chen *et al.* [5] give a linear-time algorithm for this case.

A 2.5D terrain is a polyhedral surface that intersects every vertical line at most once and whose projection onto the x, y-plane is a simple polygon with no holes. The 2.5D Terrain Guarding Problem is therefore a natural extension of the 1.5D problem to the next dimension. Finding a minimum number of guards for a 2.5D terrain is NP-complete and Eidenbenz shows that it cannot be approximated within a sub-logarithmic factor unless NP $\subseteq$ DTIME($n^{\log \log n}$) [9]. Eidenbenz *et al.* show that the problem is also NP-complete and equally inapproximable when guards 'float' at a given altitude that is higher than the highest point in the terrain [10] (recall that this can be solved in linear time for 1.5D terrains).

1.3 Motivation

Naturally, the motivation for 1.5D terrain guarding comes from guarding or covering terrain. The 1.5D case appears, for example, when guarding or covering a road, perhaps with security cameras or street lights. The 2.5D case has more powerful applications, most notably for providing a wireless communication network that covers a given region. Its proven intractability and inapproximability, however, motivate us to look towards the 1.5D case for insight. The 1.5D case is also applicable, for example, if we only need to cover the path between two points on a polyhedral terrain. It has been pointed out [3] that the 1.5D terrain guarding problem can be utilized in heuristic methods for the 2.5D case.

The recent results of Ben-Moshe *et al.* [3] and Clarkson and Varadarajan [7] showed that constant-factor approximation algorithms exist for TG. Unfortunately they do not provide a small constant guaranteed approximation factor. Efforts to design an exact polynomial-time algorithm for TG have been unsuccessful and it is very possible that no such algorithm exists. If TG is NP-hard and P $\neq$ NP, then the best algorithm running in polynomial time will be the approximation algorithm with the lowest approximation factor. For this reason there is significant motivation to minimize the approximation factor.

The greedy algorithm for Set Cover, which achieves the optimal approximation factor of $O(\log n)$, repeatedly picks the set that contains the most uncovered elements. Similarly, the natural greedy algorithm for terrain guarding repeatedly picks the guard that sees the most unguarded vertices. There are terrains for which this method achieves a logarithmic approximation factor (such a terrain, provided by Ben-Moshe [2], is described in [15]). There are other natural greedy-like algorithms that one might consider. For example, one could repeatedly pick the guard that maximizes the leftmost unguarded vertex or the lowest unguarded vertex. Terrains exist for these algorithms that prove they do

not achieve constant approximation factors. The apparent absence of simple algorithms that achieve constant approximation factors motivates us to consider more sophisticated techniques.

1.4 Our Contribution

Our result is a 4-approximation algorithm for the 1.5D terrain guarding problem. It runs in $O(n^2)$ time for TG and can be modified slightly to run in $O(n^2)$ time for TG-C.

1.5 Organization

The rest of the paper is organized as follows. In Section 2 we introduce notation and some small but fundamental lemmas. In Section 3 we give our 4-approximation algorithm for TG; the modifications required for TG-C are explained in Section 3.6. In Section 4 we discuss open problems and suggest directions for future work regarding 1.5D terrain guarding.

2 Preliminaries

2.1 Terminology and Notation

An instance of the 1.5D terrain guarding problem is simply an x-monotone chain T. This chain is a sequence of vertices $v_1, \ldots, v_n$ and edges $e_i = (v_i, v_{i+1})$, $i = 1 \ldots n-1$ such that the x-coordinate of v_i is smaller than that of v_j if $i < j$. Given two points p and q on T (not necessarily vertices of T), we say that $p < q$ if the x-coordinate of p is smaller than that of q.

For a point p we use $L(p)$ to denote the leftmost point that sees p and $R(p)$ to denote the rightmost point that sees p. It is not difficult to see that $L(p)$ and $R(p)$ will always be vertices, whether p is a vertex or not. We use $T_L(p)$ to denote the terrain restricted to the interval $[v_1, p]$ and use $T_R(p)$ to denote the terrain restricted to $[p, v_n]$. $CH(T)$ is the (upper) convex hull of T. We use $CH_L(p)$ to denote the convex hull of $T_L(p)$ and use $CH_R(p)$ for that of $T_R(p)$. If a point p sees every unguarded point that another point q sees we say that p *dominates* q. We can also say that a set S dominates a point p if every unguarded point seen by p is also seen by some vertex in S. We say that p dominates q with respect to a certain region of T if p sees every unguarded point in that region that q sees.

We consider a minimum guarding set G_{OPT} for the terrain T. We assume there is some mapping g of points of T to guards in G_{OPT} such that, for a point p, $g(p)$ is a guard in G_{OPT} that sees p. We say that $g(p)$ is the guard *responsible for* p. g is surjective but never injective (since $|G_{OPT}| < n$); we use it to simplify the explanation of our accounting scheme.

2.2 Elementary Lemmas

We will now state and prove several small but fundamental lemmas that we will use in the rest of the paper. These lemmas and corollaries can be used with left

and right interchanged; this is stated explicitly for Corollary 1 as an example but is not stated for the others. Also note that these lemmas involve points on the terrain that need not necessarily be vertices.

Lemma 1 (Order Claim [3, 5]). *For points a, b, c, d such that $a \leq b < c \leq d$, if a sees c and b sees d then a sees d.*

Proof. This becomes quite clear with the help of a diagram (see Figure 2). It is trivially true if $a = b$ or $c = d$; otherwise we know that $a < b < c < d$. In this case b cannot be above $\overline{ac}$ and c cannot be above $\overline{bd}$ (otherwise the fact that a sees c and b sees d would be violated). This means that the two line segments must cross; we call their intersection point p. Considering the triangle formed by a, p, and d, we note that no point on the terrain can be above the lower hull and $\overline{ad}$ is the upper hull. Therefore no point on the terrain can be above $\overline{ad}$.

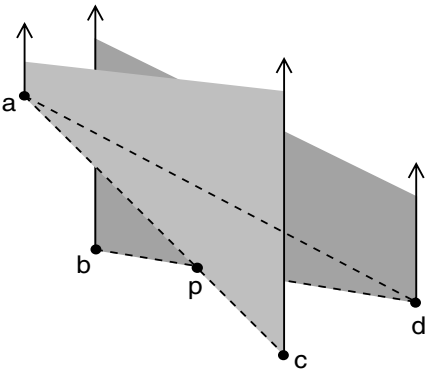

Fig. 2. The shaded areas are terrain free and their union contains $\overline{ad}$

Corollary 1. *For points u, v, w with $u \leq v < w$, if u and v can both be seen from $T_R(w)$ then $R(v) \leq R(u)$.*

Corollary 1 (Symmetric Version). *For points u, v, w with $u < v \leq w$, if v and w can both be seen from $T_L(u)$ then $L(w) \leq L(v)$.*

Lemma 2. *For an interval $[a, b]$ where a sees b, any guard in (a, b) is dominated with regard to $T_R(b)$ by a.*

Proof. Let p be a guard in (a, b) and let q be some point in $T_R(b)$ seen by p. If $q = b$ we know that a sees q. Otherwise the order claim, applied to a, p, b, q, states that a sees q.

Corollary 2. *For points p and q such that $p < q < R(p)$, we know that $R(q) \leq R(p)$.*

Lemma 3 (Lip Lemma). *For an interval $[a, b]$ where a sees b, if there are no unguarded points in (a, b) then $\{a, b\}$ dominates any guard in $[a, b]$.*

Proof. This follows from Lemma 2 since a and b see each other.

Lemma 4. *For a point q, any guard p in $T_L(q)$ is dominated with regard to $T_R(q)$ by a guard in $CH_L(q)$. In particular, p is dominated by the rightmost point in $T_L(p) \cap CH_L(q)$.*

Proof. Let u be the rightmost point in $T_L(p) \cap CH_L(q)$. If p is on $CH_L(q)$ then $p = q$ and the lemma clearly holds. Otherwise let w be the first point on $CH_L(q)$ to the right of u. Now u sees w, so u dominates p with regard to $T_R(q) \subseteq T_R(w)$ by Lemma 2.

Corollary 3. *For points p and q, if $L(q) \leq p \leq q$ then $L(q)$ is on $CH_L(p)$.*

3 The Algorithm

Our algorithm works by repeatedly finding an unguarded point u and a set S of up to 4 points such that S must dominate $g(u)$. By doing so, we achieve an approximation factor of 4, since we charge at most 4 guards to each guard in G_{OPT}. Our algorithm does not require any knowledge of previously placed guards other than which points are unguarded. The rest of this section basically deals with how to find an appropriate unguarded point. We first explain the algorithm as applied to TG, and in Section 3.6 we explain the minor modifications required for TG-C.

3.1 Introduction to GuardRight

Consider an unguarded vertex p not on $CH(T)$ along with a vertex c that can see every unguarded vertex in the range $[L(R(p)), p)$. c is like a good potential guard that lets us focus on the unguarded points in $[p, R(p)]$. Note that if we place a guard at c, no unguarded vertex in $[L(R(p)), R(p))$ can be seen from outside $[L(R(p)), R(p)]$. For this reason we say that the interval $[L(R(p)), R(p)]$ is *pseudo-independent*. Our algorithm repeatedly finds appropriate (p, c) pairs or advances trivially if such points are not available. If there is only one unguarded vertex s, we place a guard there that dominates $g(s)$. Otherwise consider the two leftmost unguarded vertices s and t with $s < t$. If $s \in CH(T)$ and $t \in CH(T)$ we just place a guard at $R(s)$ that must dominate $g(s)$. If $s \notin CH(T)$ then $p \leftarrow s$ and $c \leftarrow s$. If $s \in CH(T)$ but $t \notin CH(T)$ then $p \leftarrow t$ and $c \leftarrow s$.

If an appropriate (p, c) pair is found, our algorithm calls a recursive subroutine GuardRight(p, c). GuardRight will either find an unguarded vertex $u \in [p, R(p))$ for which $g(u)$ can be dominated by 4 guards or will find a pseudo-independent subinterval of $[p, R(p)]$, *i.e.* a pseudo-independent 'pocket' of the terrain that it can recurse into with new parameters p' and c'. At this point we introduce some new terminology and notation that depends on the parameters of GuardRight. It should be emphasized that this notation applies only to a particular call to GuardRight. We say that a *left vertex* is a vertex in $CH([L(R(p)), p]) - \{p\}$. A *right vertex* is a vertex in $[p, R(p)]$. An *exposed vertex* is an unguarded vertex in $[p, R(p))$ that can be seen by a left vertex. A *sheltered vertex* is an unguarded vertex in $[p, R(p))$ that cannot be seen by a left vertex. For an exposed vertex v we provide additional notation: $R'(v)$ is the rightmost left vertex that sees v and $L'(v)$ is the leftmost right vertex that sees v. $R'(v)$ and $L'(v)$ are undefined unless v is an exposed vertex.

Lemma 5. **(a)** *If v is a sheltered vertex then $L(R(p)) \leq p \leq L(v) \leq v \leq R(v) \leq R(p)$.* **(b)** *If v is an exposed vertex then $L(R(p)) \leq L(v) \leq R'(v) < p \leq L'(v) \leq v \leq R(v) \leq R(p)$.*

Proof. **(a)** $L(R(p)) \leq p$ since $R(p)$ sees p. $p \leq L(v)$ otherwise v would be an exposed vertex. $L(v) \leq v$ by definition. $R(v) \leq R(p)$ by Corollary 2. **(b)** $L(R(p)) \leq L(v)$ by Corollary 1 if $p < v$ and by the Order Claim otherwise. $L(v) \leq R'(v) < p \leq L'(v) \leq v$ by definition. $R(v) \leq R(p)$ by Corollary 2.

Lemma 6. *For an exposed vertex v, $L'(v)$ sees $R(v)$.*

Proof. If $v = p$ this is clearly true since $p = L'(p)$. Otherwise, it is easy to see that this is true as long as v is not above the line passing through $L'(v)$ and $R(v)$. $L'(v)$ cannot be below the line passing through p and v otherwise v would be seen by a vertex in $[p, L'(v))$ which contradicts the definition of $L'(v)$. Similarly, $R(v)$ cannot be below the line passing through v and $R(p)$. It should now be clear that v is not above the line passing through $L'(v)$ and $R(v)$ (see Figure 3). The rest follows trivially.

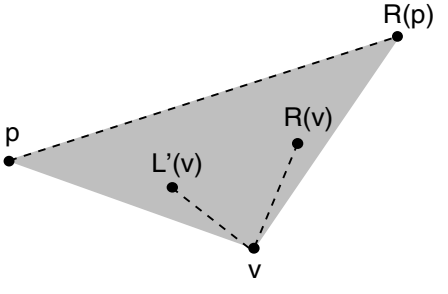

Fig. 3. $L'(v)$ and $R(v)$ must be in the shaded region

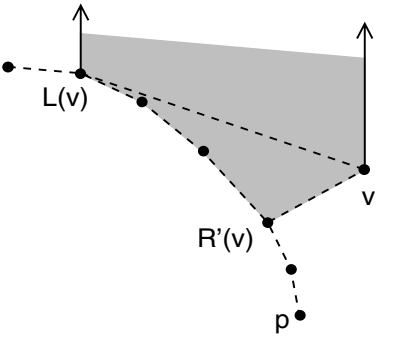

Fig. 4. The shaded region is terrain free, so every left vertex in $[L(v), R'(v)]$ must see v

Lemma 7. *For an exposed vertex v the set of left vertices that see v is contiguous,* i.e. *every left vertex in $[L(v), R'(v)]$ sees v.*

Proof. Consider $CH([L(v), R'(v)])$. This is a subset of $CH([L(R(p)), p]) - \{p\}$ since $L(v)$ and $R'(v)$ are both on $CH([L(R(p)), p])$. So we can see that $CH([L(v), R'(v)])$ is a set of left vertices and we know that no left vertex not in the set can see v. Now we will show that if w is a vertex in the set, $w \neq L(v)$, w sees v, and w' is the first vertex in the set to the left of w, then w' also sees v. w' must be above the line passing through v and w since $w \neq L(v)$. Since w' and w are consecutive points on the convex hull, w sees w'. Now we can see that $\overline{w'w}$ and $\overline{wv}$ are line segments that do not interfere with the terrain, so $\overline{w'v}$ cannot interfere with the terrain since it is above $\overline{w'w}$ and $\overline{wv}$. Therefore w' sees v. It is easy to extend this into an induction proof for the lemma. See Figure 4.

Lemma 8. *For any vertex v in $[L(R(p)), p)$, if w is the rightmost left vertex in $T_L(v)$ then $\{c, w\}$ dominates v.*

Proof. By Lemma 4 we know that w dominates v with regard to $T_R(p)$. If $w \neq v$ then v cannot see any vertex to the left of w so w dominates v with regard to $T_L(w)$. c can see every unguarded vertex in $[L(R(p)), p)$. Since $L(R(p)) \leq w$, we can see that $T_L(w) \cup [L(R(p)), p) \cup T_R(p) = T$. Therefore $\{c, w\}$ dominates v.

3.2 Finding a Good Left Vertex

Lemma 8 tells us that, as long as we place a guard at c when we place other guards, we needn't place any guard in $[L(R(p)), p)$ unless it is on a left vertex. The first thing we note is that there must be at least one exposed vertex in $[p, R(p))$, namely p. There may or may not be a sheltered vertex in $[p, R(p))$. We define b as the leftmost left vertex such that some exposed vertex v is seen by b but not by any left vertex to the right of b. In other words, b is the leftmost $R'(v)$ over all exposed vertices v. We define d as the leftmost exposed vertex for which $R'(d) = b$.

Lemma 9. *Every exposed vertex in $(L'(d), R(p))$ is seen by $L(d)$.*

Proof. If $d = p$ then the proof follows easily from the symmetric version of Corollary 1, so we will assume this is not the case. First we will prove that there are no exposed vertices in $(L'(d), d)$. Assume for the sake of contradiction that there is an exposed vertex v in $(L'(d), d)$. We can apply the order claim to $R'(v), L'(d), v, d$ to see that $R'(v)$ sees d. This tells us that $R'(v) \leq R'(d)$, which violates the definition of d, so there cannot be any such vertex v. Now we show that $L(d)$ sees every exposed vertex in $(d, R(p))$. Let w be an exposed vertex in $(d, R(p))$. We have $L(w) < d < w$ so by the symmetric version of Corollary 1 we know that $L(w) \leq L(d)$. By the definition of d we know that $R'(d) \leq R'(w)$. Therefore $L(d) \in [L(w), R'(w)]$, so by Lemma 7 we know that $L(d)$ sees w.

Lemma 10. *Any guard in $[L(R(p)), p)$ that sees d is dominated by $\{L(d), c\}$.*

Proof. Let v be a guard in $[L(R(p)), p)$ that sees d. Since c sees every unguarded vertex in $[L(R(p)), p)$ it suffices to prove that $L(d)$ dominates v with regard to $T_R(p)$. $L(d) \leq v$, so by Lemma 2 $L(d)$ dominates v with regard to $T_R(d)$. Now we show that no left vertex that sees d can see any exposed vertex to the left of d. It follows from the definition of $b = R'(d)$ and from Lemma 7 that any exposed vertex seen from the left of $R'(d)$ must be seen by $R'(d)$. However, d is the leftmost exposed vertex seen by $R'(d)$, so no exposed vertex to the left of d can be seen by $R'(d)$. In other words, no exposed vertex to the left of d can be seen by a left vertex that sees d. This, along with Lemma 4, tells us that v cannot see any unguarded vertices in $[p, d)$. Since v cannot see any sheltered vertices at all, this means that $L(d)$ dominates v with regard to $T_R(p)$. v cannot see anything left of $L(R(p))$ except possibly if $v = L(d)$, so $\{L(d), c\}$ dominates v over the entire terrain.

Recall that, while searching for a suitable vertex u for which we can dominate $g(u)$ with 4 guards, either we find one right away or we find some pseudo-independent pocket (*i.e.* a subinterval of $[p, R(p))$) that we can recurse upon.

3.3 The Terminal Case

We first consider the case where there are no sheltered vertices in $(L'(d), R(d))$. We place guards at $\{c, L(d), L'(d), R(d)\}$ and claim that these guards dominate any guard that sees d. Lemma 9 tells us that every exposed vertex in $(L'(d), R(d))$ is seen by $L(d)$, and since there are no sheltered vertices in $(L'(d), R(d))$ there are no longer any unguarded vertices in $(L'(d), R(d))$. $L'(d)$ sees $R(d)$ by Lemma 6. Therefore, by Corollary 3, any guard in $[L'(d), R(d)]$ is dominated by $\{L(d), L'(d), R(d), c\}$. By Lemma 10 any guard in $[L(R(p)), p]$ that sees d is dominated by $\{L(d), L'(d), R(d), c\}$. Any guard that sees d must either be in $[L(R(p)), p]$ or in $[L'(d), R(d)]$, so $\{L(d), L'(d), R(d), c\}$ dominates any guard that can see d.

3.4 The Recursive Case

If there are sheltered vertices in $(L'(d), R(d))$ our job is slightly more complicated and requires recursion (this is where we find our pseudo-independent pocket). We require another subroutine, GuardLeft, that is simply a mirror image of GuardRight. For a call to GuardLeft(p', c') the condition that c' must satisfy is flipped horizontally: every unguarded vertex in $(p', R(L(p'))]$ must be seen by c'. Also, p' cannot be on $CH(T)$.

Let q be the rightmost sheltered vertex (note that q is not necessarily in the interval $(L'(d), R(d))$, but it must be in $(L'(d), R(p))$). We will show that the preconditions are satisfied if we call GuardLeft$(q, L(d))$. By Corollary 2 $R(L(q)) \leq R(p)$ and by the definition of q any unguarded vertex in $(q, R(p))$ is an exposed vertex. Therefore by Lemma 9 every unguarded vertex in $(q, R(L(q)))$ is seen by $L(d)$. If $R(L(q)) < R(p)$ then either $R(L(q))$ is already guarded or it is an exposed vertex and is seen by $L(d)$. If $R(L(q)) = R(p)$ then $L(d)$ sees $R(L(q))$ since every vertex in $CH([L(R(p)), p])$ sees $R(p)$. Therefore every unguarded vertex in $(q, R(L(q))]$ is seen by $L(d)$. We know q is unguarded and $q \in (p, R(p))$ (and is therefore not on $CH(T)$), so the preconditions are satisfied.

In this way we can do a sort of recursive zig-zagging where each call to GuardRight will spawn a call to GuardLeft and each call to GuardLeft will spawn a call to GuardRight. It is not difficult to see that eventually, after at most a linear number of these zig-zagging steps, we will arrive at a terminal case. At this point we can simply place our 4 guards and, if we need to, start a brand new call to GuardRight.

3.5 Time Complexity

It is clear that at most $O(n)$ initial calls to GuardRight can be made. For TG it is also easy to see that an initial call to GuardRight will result in a number of guards being placed in $O(n^2)$ time. We can therefore give an upper bound of $O(n^3)$ for the running time of TG.

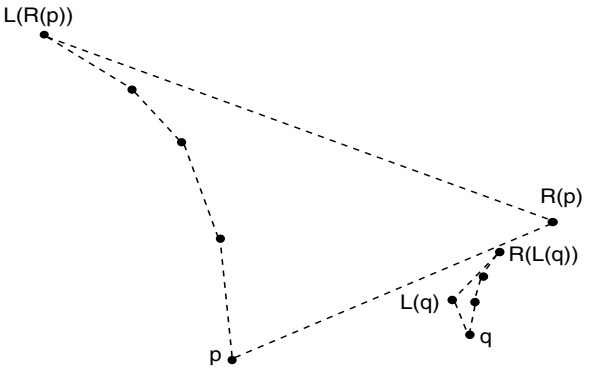

Fig. 5. The nested interval $[L(q), R(L(q))]$ can be handled independently with the help of a dominant outside vertex

If we want TG to be more efficient, we can make GUARDRIGHT(p, c) continue placing guards until $[p, R(p))$ has been completely guarded. This changes things slightly; on a given iteration, p is not necessarily unguarded so there is not necessarily an exposed vertex. If there is no exposed vertex, however, we can just recurse immediately by calling GUARDLEFT(q, c) so this is not a problem. To increase efficiency, we can sort the exposed vertices v by $R'(v)$ (breaking ties using the x-coordinates of exposed vertices) to find an appropriate b and d faster in each iteration. A call to GUARDRIGHT(p, c), ignoring all recursive calls that it spawns, can now run in $O(n + m \log m)$ time, where m is the number of exposed vertices in $[p, R(p))$. It is easy to see that the 'n' terms, added up over the entire course of the algorithm, will cost $O(n^2)$ time since there will be at most $O(n)$ calls to GUARDRIGHT. Any vertex will be an exposed vertex for at most one call to GUARDRIGHT, so the sum of all $m \log m$ factors encountered will actually be bounded by $O(n \log n)$. All other overhead incurred by the algorithm can be dealt with in $O(n^2)$ time, so the running time of TG is bounded by $O(n^2)$.

3.6 Modifications for TG-C

No real modifications need to be made to apply our TG algorithm to TG-C. However, we need to keep track of more information if we want our algorithm to run as efficiently as possible. When dealing with TG-C the only real problem is finding b and d at each iteration of a call to GUARDRIGHT. Instead of exposed vertices and sheltered vertices, we consider exposed edge sections and sheltered edge sections. It is not difficult to see that for each edge of the terrain, at most one contiguous section will be exposed and at most one will be sheltered. From left to right on an edge, we can have a guarded section, a sheltered section, an exposed section, and another guarded section, though not all of these sections will necessarily exist. For an exposed section, the leftmost point will have the leftmost R'.

$L(p)$ is always a vertex regardless of where p is. If we keep track of the transition points for the function $L(p)$ (there are only $O(n)$ of them [7]) then we can know where exposed sections end and sheltered sections begin. For every edge, our algorithm also keeps track of where the unguarded section starts and ends (it must be contiguous). After placing a guard, updating the unguarded section on every edge can be done quite easily in linear time. Assume we have just placed a guard at g. To the left of g call the first vertex v_1 and consider the edge e_1 whose left endpoint is v_1. Mark down that every point on e_1 is guarded. Now, moving left from v_1, find the first vertex above the line going through g and v_1; call this v_2, define e_2 appropriately and mark down that every point on e_2 above the line going through g and v_1 is guarded. It is easy to see how we can proceed to update the unguarded section of each edge in linear time. Since we place $O(n)$ guards the total cost of updating guarded edge sections of the terrain is $O(n^2)$.

If we do all of the aforementioned maintenance, we will only need to consider the leftmost point in each exposed section when looking for b and d. Therefore we do not need to worry about asymptotically more points in TG-C than in TG. The running time therefore remains $O(n^2)$.

4 Conclusions and Future Work

The 1.5D terrain guarding problem is not known to be in P. Constant-factor approximation algorithms for the problem have only recently been developed. We have provided an $O(n^2)$ time 4-approximation algorithm for both the discrete and continuous versions of the problem. Ours is the best known algorithm for the 1.5D terrain guarding problem.

The most pressing and obvious question regarding the 1.5D terrain guarding problem is whether or not it is NP-complete. All of our attempts at an NP-hardness proof have been stymied by the Order Claim. On the other hand, attempts at designing an exact polynomial-time algorithm have also been unsuccessful. If the problem is not NP-hard, we would be interested in a polynomial-time algorithm. If the problem is NP-hard, we would be interested in approximability thresholds, *e.g.* whether it is APX-complete or admits a PTAS or even an FPTAS. If the problem is APX-complete, the approximation factor should be lowered as much as possible.

Acknowledgements

This research was done under the supervision of Will Evans; I am extremely grateful for the help and guidance he gave me. I am also grateful to Boaz Ben-Moshe for the helpful discussions we had. Finally, I thank the anonymous referees for their helpful comments.

References

1. A. Aggarwal. *The art gallery problem: Its variations, applications, and algorithmic aspects*. PhD thesis, Johns Hopkins University, 1984.
2. B. Ben-Moshe. Personal communication, 2005.
3. B. Ben-Moshe, M. Katz, and J. Mitchell. A constant-factor approximation algorithm for optimal terrain guarding. In *Symposium on Discrete Algorithms*, 2005.
4. H. Brönnimann and M. T. Goodrich. Almost optimal set covers in finite VC-dimension. *Discrete & Computational Geometry*, 14, 1995.
5. D. Z. Chen, V. Estivill-Castro, and J. Urrutia. Optimal guarding of polygons and monotone chains (extended abstract), 1996.
6. V. Chvátal. A combinatorial theorem in plane geometry. *J. Comb. Theory Series B*, 18:39–41, 1975.
7. K. L. Clarkson and K. Varadarajan. Improved approximation algorithms for geometric set cover. In *Symposium on Computational Geometry*, 2005.
8. S. Eidenbenz. *(In-)Approximability of Visibility Problems on Polygons and Terrains*. PhD thesis, ETH Zurich, 2000.
9. S. Eidenbenz. Approximation algorithms for terrain guarding. *Information Processing Letters*, 82(2):99–105, April 2002.
10. S. Eidenbenz, C. Stamm, and P. Widmayer. Positioning guards at fixed height above a terrain — an optimum inapproximability result. *Lecture Notes in Computer Science*, 1461, 1998.
11. S. Eidenbenz, C. Stamm, and P. Widmayer. Inapproximability results for guarding polygons and terrains. *Algorithmica*, 31(1):79–113, 2001.
12. U. Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45(4):634–652, July 1998.
13. M. Garey and D. Johnson. *Computers and Intractibility: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Co., 1979.
14. D. Haussler and E. Welzl. ϵ-Nets and Simplex Range Queries. *Discrete & Computational Geometry*, 2:127–151, 1987.
15. James King. Approximation algorithms for the 1.5 dimensional terrain guarding problem. Master's thesis, University of British Columbia, 2005.
16. A. A. Kooshesh and B. M. E. Moret. Three-coloring the vertices of a triangulated simple polygon. *Pattern Recognition*, 25, 1992.
17. D. T. Lee and A. K. Lin. Computational complexity of art gallery problems. *IEEE Transactions on Information Theory*, 32:276–282, 1986.
18. M. Sharir and P. K. Agarwal. Davenport-Schinzel sequences and their geometric applications. Technical report, 1995.