A Constant-Factor Approximation Algorithm for Optimal 1.5D Terrain Guarding*

Boaz Ben-Moshe† Matthew J. Katz‡ Joseph S. B. Mitchell§

August 29, 2006

Abstract

We present the first constant-factor approximation algorithm for a non-trivial instance of the optimal guarding (coverage) problem in polygons. In particular, we give an $O(1)$-approximation algorithm for placing the fewest point guards on a 1.5D terrain, so that every point of the terrain is seen by at least one guard. While polylogarithmic-factor approximations follow from set cover results, our new results exploit geometric structure of terrains to obtain a substantially improved approximation algorithm.

1 Introduction

For a geometric domain $\mathcal{D}$, the *optimal guarding (coverage) problem* is to determine the smallest number, k^*, of *guards* (e.g., points) that can be placed in $\mathcal{D}$ so that each point of $\mathcal{D}$ is *seen* by at least one guard. The optimal guarding problem is an instance of a set cover problem that is induced by a geometric setting. The many related problems, both combinatorial and algorithmic, fall into the general category known as *art gallery problems*, which have been studied extensively; see, e.g., [12, 14, 16, 17].

We give the first constant-factor approximation algorithm for a non-trivial instance of the optimal guarding problem. All prior approximation bounds were polylogarithmic in n and/or k^*.

The instance we study here is the 1.5D *terrain guarding problem*, in which the input domain is the two-dimensional region above an x-monotone polygonal chain having n vertices. We restrict guards to be placed at points on the terrain; with no restriction on guard placement, a single guard (at a high enough altitude) suffices to see the terrain.

Related Work. The classical combinatorial result, the "art gallery theorem", states that $\lfloor n/3 \rfloor$ guards are sufficient, and sometimes necessary, to guard an n-vertex simple polygon [14]. Combinatorial results on the number of guards needed for various forms of guarding on terrains are given in [2].

The optimal guarding problem is known to be NP-hard, even if $\mathcal{D}$ is a simple polygon [15]. Thus, efforts have concentrated on the approximability of optimal guarding problems. Ghosh [10]

*An extended abstract of this paper appeared in [1].

†College of Judea & Samaria, Ariel 44837, Israel. benmo@yosh.ac.il.

‡Ben-Gurion University, Beer-Sheva 84105, Israel. matya@cs.bgu.ac.il.

§Stony Brook University, Stony Brook, NY 11794-3600. jsbm@ams.sunysb.edu.

gave an $O(\log n)$-approximation for optimal coverage of a polygon by vertex guards, based on standard set cover results. Recent interest [6, 11] has focused on methods of efficiently applying the Brönnimann-Goodrich technique [3], which exploits finiteness of VC-dimension. Efrat and Har-Peled [6] obtain an $O(\log k^*)$-approximation algorithm for polygon guarding with vertex guards, using time $O(n(k^*)^2 \log^4 n)$, where k^* is the optimal number of vertex guards. Their technique can be applied to non-vertex guards, lying at points of a dense grid, adding to the running time a factor polylogarithmic in the grid density. (No approximation algorithm is known if the guards are completely unrestricted and all of the polygon is to be guarded.) Their results apply also to polygons with holes and to 2.5D terrains, still with polylogarithmic approximation factors. Most recently, Cheong et al. [5] have shown how to place k guards in order to optimize (approximately) the total area seen by the guards.

Eidenbenz et al. [7, 9] have shown that it is APX-hard to solve the optimal guarding problem in simple polygons; thus, there exists an $\epsilon > 0$ such that it is NP-hard to obtain a $(1+\epsilon)$-approximation algorithm. They further show that guarding polygons with holes, as well as guarding 2.5D terrains, is as hard as set cover; these problems do not have polynomial-time algorithms with approximation ratio less than $c \ln n$, for some $c > 0$, unless $P = NP$ (and do not have approximation ratio $\frac{1-\epsilon}{12} \ln n$, for any $\epsilon > 0$, unless $NP \subset TIME(n^{O(\log \log n)})$). These hardness of approximation results are complemented by $O(\log n)$-approximation algorithms based on greedy set cover [8].

For 1.5D terrains, Chen et al. [4] claim that a modification of the hardness proof of [15] shows that the problem is NP-hard (details are omitted and are still to be verified). In the very restricted setting where all guards are to be placed at a constant altitude (above "sea level", not above the terrain surface), the optimal guarding problem is readily solved in polynomial time (it reduces to a one-dimensional problem on intervals); in fact, it can be solved in linear time [7, 13].

Motivation. The terrain guarding problem arises in optimal placement of antennas for communication networks. We are motivated to study the problem in two dimensions, in an effort to understand better the much more difficult three-dimensional (2.5D) terrain guarding problem (which Eidenbenz shows has no better than a log-approximation). Further, the two-dimensional problem shows up as a subproblem in heuristics to solve the three-dimensional problem, and it arises directly in applications of coverage along a highway, security lamp and camera placement along walls and streets. In some applications, one only needs to cover a portion of the input domain; our results can be extended to yield an algorithm for covering a subset of the terrain surface optimally.

Our Results. Our main result is an efficient $O(1)$-approximation algorithm for the terrain guarding problem in two dimensions. Our results rely on developing several geometric properties of terrains, in order that our algorithm can exploit the special structure.

The main difficulty in proving our main result is in showing that one can obtain a constant-factor approximation to the dominating set problem in graphs that can be realized as the visibility graph of the vertices of a terrain polygon (i.e., to the problem of placing vertex guards on a terrain in order to see all other vertices). It is then relatively straightforward to extend this result to apply to placing arbitrary guards on the terrain in order to guard all of the terrain.

It is worth noting that some natural and simple approaches to the problem do not yield $O(1)$-approximations: One can give examples showing a subset of vertices guarding T, without redundant vertices, that is arbitrarily bad with respect to optimal (see Figure 1); similarly, simple sweep

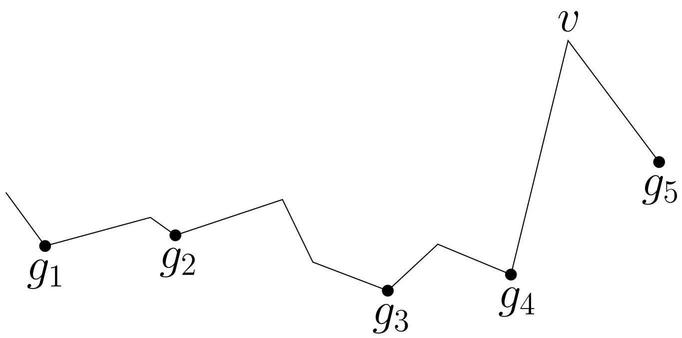

Figure 1: None of the guards $g_1, \ldots, g_5$ is redundant, but a single guard at v would suffice.

approaches that add guards "as needed" can be arbitrarily bad.

We begin with some definitions and basic structural results.

2 Preliminaries

Let T be a terrain of complexity n in the plane; more precisely, let T be an x-monotone polygonal curve specified by n vertices, $v_1, \ldots, v_i = (x_i, y_i), \ldots, v_n$, indexed in x-increasing order. We refer to the *terrain polygon*, P_T, which is the closed region of the plane bounded from below by T and from the sides by the upwards vertical rays emanating from v_1 and v_n, respectively.

Let $p = (p_x, p_y)$ and $q = (q_x, q_y)$ be two points on the polygonal curve T. We say that p *sees* q (and q *sees* p) if and only if the line segment $\overline{pq}$ is contained in P_T. We write $p < q$ if p is to the left of q (i.e., $p_x < q_x$). For any two vertices $u, v \in T$ with $u < v$, the *subterrain* $[u, v]$ of T is simply the portion of T between u and v (including u and v). For a point $p \in T$, let $L(p)$ (resp., $R(p)$) denote the leftmost (resp., rightmost) point on T that sees p. It is easy to see that $L(p)$ and $R(p)$ are necessarily vertices of T.

A *guard* is a point of P_T that we consider to be able to view ("guard" or "cover") other points of P_T. A *terrain guard* is any point on T; a *vertex guard* is a vertex of T.

The *must-guard set*, M, is a subset of P_T that we require to be seen by a set of guards. The *dominating set problem on terrains* (DSPT) is to compute a minimum-cardinality set of vertex guards for $M = \{v_1, \ldots, v_n\}$, the vertex set of T. We spend most of our effort in obtaining an approximation algorithm for DSPT; then, we describe how this result extends to the case of terrain guards and to the case of $M = T$. (Note that a set of guards that sees $M = T$ necessarily sees all of P_T; however, it is not the case that guarding the boundary of a general simple polygon P implies guarding the interior of P.)

We say that a subterrain $[u, v]$ *can be guarded from the right (resp., left)* if there exists a set of vertex guards to the right of v (resp., to the left of u) that cover $M \cap [u, v]$.

Claim 2.1 *[Order Claim] Let $a < b < c < d$ be four points on T. If a sees c and b sees d, then a sees d.*

Proof: Since a sees c, b lies below the line segment $\overline{ac}$; similarly, c lies below $\overline{bd}$. Thus, $\overline{ac}$ and $\overline{bd}$ cross each other at a point ξ (above T), and T lies below the chain (a, ξ, d), implying that a sees d. $\square$

3 The Main Algorithm

In this section we outline a constant-factor approximation algorithm for the dominating set problem on terrains (DSPT): Find a minimum-size subset G of vertices of T, such that each vertex of T is visible from one of the vertices in G. (The missing details are presented in Section 4.)

We begin by computing the (upper) convex hull of T, denoted $CH(T)$. Notice that if m is the number of vertices in $CH(T)$, then at least $\lfloor \frac{m}{2} \rfloor$ guards are needed in order to guard T. In the first stage, we thus place a guard at each vertex of $CH(T)$. Notice that if T' is a subterrain defined by two consecutive vertices of $CH(T)$, then no point in its interior is visible from outside T'. We therefore may consider each of the $m-1$ subterrains that are defined by the vertices of $CH(T)$ separately (see Figure 2).

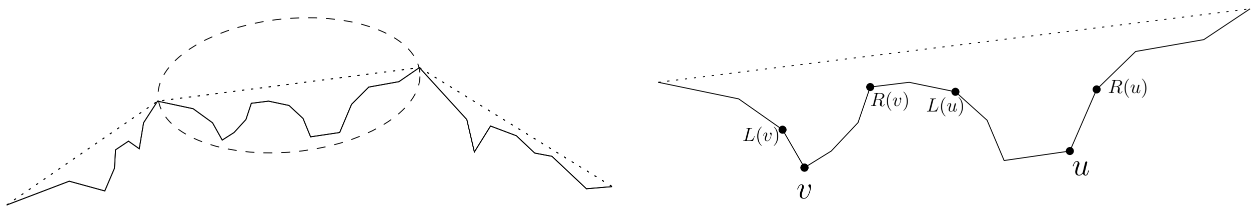

Figure 2: The overall structure of the algorithm.

Let $T'' = [u, v]$ be a subterrain of T'. We say that T'' *requires a local guard* if there exists a vertex w in the interior of T'' that is not seen from $(T' - T'') \cup \{u, v\}$, or, in other words, $u < L(w) < R(w) < v$.

Let T' be one of the subterrains obtained from the first stage. In the second stage we partition T' into subterrains such that each of them does not require a local guard. We do this as follows. For each internal vertex v of T', we compute the points (vertices) $L(v)$ and $R(v)$. Let $T(v)$ denote the subterrain $[L(v), R(v)]$. Viewing the subterrains $T(v)$ as open horizontal intervals and considering only *minimal* intervals (i.e., intervals that do not contain other intervals), we compute a maximal set $\mathcal{S}$ of disjoint such intervals. Let $\mathcal{S}'$ be the complementary set of subterrains; thus, any two consecutive subterrains in $\mathcal{S}$ define at most one subterrain in $\mathcal{S}'$.

Claim 3.1 *Let T'' be a subterrain in $\mathcal{S} \cup \mathcal{S}'$. Then T'' does not require a local guard.*

Proof: Follows immediately from the construction. □

We place guards at the end vertices of the subterrains in $\mathcal{S}$. In addition, for each $T'' \in \mathcal{S} \cup \mathcal{S}'$, we place guards at the at most 4 vertices $R(a_l), R(a_r), L(b_l), L(b_r)$, where a_l (resp., a_r) is the leftmost (resp., rightmost) internal vertex of T'' that is seen from the right of T'', and b_l (resp., b_r) is the leftmost (resp., rightmost) internal vertex of T'' that is seen from the left of T''. Since $|\mathcal{S}|/2$ is clearly a lower bound on the number of guards needed to guard T', we only increase the number of vertices by a constant factor.

We now "solve" each of the subterrains in $\mathcal{S} \cup \mathcal{S}'$ *separately*. Each such subterrain is considered to be a *base case*, in that it does not require a local guard, and is solved using the base-case algorithms detailed in Section 4. The *independence property* (based on Claims 3.3-3.4) shown below justifies this approach, i.e., that the subterrains in $\mathcal{S} \cup \mathcal{S}'$ may be solved separately without hurting the approximation bound.

The overall structure of the algorithm is thus as follows.

- Given a terrain T compute its (upper) convex hull, place guards at the vertices of the convex hull, and solve each subterrain T' separately.
- Given a subterrain T' partition it into subterrains $\mathcal{S} \cup \mathcal{S}'$, as described above. For each subterrain $T'' \in \mathcal{S} \cup \mathcal{S}'$, place guards at the end vertices of T'' and at the vertices $R(a_l), R(a_r), L(b_l), L(b_r)$, and solve the remaining unguarded fragments of T'' separately, using the base-case algorithms detailed in Section 4.

The rest of this section deals with the independence property and its proof.

Lemma 3.2 *[The Independence Property] Let T be a terrain, and let $T_1, \ldots, T_k$ be k disjoint subterrains of T. (Two subterrains may have a common end vertex.) Assume that the size l of an optimal solution for $T_1 \cup \cdots \cup T_k$ (by placing guards on T) is greater than or equal to k/c_1, for some constant c_1. And assume that for each subterrain T_i, we can compute a c_2-solution for (the yet unguarded fragments of) T_i, for some constant c_2. Then we can compute a c-solution for $T_1 \cup \cdots \cup T_k$, for some constant c. (A c_0-solution is a solution whose size is at most c_0 times the size of an optimal solution; thus a c-solution is of size at most cl.)*

We construct a c-solution for $T_1 \cup \cdots \cup T_k$. We begin by placing at most $2k$ guards at the end vertices of the subterrains $T_1, \ldots, T_k$. We need the following two claims.

Claim 3.3 *Let $r_1 < r_2 < \cdots < r_m$ be the internal vertices of T_i that can be seen from the right of T_i. Then*

1. *$R(r_1) \geq R(r_2) \geq \cdots \geq R(r_m)$.*
2. *If r_i, for $1 < i < m$, cannot be seen both from $R(r_m)$ and from $R(r_1)$, then it can only be seen (when viewing from the right of T_i) from vertices in the interior of the subterrain $[R(r_m), R(r_i)] \subseteq [R(r_m), R(r_1)]$.*

Proof: The first part follows immediately from the Order Claim. If $R(r_2) > R(r_1)$, then by the Order Claim r_1 must see $R(r_2)$, which is impossible since r_1 does not see beyond $R(r_1)$. Similarly, we argue that $R(r_2) \geq R(r_3)$, etc.

To prove the second part, assume that r_i (for some $1 < i < m$) cannot be seen from $R(r_m)$ or from $R(r_1)$. Then r_i cannot be seen from a vertex to the left of $R(r_m)$ (and to the right of T_i), since if r_i is seen from such a vertex, then, by the Order Claim, it is also seen from $R(r_m)$. Similarly, r_i cannot be seen from a vertex to the right of $R(r_1)$, since if it is seen from such a vertex, then, by the Order Claim, r_1 is also seen from this vertex, which is impossible (by the definition of $R(r_1)$). $\Box$

Put $R(T_i) = [R(r_m), R(r_1)]$, and let $L(T_i)$ be the symmetric subterrain that is defined by considering the internal vertices of T_i that can be seen from the left of T_i .

Claim 3.4 *Let T_i, T_j be two of the subterrains above, such that T_j lies to the right of T_i. Let $R_j(T_i) \subseteq R(T_i)$ be the subterrain defined by considering only the internal vertices in T_i that can be seen from the right of T_j. Then $R(T_j)$ lies to the left of $R_j(T_i)$, where the right end vertex of $R(T_j)$ and the left end vertex of $R_j(T_i)$ may coincide.*

Proof: Assume there is an internal vertex in T_i that can be seen from the right of T_j. (Otherwise, $R_j(T_i)$ is empty.) Let u be the leftmost vertex in T_j that can be seen from the right of T_j. Then $R(u)$ defines the right end vertex of $R(T_j)$. By the Order Claim, if v is any vertex in T_i that can be seen from the right of T_j, then $R(v) \geq R(u)$. □

A symmetric claim can be formulated using the subterrains $L(T_i)$ and $L_i(T_j)$ instead of $R(T_j)$ and $R_j(T_i)$, respectively.

We are now ready to continue the construction of a c-solution for $T_1 \cup \cdots \cup T_k$. For each T_i, we place guards at the end vertices of the subterrains $R(T_i)$ and $L(T_i)$. We have thus placed at most $4k$ guards in this step (and at most $6k$ guards in the first two steps). Next, for each T_i we compute a c_2-solution $\mathcal{V}_i$ for (the yet unguarded fragments of) T_i. We claim that the at most $6k$ guards of the first two steps together with the sets $\mathcal{V}_i$ form a c-solution for $T_1 \cup \cdots \cup T_k$.

Let $\mathcal{V}_{opt}$ be an optimal solution for $T_1 \cup \cdots \cup T_k$. Recall that we are assuming that $|\mathcal{V}_{opt}| \geq k/c_1$, for some constant c_1. Let $\mathcal{U}$ denote the set of at most $4k$ vertices that are the end vertices of the subterrains $R(T_i)$ and $L(T_i)$. Let $v \in \mathcal{V}_{opt} - \mathcal{U}$. We observe that there is at most one subterrain T_i to the left of v, that has an internal vertex that is not seen by a vertex of $\mathcal{U}$ but is seen from v. If there are two such subterrains T_i and T_j, where T_j is to the right of T_i, then, by Claim 3.3, v is an internal vertex of both $R(T_i)$ and of $R(T_j)$. More precisely, v is in $R_j(T_i)$ (but not its right end vertex) and v is an internal vertex of $R(T_j)$. But this is impossible by Claim 3.4. Thus v can help guarding at most one subterrain to its left, at most one subterrain to its right, and possibly the subterrain in which it lies. Our construction replaces v with at most $3c_2$ guards.

4 Base-Case Algorithms

Let T be a terrain with n vertices, and let $\mathcal{G}$ be the subset of vertices of T where guards have already been placed. Let A be a subterrain of T that does not require a local guard, and let A' be the subset of vertices of A that cannot be seen by $\mathcal{G}$. We wish to compute a set of guards $\mathcal{V}(A')$ located at vertices of T, some of which may be located within A, that together see all vertices in A', and such that the size of $\mathcal{V}(A')$ is within some constant factor of the size of an optimal such set of guards.

We distinguish between three base cases.

(Case 0) We require that $\mathcal{V}(A')$ consists only of vertices that lie to the *left* of subterrain A.

(Case 1) We require that $\mathcal{V}(A')$ consists only of vertices that lie *left* of or *within* subterrain A.

(Case 2) We make no requirements of the set $\mathcal{V}(A')$ of vertices of T that guard subterrain A.

Each subterrain $T'' \in \mathcal{S} \cup \mathcal{S}''$ (see Section 3) is passed to the Case 2 algorithm, which in turn may pass it on to the Case 1 algorithm; see Figure 3.

Definition 4.1 *Let T be a terrain and let $\mathcal{G}$ be the subset of vertices of T where guards have already been placed. Let T' be the subset of vertices of T that cannot be seen by $\mathcal{G}$. We say that two subterrains $T_1, T_2 \subset T$ are* guard independent *with respect to $\mathcal{G}$ if the set of vertices of T that are seen by $T_1 \cap T'$ and the set of vertices seen by $T_2 \cap T'$ are disjoint. In other words, T_1 and T_2 are* guard-independent *subterrains if any guard g that can see a vertex in $T_1 \cap T'$ cannot see a vertex in $T_2 \cap T'$ and vice versa.*

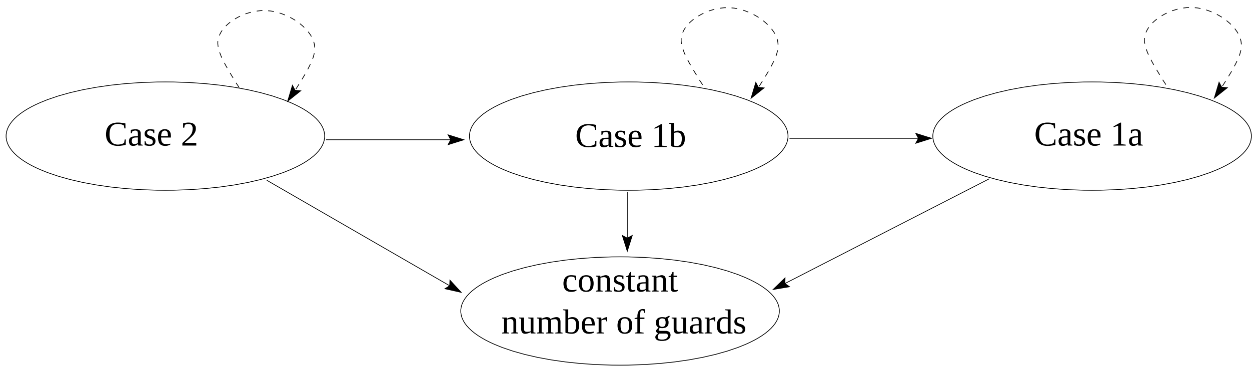

Figure 3: The program flow of the base case algorithms: A dashed line denotes a recursive call, while a solid line denotes a call to another (simpler) algorithm.

4.1 Case 0

Let A be a subterrain of T such that it is possible to guard the set of (so far unguarded) vertices A' using only guards at vertices of T to the left of A. Our goal is to determine a minimum-cardinality such set of *left guards* for A'. In this specially constrained case, we are able to determine an *optimal* set, $\mathcal{V}(A')$, of guards, using the following algorithm:

While A' contains an unguarded vertex, do

Place a guard at $L(a)$, where a is the leftmost vertex in A' that is not yet guarded.

Lemma 4.2 *The algorithm above computes an optimal subset of left guards for A'.*

Proof: When the algorithm locates a guard at vertex $L(a)$, no vertex to the left of $L(a)$ can see a (by definition of $L(a)$), and any vertex v to the right of $L(a)$ (and still to the left of A) that sees a is "dominated" by $L(a)$ (by the Order Claim), in that a guard at $L(a)$ will see any vertex of A' that v sees. We continue by induction. $\square$

4.2 Case 1

Let $A = [a, b]$ be a subterrain that does not require a local guard, let A' be the subset of vertices of A that remain to be guarded, and assume that A' can be guarded from the left of A, that is, by placing guards only at vertices of T to the left of A.

In this case (Case 1), our goal is to compute a set of guards $\mathcal{V}(A')$ for A' such that each guard in $\mathcal{V}(A')$ is either to the left of A or within A (but not to the right of A). We present a constant-factor approximation algorithm for computing a minimum-cardinality such set of guards.

We will need the following claim, which tells us that it does not make sense to place a guard within A if its sole purpose is to view rightwards.

Claim 4.3 *Let $A = [a, b]$ be a subterrain as above. Let $u \neq b$ be any vertex in A. Then $L(u)$ dominates u, in the sense that any vertex $v \in A'$ to the right of u that is seen by u is also seen by $L(u)$.*

Proof: Let $v \in A'$ be a vertex to the right of u that is seen by u. Recall that by our assumption $L(v)$ lies to the left of A. We may also assume that $L(u) \neq L(v)$, since otherwise $L(u)$ clearly sees v. Now, on the one hand, both u and $L(u)$ must lie below the line $l(L(v), v)$, and, on the other hand, $L(u)$ must lie above the line $l(L(v), u)$. Thus $L(u)$ can see v (see Figure 4). $\square$

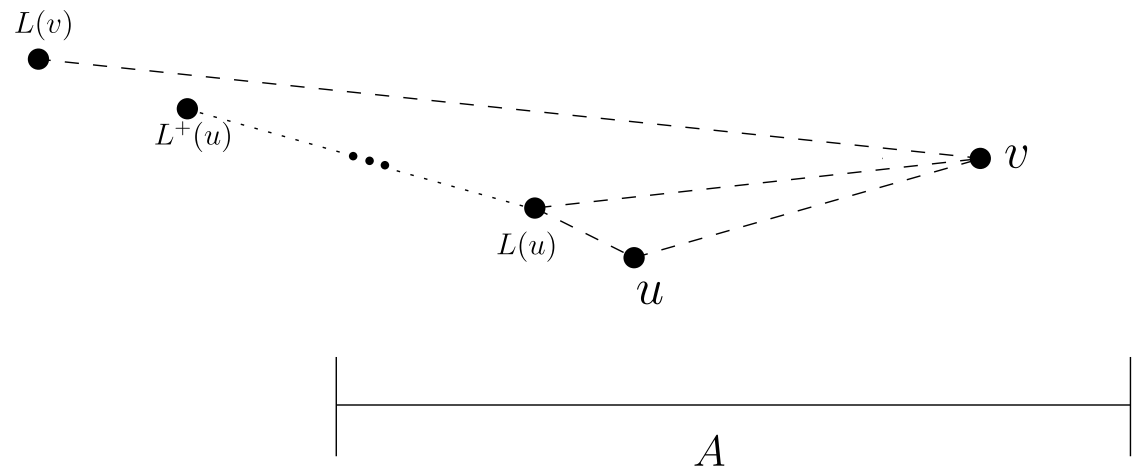

Figure 4: If u can see v, then $L(u)$ can also see v. If $L(u) \in A$ we apply Claim 4.3 to $L(u)$ to conclude that $L(L(u))$ dominates $L(u)$ and therefore also sees v, etc., until we reach a vertex that is not in A that dominates u.

Corollary 4.4 *There is always a vertex to the left of A that dominates u (with respect to the vertices of A' to the right of u). We shall denote by $L^+(u)$ any such vertex.*

Proof: If $L(u) \in A$, then we apply Claim 4.3 to $L(u)$ to conclude that $L(L(u))$ also sees v, etc., until we reach a vertex that is not in A. This vertex clearly dominates u. $\square$

Corollary 4.5 *$L^+(u)$ dominates any vertex w that lies between $L^+(u)$ and u (with respect to the vertices of A' to the right of u).*

Proof: Follows from the fact that w must lie below the chain $u, L(u), L(L(u)), \ldots, L^+(u)$. $\square$

We consider two subcases, according to whether or not the endpoints of A see each other.

4.2.1 Case 1a: The endpoints of A see each other.

Let $A = [a, b]$ be a subterrain such that a and b see each other. For a vertex $q \in A$, $q \neq b$, that is visible from b, we denote by $A'_l(q)$ the vertices of A' that lie to the left of q and are not visible from b or $L^+(q)$. Similarly, let $A'_r(q)$ denote the vertices of A' that lie to the right of q and are not visible from b or $L^+(q)$. If both $A'_l(q)$ and $A'_r(q)$ are nonempty, we say that q implies a *non-trivial division*.

Claim 4.6 *If there exists a vertex $q \in A$, $q \neq b$, such that, b sees q, and both $A'_l(q)$ and $A'_r(q)$ are nonempty (i.e., q implies a non-trivial division), then $A'_l(q)$ and $A'_r(q)$ are guard independent.*

Proof: The vertices that can see one or more vertices of $A'_l(q)$ can lie either to the right of $L^+(q)$ and to the left of A, or in the subterrain $[a, q]$. This follows from the fact that b can see q, and that any vertex to the left of $L^+(q)$ cannot see into $A'_l(q)$. Similarly, the vertices that can see one or more vertices in $A'_r(q)$ can lie either to the left of $L^+(q)$ or in the subterrain (q, b). Notice that q cannot see a vertex of $A'_r(q)$, since $L^+(q)$ cannot see such a vertex and, according to Claim 4.3, $L^+(q)$ dominates q with respect to the subterrain to the right of q. □

We say that A is in the *single-pocket case* if each vertex $q \in A$ to the left of b that is visible from b implies a trivial division (either $A'_l(q)$ is empty or $A'_r(q)$ is empty).

Claim 4.7 *Consider a subterrain A that is in the single-pocket case. Then there exists an (open) subterrain $A^* = (c, d) \subset A$, $d \neq b$, such that, b cannot see any vertex in A^*, and all the vertices of A' that are not visible from b, $L^+(c)$, and $L^+(d)$ are in A^*.*

Proof: We may assume that there is no vertex q (to the left of b and visible from b) for which *both* $A'_l(q)$ and $A'_r(q)$ are empty; otherwise, we could place guards at b and $L^+(q)$ to see all of A'. Let c be the rightmost vertex (to the left of b and visible from b) that implies a (trivial) division in which $A'_l(c)$ is empty. Let d be the first vertex to the right of c that is visible from b. Then d implies a (trivial) division in which $A'_r(d)$ is empty. Put $A^* = (c, d)$. By definition, b cannot see any vertex in A^*. Also, any vertex in A' that is not visible from b, $L^+(c)$, or $L^+(d)$ must lie in A^*, since, by definition, b and $L^+(c)$ cover $[a, c) \cap A'$ as well as c, and b and $L^+(d)$ cover $(d, b] \cap A'$ as well as d. □

The single-pocket case. Consider the single-pocket case, where $A^* = (c, d)$ denotes the "pocket". Notice that (i) none of the vertices in $(d, b]$ can see into the pocket A^* (by the Order Claim), and (ii) the vertices c and d see each other (since b sees both c and d but does not see any vertex in A^*). Also we know that b, $L^+(c)$ and $L^+(d)$ together cover $A' - A^*$.

For a subset of vertices S, we denote by $V(S, p)$ the subset of vertices of S that are visible from a vertex p, and by $V(S, P)$ the subset of vertices of S that are visible from at least one of the vertices of P.

Lemma 4.8 *If $V(A', L^+(d)) \cup V(A', d) \not\supseteq A' - A^*$, then (i) one must place at least one guard outside of A^* in order to guard $A' - A^*$, and (ii) there exists a constant-size set of guards U that guards $A' - A^*$, and any other set of guards U' that guards $A' - A^*$ includes a guard g' such that $V(A' \cap A^*, U) \supseteq V(A' \cap A^*, g')$.*

Proof: To prove the first part, we observe that $L^+(d)$ dominates any vertex in A^* with respect to the subterrain to the right of d (applying Corollary 4.5), and d dominates any vertex in A^* with respect to the subterrain to the left of c (applying the Order Claim to d and c). Now let $v \in A' - A^*$ be a vertex that is not seen from $L^+(d)$ or from d (such a vertex exists by our assumption). Clearly any guard that sees v cannot lie in A^*.

To prove the second part, put $U = \{L^+(d), L^+(c), b\}$. We already know that U guards $A' - A^*$. We now show that if g' is a guard that sees v, then $V(A' \cap A^*, U) \supseteq V(A' \cap A^*, g')$. If v is to the right of A^*, then in order to guard v one must locate a guard g' either to the left of $L^+(d)$ or to the right of d. In both cases $V(A' \cap A^*, g') = \emptyset$. Otherwise, if v is to the left of A^*, then in order to guard v one must locate a guard g' either in $[L^+(c), c]$ or to the right of d. In the former case it is possible that g' sees into A^*, but it is dominated by $L^+(c)$. □

The lemma above implies that if $V(A', L^+(d)) \cup V(A', d) \not\supseteq A' - A^*$, then we may place three guards at $L^+(d)$, $L^+(c)$, and b, and charge these guards to the guard g' (in the proof above).

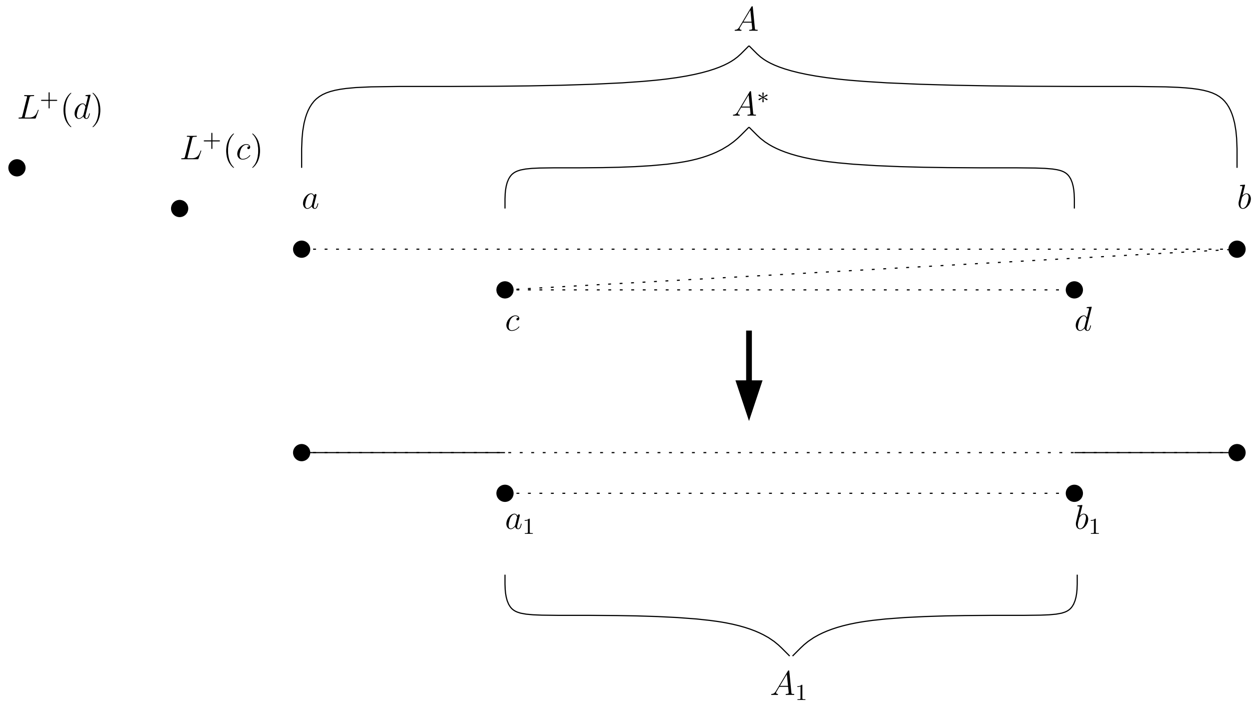

Figure 5: We can reduce the problem A to the Case 1a problem $A_1 = [a_1, b_1] = [c, d]$, since $A' - A^* \subseteq V(A', L^+(d)) \cup V(A', d)$.

We now consider the case in which $V(A', L^+(d)) \cup V(A', d) \supseteq A' - A^*$ (see Figure 5). In this case we say that A is *reducible*, and reduce $A_0 = A$ to the Case 1a problem $A_1 = [a_1, b_1] = [c, d]$. Now, if A_1 is also reducible, that is, if A_1 is a single-pocket case with pocket $A_1^* = (c_1, d_1)$, and $V(A', L^+(d_1)) \cup V(A', d_1) \supseteq A' - A_1^*$, where A' is the *original* A', then we reduce A_1 to the Case 1a problem $A_2 = [a_2, b_2] = [c_1, d_1]$. This process continues until we reach a problem $A_i = [a_i, b_i]$, $i > 0$, that is not reducible.

Notice that if we now place two guards at b_i and at $L^+(b_i)$, then all remaining unguarded vertices in A' would necessarily lie in the interior of A_i. This is because $b_i = d_{i-1}$ and A_{i-1} was reducible. Thus we place these two guards and proceed as follows, according to the state that we entered. There are three possible such states:

- A_i is not a single-pocket case, i.e., there is a vertex $q \in A_i$ that implies a non-trivial division.
- A_i is a single-pocket case, but $V(A', L^+(d_i)) \cup V(A', d_i) \not\supseteq A' - A_i^*$.
- A_i has a constant-size solution.

In the first state, we divide A_i into two guard-independent subterrains (see Claim 4.6) as follows. Let q_l be the leftmost vertex in A_i that is seen from b_i and implies a non-trivial division. Let a_i'

be the rightmost vertex to the left of q_l that is seen by b_i. Notice that q_l and a'_i see each other, since b_i sees both of them and does not see any vertex between them. Also notice that b_i and $L^+(a'_i)$ together cover the subterrain $[a_i, a'_i]$, since a'_i implies a (left) trivial division. (Notice that it is impossible that a'_i implies a right trivial division, since, if it would, then so would q_l.)

Thus, in addition to the two guards that were already placed (i.e, at b_i and at $L^+(b_i)$), we also place guards at $L^+(q_l)$ and at $L^+(a'_i)$. We now solve the two guard-independent subterrains $A_l = [a'_i, q_l]$ and $A_r = (q_l, b_i]$ by applying to each of them the Case 1a algorithm. We charge the four guards that were placed to the increase by one of the lower bound due to the presence of a vertex that implies a non-trivial division.

In the second state we also place guards at $L^+(d_i)$ and $L^+(c_i)$ (in addition to the guards at b_i and $L^+(b_i)$) according to Lemma 4.8 above, and solve the subproblem $[c_i, d_i]$ by applying the Case 1a algorithm. We charge the four guards that were placed to the guard g' (see above). In the third state we simply solve the original problem with a constant number of guards.

All the above implies the following algorithm for Case 1a.

Algorithm 4.9 (Case 1a)

1. *If there exist 2 guards that together see all vertices in A' — done.*
2. *Let Q be the set of all vertices $q \in A$ that imply a non-trivial division.*
3. *If $Q \neq \emptyset$, let q_l be the leftmost vertex in Q. Locate three guards at b, $L^+(q_l)$, and $L^+(a')$, where a' is the rightmost vertex to the left of q_l that is visible from b. Solve each of the (guard-independent) subterrains $A_l = [a', q_l]$ and $A_r = (q_l, b]$ recursively using the Case 1a algorithm.*
4. *If $Q = \emptyset$ (the single-pocket case)*
   (a) *Compute the pocket $A^* = (c_0, d_0)$.*
   (b) *If $A = A_0$ is not reducible, place a guard at b. Otherwise, reduce until a subterrain $A_i = [a_i, b_i]$ that is not reducible is reached and place guards at b_i and $L^+(b_i)$.*
   (c) *If A_i, $i \geq 0$, is a single-pocket case, then place guards at $L^+(c_i)$ and $L^+(d_i)$ and solve $[c_i, d_i]$ recursively using the Case 1a algorithm.*
   (d) *Else, solve A_i recursively using the Case 1a algorithm.*

The following lemma summarizes the result for Case 1a.

Lemma 4.10 *The Case 1a algorithm above computes a set of guards $\mathcal{V}(A')$ for A', whose size is bounded by some constant times the size of a minimum-cardinality such set of guards.*

We now turn to the general case, where A's endpoints do not see each other.

4.2.2 Case 1b: The endpoints of A do not see each other.

We begin by computing the (upper) convex hull of A. Each of the edges of this convex hull corresponds to a subterrain whose endpoints see each other. Some of these subterrains may already be fully guarded (by $\mathcal{G}$); we consider only those that are not yet fully guarded. Let $A_1 = [a, b]$ be the leftmost subterrain that is not yet fully guarded, let A'_1 be the subset of vertices in A_1 that are not yet guarded, and let u be the leftmost vertex in A_1'. If there exists a single vertex to the left of A that sees all vertices in A_1', then $L(u)$ is necessarily such a vertex. (Note that $L(u)$ is necessarily to the left of A since $u \in A_1'$.) Moreover, in this case, any vertex g (either to the left of A or in A) that sees u is dominated by the pair of vertices $L(u)$ and $L(b)$. Thus, in this case we can place guards at $L(u)$ and at $L(b)$ and charge this to the guard of the optimal solution that sees u.

Assume now that A_1' cannot be fully guarded by a single vertex to the left of A. In this case, we place a guard at $L(b)$ and distinguish between two cases. If after placing a guard at $L(b)$ there is no vertex to the right of b that remains to be guarded, then apply the Case 1a algorithm to subterrain A_1, and charge the guard at $L(b)$ to the one time event of leaving Case 1b. If however there still remains an unguarded vertex to the right of b, then the two subterrains A_1 and $A_2 = A - A_1$ are guard independent. Indeed, any guard that can help guarding A_1' must lie in $(L(b), b]$ and any guard that can help guarding A_2' must lie either in A_2 or to the left of $L(b)$. Notice that b cannot help in guarding A_2', since $L(b)$ dominates b with respect to visibility to the right of b.

Algorithm 4.11 (Case 1b)

1. *If all of A is guarded — done.*
2. *Let $A_1 = [a, b]$ be the leftmost subterrain that is not yet fully guarded, let A_1' be the subset of remaining unguarded vertices in A_1, and let u be the leftmost vertex in A_1'.*
3. *If there exists a single vertex to the left of A_1 that sees all vertices in A_1', then locate guards at $L(u)$ and at $L(b)$ go to Step #1.*
4. *Else, locate a guard at $L(b)$ and solve the two guard independent subproblems A_1, using the Case 1a algorithm, and $A - A_1$, using the Case 1b algorithm.*

The following lemma summarizes the result for Case 1b.

Lemma 4.12 *The Case 1b algorithm above computes a set of guards $\mathcal{V}(A')$ for A', whose size is bounded by some constant times the size of a minimum-cardinality such set of guards.*

4.3 Case 2

Given a terrain T, let $A = [a, b]$ be a subterrain that does not require a local guard, let A' be the subset of all vertices in A that are not yet guarded. In this case (Case 2), our goal is to compute a set of guards $\mathcal{V}(A')$ for A', where the guards in $\mathcal{V}(A')$ may be located anywhere in T. We present a constant-factor approximation algorithm for computing a minimum-cardinality such set of guards.

We may assume that the endpoints of A are in A', because otherwise we can simply replace A with the subterrain $[a', b']$, where a' (resp., b') is the leftmost (resp., rightmost) vertex in A', and any (Case 2) solution to A is also a (Case 2) solution to $[a', b']$ and vice versa. In particular we may assume that a and b are not vertices of the convex hull of the initial terrain (since A is contained in a subterrain defined by two consecutive vertices v_1, v_2 of the convex hull of the initial terrain, and we already placed guards at these vertices).

We will need the following observation.

Observation 4.13 *For any vertex $v \in A$, the vertices $L(v)$ and $R(v)$ can see each other.*

(If there were a vertex v for which $L(v)$ and $R(v)$ cannot see each other, then the upper angle formed by $L(v)$, v, and $R(v)$ would be greater than π, and v would be a vertex of the convex hull of the original terrain (in between v_1 and v_2), contradicting our assumptions above.)

A vertex $m \in A$ is called a *shared vertex* if m can be seen both from the left of A and from the right of A. For a shared vertex m, let $A'_l(m)$ (resp., $A'_r(m)$) denote the subset of vertices of A' that lie to the left (resp., to the right) of m and are not visible from $L(m)$, m, or $R(m)$. As in Case 1a, we say that m implies a *non-trivial division* if both $A'_l(m)$ and $A'_r(m)$ are not empty.

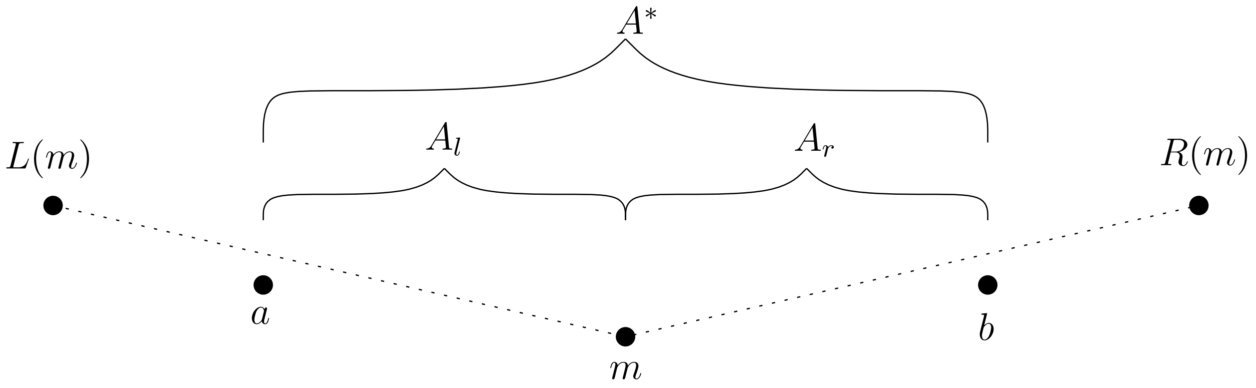

Figure 6: The two subterrains defined by a shared vertex that implies a non-trivial division are guard independent.

Claim 4.14 *Let m be a shared vertex that implies a non-trivial division. Then the two subterrains $A_l = [a, m)$ and $A_r = (m, b]$ are guard independent.*

Proof: Any guard that can see a vertex in $A'_l(m)$ must lie either between $L(m)$ and m, or to the right of $R(m)$, while any guard that can see a vertex in $A'_r(m)$ must lie either to the left of $L(m)$, or between m and $R(m)$. See Figure 6. □

Claim 4.15 *There is at least one shared vertex in $A = [a, b]$.*

Proof: a is surely seen from the left of A (e.g., by the vertex immediately to its left), and b is surely seen from the right of A. Let c be the rightmost vertex in A that is seen from the left of A. (If $c = b$, then let c be the previous vertex in A that is seen from the left of A.) Let d be the vertex immediately to the right of c. Then d is necessarily seen from the right of A. We observe that either $L(c)$ or $R(d)$ must lie above the line through c and d. (Otherwise, c and d are vertices of the convex hull of the original terrain — impossible, see assumptions and observation above.) Thus, either c or d is a shared vertex; e.g., if $L(c)$ lies above the line through c and d it must also see d, so d is a shared vertex in this case. □

Let M be the set of all shared vertices in A. According to the claim above $M \neq \emptyset$. We next show that any subterrain of A without shared vertices (i.e., vertices of M) can be nicely divided into two subterrains.

Claim 4.16 *Let $A^* = [l, r] \subset A$ be a subterrain such that $A^* \cap M = \emptyset$. Let u be a vertex in A^*. If u can be seen from the left of A, then any vertex in A^* to the right of u can also be seen from the left of A (and therefore cannot be seen from the right of A).*

Proof: Since A does not require a local guard, any vertex in A^* can be seen either from the left of A or from the right of A, but since $A^* \cap M = \emptyset$, it can be seen from only one of these sides. Assume there is a vertex to the right of u that is seen from the right of A. Let v be the leftmost such vertex, and let u' be the vertex immediately to the left of v. Then u' (which is possibly u) is seen from the left of A. Observe that either $L(u')$ or $R(v)$ must lie above the line through u' and v (since otherwise u' and v would be vertices of the convex hull of the original terrain), and therefore at least one of the two vertices u', v is a shared vertex — contradicting our assumption. We conclude that there is no vertex to the right of u that is seen from the right of A.

Now let d be the leftmost vertex in A^* that can be seen from the left of A. If there is no such d, then every vertex in A^* is seen from the right of A but not from the left of A, and if $d = l$, then any vertex in A^* is seen from the left of A but not from the right of A. Otherwise, let c be the vertex immediately to the left of d. Then every vertex in the subterrain $A_l = [l, c]$ is seen from the right of A but not from the left of A, and every vertex in the subterrain $A_r = [d, r]$ is seen from the left of A but not from the right of A. $\square$

Claim 4.17 *Assume that for each $m \in M$, m implies a trivial division. Then, by placing a constant number of guards, one can reduce A either to a single instance of Case 1, or to two guard-independent instances of Case 1.*

Proof: Let $m \in M$ be a shared vertex (such a vertex exists as shown in Claim 4.15). We know that m implies a trivial division. Assume, e.g., that (after placing guards at $L(m)$, m, and $R(m)$) A_l is fully guarded but A_r is not. Let m_1 be the rightmost shared vertex for which A_l is fully guarded. We now distinguish between two possible situations: (i) there exists another shared vertex to the right of m_1, and (ii) m_1 is the rightmost shared vertex. In both cases the subterrain $[a, m_1]$ is fully guarded by $L(m_1)$, m_1 and $R(m_1)$. Consider the former more general situation. Let m_2 be the shared vertex immediately to the right of m_1. m_2 implies a trivial division such that A_r is fully guarded (after placing guards at $L(m_2)$, m_2 and $R(m_2)$). Put $A^* = (m_1, m_2)$. A^* does not have a shared vertex, and $[m_2, b]$ is fully guarded by $L(m_2)$, m_2 and $R(m_2)$. We now use Claim 4.16 to divide (m_1, m_2) into two subterrains $A_l = (m_1, c], A_r = [d, m_2)$ (one of them might be empty), such that any vertex in A_l is seen from the right of A but not from the left of A, and any vertex in A_r is seen from the left of A but not from the right of A. Observe that $L(m_1)$ and m_1 dominate the visibility of any vertex in $[a, m_1)$ with respect to A_l, so none of the vertices to the left of A_l can help in guarding the remaining unguarded vertices in A_l. Similarly, $R(m_2)$ and m_2 dominate the visibility of any vertex in $(m_2, b]$ with respect to A_r. Also if we place guards at c and $R(c)$ and at d and $L(d)$ then the two subterrains A_l and A_r are guard independent and both can be treated as Case 1 problems (see Figure 7).

Consider the second situation (i.e., there is no shared vertex in $A_r = (m_1, b]$). We apply Claim 4.15 to observe that no vertex in A_r can be seen from left of A (b can be seen from right of A and therefore any vertex in A_r can be seen from the right of A, and if there were a vertex that can also be seen from the left of A then we would have a shared vertex). Thus, locating guards at $L(m_1)$, m_1 and $R(m_1)$ guarantees that the only vertices that can help guarding A_r are within A_r

or to the right of A_r which is a Case 1 problem. (Once again none of the vertices in $[a, m_1)$ can help guard A_r, since $L(m_1)$ and m_1 dominate these vertices with respect to A_r.) □

Algorithm 4.18 (Case 2)

1. *If there exist two (or any constant number of) guards that together see all vertices in A' — done.*
2. *Compute the set M of all shared vertices.*
3. *If there exists $m \in M$ that implies a non-trivial division, then locate guards at $L(m)$, m, and $R(m)$ and solve (recursively) each of the two subproblems A_l and A_r.*
4. *Else, for each $m \in M$ one of the sides is fully guarded by $L(m)$, m, and $R(m)$ and the other is not. Use Claim 4.17 in order to reduce A (by placing a constant number of guards) either to a single instance of Case 1, or to two guard-independent instances of Case 1. (Notice that if A is reduced to a single instance of Case 1, we charge the guards that were placed in doing so to the one time event of leaving Case 2.)*

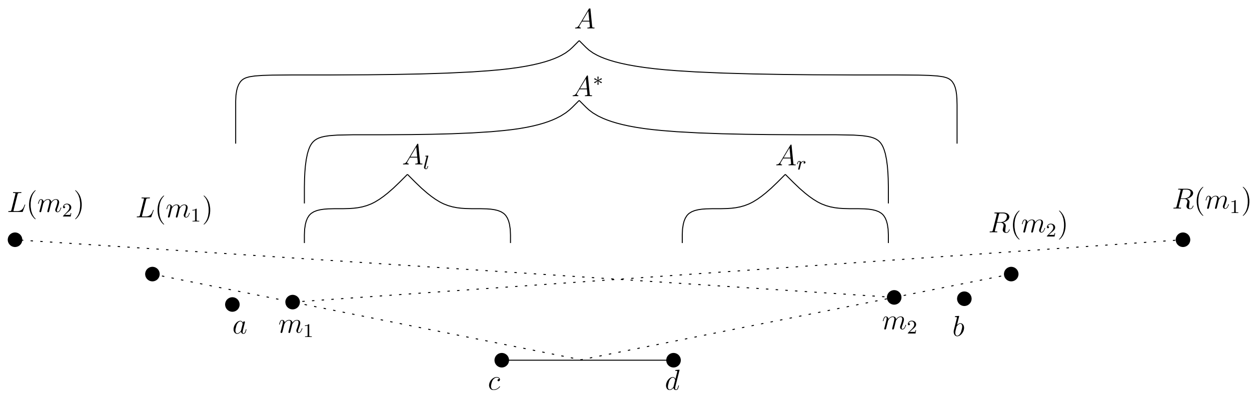

Figure 7: By locating guards at $L(m_1)$, m_1, m_2 and $R(m_2)$ we obtain two guard independent subproblems, A_l and A_r, both of Case 1.

The following lemma summarizes the result for Case 2.

Lemma 4.19 *The Case 2 algorithm above computes a set of guards $\mathcal{V}(A')$ for A', whose size is bounded by some constant times the size of a minimum-cardinality such set of guards.*

5 Algorithm Analysis

Throughout our description of the approximation algorithm for the dominating set problem on terrains (DSPT), whenever guards were placed we gave an appropriate charging argument to justify why we could afford to place them. Consequently, we have shown that the size of the guarding set that is computed by the algorithm is bounded by a constant times the size of a minimum-size such set: we have obtained an $O(1)$-approximation, as desired. We have not attempted to minimize the constant factor; we leave this task to future work.

Concerning the running time of the algorithm, it is clear that it is polynomial. The following lemma shows that the running time is $O(n^2)$.

Lemma 5.1 *The running time of the constant-factor approximation algorithm for the dominating set problem on terrains (DSPT) is $O(n^2)$.*

Proof: As a preliminary stage, we compute the *visibility graph* $\mathrm{V}G_T(V)$ of the terrain vertices. This can be done easily in $O(n^2)$ time (or in output-sensitive time, but this does not yield an improved overall time bound for our algorithm). For each vertex $v \in V$, we compute in linear time the subset of vertices in V that lie to the left (alternatively, to the right) of v and are visible from v; the vertices in these subsets are found one by one in decreasing (alternatively, increasing) x-order. In particular the vertices $L(v)$ and $R(v)$ are the leftmost in the left list and rightmost in the right list, respectively. It remains to show that the subsequent stages of the algorithm can be carried out within the quadratic time bound.

In the first stage, we simply compute the (upper) convex hull of T in $O(n)$ time. In the second stage, we partition each "concave" subterrain (corresponding to an edge of the convex hull) into subterrains that do not require a *local guard* (see Section 3). This can be done in $O(m \log m)$ time per "concave" subterrain of m vertices, and therefore in overall $O(n \log n)$ time.

In the final stage, each of the subterrains obtained in the second stage is solved separately, using the base-case algorithms, in $O(m^2)$ time, where m is the size of the subterrain (see Lemma 5.2 below). Since the subterrains are disjoint, the overall running time of this stage is $O(n^2)$. $\square$

Lemma 5.2 *Let A, $A \subset T$, be a subterrain that does not require a* local guard. *Then A can be solved using the base-case algorithms (with minor modifications) in $O(m^2)$ time, where m is the size of A.*

Proof: For the lemma to be true, we need to modify the first step in each of the base-case algorithms, so that, unless all vertices in A' are already guarded, one must continue by recursive calls. This change increases the depth of the recursion by only a constant number of levels.

Case 2: The condition in the modified Step #1 can be checked in constant time. In Step #2 all *shared vertices* can be computed in linear time, since one only needs to consider the *ranges* of the vertices in A (which were computed in the preliminary stage of the main algorithm). Checking in Step #3 whether a *non-trivial division* is possible is also done in linear time, since for each shared vertex one only needs to consider a constant number of vertices. Moreover, the number of such divisions is clearly $O(m)$. In Step #4 we reduce the problem to two guard-independent instances of Case 1; the reduction is done in constant time.

Case 1b: Step #2 is clearly linear in m. In Steps #3 and #4 we reduce the problem to a smaller one, therefore the number of these reductions in $O(m)$.

Case 1a: Computing all vertices that imply a non-trivial division (in Step #2) is done in linear time (since, for each vertex $q \in A$, one only needs to consider a constant number of vertices). In Step #3 we reduce the problem to two guard-independent and disjoint problems, so the number of these reductions is $O(m)$; the reduction itself takes only constant time. In Step #4 we reduce the problem to a smaller one. Again, the number of these reductions is clearly $O(m)$, and the reduction itself takes only constant time.

□

The following theorem summarizes this section.

Theorem 5.3 *The algorithm of Sections 3 and 4 computes a constant-factor approximation for the dominating set problem on terrains (DSPT) in $O(n^2)$ time.*

6 The Terrain Guarding Algorithm

In this section we generalize the constant-factor approximation algorithm for the dominating set problem on terrains (DSPT) to the general 1.5D terrain guarding problem, where guards can be placed anywhere on the terrain and all points of the terrain (not only its vertices) must be guarded. For this we present a reduction from the general 1.5D terrain guarding problem to DSPT. Figure 8 (right) shows that a solution to DSPT is not necessarily a solution to the general 1.5D terrain guarding problem.

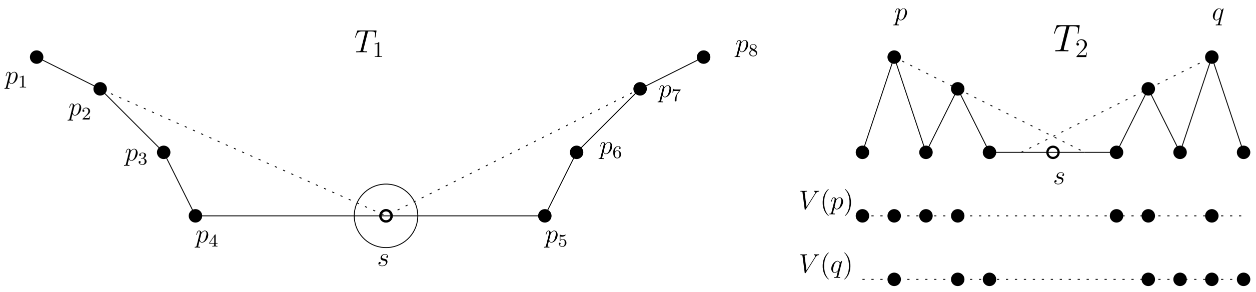

Figure 8: Left: a single guard at s sees all points of the terrain T_1, but if one may locate guards only at vertices, then two guards are needed. Right: all vertices of the terrain T_2 can be guarded with two guards (at p and at q), but in order to guard all points of T_2 three guards are needed.

Let V be the set of vertices of T and put $n = |V|$.

Observation 6.1 *Any solution to the general 1.5D terrain guarding problem can be transformed into another solution whose size is at most twice the size of the original solution, such that all guards in the new solution are vertex guards, i.e., located at vertices of the terrain.*

Proof: Replace each guard g in the original solution that is not a vertex of the terrain, with the two endpoints v_i, v_{i+1} of the edge T on which g lies. Clearly, any point of T that is seen by g is also seen by at least one of these two vertices. □

Lemma 6.2 *There exists a set U of points on T, such that every subset of vertices $V' \subseteq V$ that guards U also guards T. Moreover $|U| = O(n^2)$ and U can be computed in $O(n^2)$ time.*

Proof: Let p be a point on T and let $\mathrm{vis}(p)$ denote the set of all points on T that are visible from p. $\mathrm{vis}(p)$ is the union of a linear number of maximal subterrains. For such a subterrain A, it is easy to see that if $p \in A$, then both endpoints of A must be vertices of T, and, if $p \notin A$, then the farther of the two endpoints of A must be a vertex of T. We refer to the endpoints of the maximal subterrains in $\mathrm{vis}(p)$ as the *visibility events* induced by p.

Let U' be the set of visibility events induced by the vertices in V. Let U'' be the set that is obtained by picking an arbitrary point of T between each pair of consecutive points in U'. Put $U = V \cup U' \cup U''$. It is clear that $|U| = O(n^2)$ and that U can be computed in $O(n^2)$ time.

The set U' induces a partition of T into $O(n^2)$ intervals. Observe that if p is a point in the interior of an interval s of this partition and p' is the point of U'' that lies in the interior of s, then $\mathrm{vis}(p) \cap V = \mathrm{vis}(p') \cap V$. This follows immediately from the definition of U', since if there were a vertex $v \in V$ such that p sees v and p' does not see v (or vice versa), then there would be a visibility event somewhere in the interior of s.

Now let V' be a subset of V that guards U and let p be any point on T, $p \notin U$. We need to show that p is guarded by V'. Let s be the interval of the partition of T induced by U' such that p lies in its interior, and let p' be the point of U'' that lies in the interior of s. Then by our assumption p' is guarded by V', and, therefore, by the observation above, so is p. $\square$

We are now ready to present the algorithm for the general 1.5D terrain guarding problem.

Algorithm 6.3 General 1.5D terrain guarding

1. *Given a terrain T with vertex set V, compute the set U.*
2. *Solve DSPT with U as the vertex set (using the DSPT algorithm as a 'black box'). Let $\mathcal{G}'$, $\mathcal{G}' \subseteq U$, denote the solution obtained.*
3. *Replace each point $g \in \mathcal{G}' - V$ with the two vertices of V adjacent to it. Let $\mathcal{G}$ be the resulting set.*
4. *Return $\mathcal{G}$.*

From Observation 6.1 and Lemma 6.2 it is clear that $\mathcal{G}$ is indeed a solution to the general 1.5D terrain guarding problem. Moreover, $\mathcal{G}$ can be computed in $O(n^4)$ time (since $|U| = O(n^2)$ and the running time of the DSPT algorithm is $O(|U|^2)$). It remains to prove that the size of $\mathcal{G}$ is bounded by some constant times the size of an optimal solution.

Let $opt(A, B)$ be an optimal solution to the problem of guarding A by placing guards at points of B, where A (alternatively, B) is either a subterrain of T or a discrete set of points of T. In particular, $opt(T, T)$ is an optimal solution to the general 1.5D terrain guarding problem. Notice that if $A' \subseteq A$ and $B' \subseteq B$, then $|opt(A, B)| \geq |opt(A', B)|$ and $|opt(A, B')| \geq |opt(A, B)|$.

Lemma 6.4 *Let c' be the (constant) approximation factor of the DSPT algorithm. Then $|\mathcal{G}| \leq 4c'|opt(T, T)|$.*

Proof: Using the observation above and Observation 6.1, $|opt(T, T)| \geq |opt(U, T)| \geq |opt(U, V)|/2 \geq |opt(U, U)|/2 \geq \mathcal{G}'/(2c') \geq \mathcal{G}/(4c')$. $\square$

The following theorem summarizes the main result of this section.

Theorem 6.5 *Algorithm 6.3 computes a constant factor approximation for the general 1.5D terrain guarding problem in $O(n^4)$ time.*

7 Conclusion

There are two notable open problems: (1) Are the DSPT and 1.5D terrain guarding problems NP-hard? (the hardness claim in [4] has gaps in the proof); and (2) Is there an approximation algorithm for guarding a simple polygon? So far, our attempts to generalize our methods to simple polygons have failed; our method strongly exploits the special structure of 1.5D terrains.

Acknowledgements

M. Katz and J. Mitchell are partially supported by grant No. 2000160 from the U.S.-Israel Binational Science Foundation. M. Katz and B. Ben-Moshe are partially supported by the MAGNET program of the Israel Ministry of Industry and Trade (LSRT consortium). J. Mitchell acknowledges support from Honda Fundamental Research Lab, NASA Ames Research (NAG2-1620), the National Science Foundation (CCR-0098172, ACI-0328930), and Metron Aviation.

References

[1] B. Ben-Moshe, M. J. Katz and J. S. B. Mitchell. A constant-factor approximation algorithm for optimal terrain guarding. In *Proc. 16th ACM-SIAM Sympos. Discrete Algorithms*, 2005, 515–524.

[2] P. Bose, T. Shermer, G. Toussaint, and B. Zhu. Guarding polyhedral terrains. *Comput. Geom. Theory Appl.*, 7:173–185, 1997.

[3] H. Brönnimann and M. T. Goodrich. Almost optimal set covers in finite VC-dimension. *Discrete Comput. Geom.*, 14:263–279, 1995.

[4] D. Z. Chen, V. Estivill-Castro, and J. Urrutia. Optimal guarding of polygons and monotone chains. In *Proc. 7th Canad. Conf. Comput. Geom.*, pages 133–138, 1995.

[5] O. Cheong, A. Efrat, and S. Har-Peled. On finding a guard that sees most and a shop that sells most. In *Proc. 15th ACM-SIAM Sympos. Discrete Algorithms*, pages 1098–1107, 2004.

[6] A. Efrat and S. Har-Peled. Locating guards in art galleries. In *2nd IFIP International Conference on Theoretical Computer Science*, 2002.

[7] S. J. Eidenbenz. *(In-)Approximability of Visibility Problems on Polygons and Terrains*. Ph.D. thesis, ETH Zürich, Switzerland, 2000.

[8] S. J. Eidenbenz. Approximation algorithms for terrain guarding. *Inform. Process. Lett.*, 82:99–105, 2002.

[9] S. J. Eidenbenz, C. Stamm, and P. Widmayer. Inapproximability results for guarding polygons and terrains. *Algorithmica*, 31:79–113, 2001.

[10] S. K. Ghosh. Approximation algorithms for art gallery problems. In *Proc. Canadian Inform. Process. Soc. Congress*, pages 429–434, 1987.

[11] H. González-Banos and J.-C. Latombe. A randomized art-gallery algorithm for sensor placement. In *Proc. 17th Annual Symposium on Computational Geometry*, pages 232–240, 2001.

[12] J. M. Keil. Polygon decomposition. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 491–518. Elsevier Science Publishers B.V. North-Holland, Amsterdam, 2000.

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

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

[15] J. O'Rourke and K. J. Supowit. Some NP-hard polygon decomposition problems. *IEEE Trans. Inform. Theory*, IT-30:181–190, 1983.

[16] T. C. Shermer. Recent results in art galleries. *Proc. IEEE*, 80(9):1384–1399, Sept. 1992.

[17] J. Urrutia. Art gallery and illumination problems. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 973–1027. North-Holland, 2000.