Exact algorithms for terrain guarding

Citation for published version (APA):

DOI:
10.1145/3186897

Document status and date:
Published: 01/06/2018

Document Version:
Accepted manuscript including changes made at the peer-review stage

Please check the document version of this publication:

• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.
Exact Algorithms for Terrain Guarding

PRADEE莎H ASHOK, International Institute of Information Technology Bangalore, India
FEDOR V. FOMIN, University of Bergen, Norway
SUDESHNA KOLAY, Eindhoven University of Technology, Netherlands, Netherlands
SAKET SAURABH, University of Bergen, Norway and Institute of Mathematical Sciences, India
MEIRAV ZEHAVI, Ben-Gurion University, Norway

Given a 1.5-dimensional terrain $T$, also known as an $x$-monotone polygonal chain, the Terrain Guarding problem seeks a set of points of minimum size on $T$ that guards all of the points on $T$. Here, we say that a point $p$ guards a point $q$ if no point of the line segment $[pq]$ is strictly below $T$. The Terrain Guarding problem has been extensively studied for over 20 years. In 2005 it was already established that this problem admits a constant-factor approximation algorithm [SODA 2005]. However, only in 2010 King and Krohn [SODA 2010] finally showed that Terrain Guarding is NP-hard. In spite of the remarkable developments in approximation algorithms for Terrain Guarding, next to nothing is known about its parameterized complexity. In particular, the most intriguing open questions in this direction ask whether, if parameterized by the size $k$ of a solution guard set, it admits a subexponential-time algorithm and whether it is fixed-parameter tractable. In this paper, we answer the first question affirmatively by developing an $O(n^{O(\sqrt{k})})$-time algorithm for both Discrete Terrain Guarding and Continuous Terrain Guarding. We also make non-trivial progress with respect to the second question: we show that Discrete Orthogonal Terrain Guarding, a well-studied special case of Terrain Guarding, is fixed-parameter tractable.

CCS Concepts:
• Theory of computation → Fixed parameter tractability;

Additional Key Words and Phrases: Terrain Guarding, Art Gallery, Exponential-Time Algorithms

ACM Reference Format:

1 INTRODUCTION

The study of terrains, also known as $x$-monotone polygonal chains, has attracted widespread and growing interest over the last few decades in the field of Discrete Computational Geometry. A terrain is a graph $T = (V, E)$ where each vertex $v_i \in V$, $1 \leq i \leq n$, is associated with a point...
(x_i, y_i) on the two-dimensional Euclidean plane such that x_1 < x_2 < \ldots < x_n, and the edge-set is E = \{\{v_i, v_{i+1}\} : 1 \leq i \leq n - 1\}. The points lying on the terrain could be the vertices v_i, 1 \leq i \leq n, or the points that lie on the edge set E. In the Terrain Guarding problem, the input is a terrain and a positive integer k, and the task is to decide whether one can place guards on at most k points on a given terrain such that each point on the terrain is seen by at least one guard. Here, we say that a point p sees a point q if no point of the line segment \overline{pq} is strictly below T. This problem is also known as the Continuous Terrain Guarding problem or Terrain Guarding in a continuous domain, since we are allowed to place guards on the continuous domain of the given terrain, and we want to cover all points lying on the terrain. Another variant of this problem is the Discrete Terrain Guarding problem, where the input consists of a terrain T = (V, E) on n vertices and a positive integer k \leq n, and the objective is to determine whether there is a subset S \subseteq V of size at most k that sees V. This problem will also be referred to as Terrain Guarding in a discrete domain.

Both discrete and continuous variants of the Terrain Guarding problem arise in a wide-variety of applications relevant to the design of various communication technologies such as cellular telephony and line-of-sight transmission networks for radio broadcasting. It also arises in applications of coverage of highways, streets and walls with street lights or security cameras [3, 15].

The visibility graphs of terrains exhibit unique properties which render the complexity of the Terrain Guarding problem difficult to elucidate. Some of these properties have already been observed in 1995 by Abello et al. [1], and some of them remain unknown despite recent advances to identify them [14]. Indeed, the Terrain Guarding problem has been extensively studied since 1995, when an NP-hardness proof was claimed but never completed by Chen et al. [6]. Almost 15 years later King and Krohn [24] finally showed that this problem is NP-hard.

Particular attention has been given to the Terrain Guarding problem from the viewpoint of approximation algorithms. In 2005, Ben-Moshe et al. [3] obtained the first constant-factor approximation algorithm for Discrete Terrain Guarding. Afterward, the approximation factor was gradually improved in [7, 13, 23], until a PTAS was proposed by Gibson et al. [18] for Discrete Terrain Guarding. Recently, Friedrichs et al. [15] showed that even the Continuous Terrain Guarding problem admits a PTAS.

The Terrain Guarding problem has also gained interest due to its deceptive resemblance to the Art Gallery problem, where instead of a terrain, it is necessary to guard a polygon. The Art Gallery problem was introduced by Klee in 1973 [28], and it is arguably one of the most well-known problems in Discrete Computational Geometry. For more information on the Art Gallery problem, we refer to the books dedicated to its study [16, 28, 30]. Note that the Art Gallery problem does not admit a subexponential-time algorithm. Indeed, the known NP-hardness reduction for the Art Gallery problem, even when restricted to orthogonal polygons, reduces a 3-SAT instance on n variables and m clauses to an instance of Art Gallery with O(n + m) vertices [28, 29]. This reduction combined with the Exponential Time Hypothesis (ETH) [8, 20] implies the following result.

**Corollary 1 (Folklore).** Unless ETH fails, there is no algorithm for Art Gallery, even when restricted to orthogonal polygons, that achieves running time of 2^{o(n)}. That is, the Art Gallery problem does not admit a subexponential-time algorithm.

In the parameterized setting, where n is the number of vertices in the polygon and k is the number of guards, clearly one can design an algorithm for the Art Gallery problem running in time n^{O(k)} by enumerating all subsets of vertices of size at most k. Interestingly, by the very recent result of Bonnet and Miltzow [5] this trivial brute-force algorithm is essentially optimal. More precisely, they proved that an algorithm solving Art Gallery in time f(k) \cdot n^{o(k/\log k)} for any function f would
Exact Algorithms for Terrain Guarding

imply that the ETH fails. The reduction given in [5] also implies that Art Gallery is W[1]-hard parameterized by \( k \). Thus it is highly unlikely that Art Gallery is fixed-parameter tractable (FPT). On the other hand, the Art Gallery problem on monotone polygons seems to be an easier variant to solve. As evidence, the monotone variant has a constant factor approximation [25] while no such result is known for Art Gallery. Since terrains are also defined by monotone chains, this gives us a hint that the Terrain Guarding problem might yield better algorithms than Art Gallery. We should point out that, to the best of our knowledge, no subexponential or FPT algorithms are known for Art Gallery in monotone polygons.

Orthogonal Terrain Guarding is a problem of independent interest that is a special case of Terrain Guarding. In this problem, the terrain is orthogonal: for each vertex \( v_i, 2 \leq i \leq n-1 \), either both \( x_{i-1} = x_i \) and \( y_i = y_{i+1} \) or both \( y_{i-1} = y_i \) and \( x_i = x_{i+1} \). In other words, each edge is either a horizontal line segment or a vertical line segment, and each vertex is incident to at most one horizontal edge and at most one vertical edge. The Orthogonal Terrain Guarding problem has already been studied from the perspective of algorithms theory [12, 21, 26, 27]. Although the PTASes designed in [18] or [15] work for the Orthogonal Terrain Guarding problem as well, there are a few studies on this particular variant of Terrain Guarding, that bring out interesting structural properties specific to this variant. A prime example is the work of Katz and Roisman [21], where they gave a relatively simple 2-approximation algorithm for the problem of guarding all vertices of an orthogonal terrain by vertices. Recently, Lyu and Üngör improved upon this result by developing a linear-time 2-approximation algorithm for Orthogonal Terrain Guarding. The papers [27] and [12] studied restrictions under which Orthogonal Terrain Guarding can be solved in polynomial time.

While by now we have quite satisfactory understanding of the approximability of Terrain Guarding, the parameterized hardness of this problem is unknown. Currently, the most fundamental open questions regarding the complexity of the Terrain Guarding problem are the following:

- Does Terrain Guarding admit a subexponential-time algorithm?
- Is Terrain Guarding FPT with respect to \( k \)?

Indeed, King and Krohn [24] state that “the biggest remaining question regarding the complexity of Terrain Guarding is whether or not it is FPT”. Moreover, interest in the design of efficient, exact exponential-time algorithms for this problem has been expressed at workshops such as the Lorentz Workshop on Fixed-Parameter Computational Geometry [17]. To the best of our knowledge, the only work which is somewhat related to the second question is the one by Khodakarami et al. [22], who introduced the parameter “the depth of the onion peeling of a terrain” and showed that Terrain Guarding is FPT with respect to this parameter.

In this paper, we address both of these questions. First, we completely resolve the first question by designing a subexponential-time algorithm for Terrain Guarding in both discrete and continuous domains. For this purpose, we develop an \( n^{O(\sqrt{k})} \)-time algorithm for Terrain Guarding in discrete domains. Friedrichs et al. [15] proved that given an instance of Terrain Guarding in a continuous domain, one can construct (in polynomial time) an equivalent instance of Terrain Guarding in a discrete domain. More precisely, given an instance \( (T = (V, E), k) \) of Terrain Guarding in a continuous domain, Friedrichs et al. [15] designed a discretization procedure that outputs an instance \( (T' = (V', E'), k) \) of Terrain Guarding in a discrete domain such that \( (T = (V, E), k) \) is a yes-instance if and only if \( (T' = (V', E'), k) \) is a yes-instance. Unfortunately, this reduction blows up the number of vertices of the terrain to \( O(n^3) \), and therefore the existence of a subexponential-time algorithm for Terrain Guarding in discrete domains does not imply that there exists such an algorithm for Terrain Guarding in continuous domains. However, observe that the reduction
does not change the value of the parameter $k$. Thus, since we solve Terrain Guarding in discrete domains in time $n^{O(\sqrt{k})}$ rather than $n^{O(\sqrt{n})}$, we are able to deduce that Terrain Guarding in continuous domains is solvable in time $n^{O(\sqrt{k})}$. Observe that, in both discrete and continuous domains, it can be assumed that $k \leq n$: to guard all of the points that lie on a terrain, it is sufficient to place guards only on the vertices of the terrain. Hence, when we solve Terrain Guarding in continuous domains, we assume that $k \leq n$ where $n$ is the number of vertices of the input continuous terrain and not of the discrete terrain output by the reduction. The next theorem summarizes our algorithmic contribution.

Theorem 1.1. Terrain Guarding in both discrete and continuous domains is solvable in time $n^{O(\sqrt{k})}$. Thus, it is also solvable in time $n^{O(\sqrt{n})}$.

Observe that our result, Theorem 1.1, demonstrates an interesting dichotomy in the complexities of Terrain Guarding and the Art Gallery problem: Corollary 1 implies that the Art Gallery problem does not admit an algorithm with running time $2^{o(n)}$, while Terrain Guarding in both discrete and continuous domains is solvable in time $2^{O(\sqrt{n \log n})}$. When we measure the running time in terms of both $n$ and $k$, the Art Gallery problem does not admit an algorithm with running time $f(k) \cdot n^{o(k/\log k)}$ for any function $f$ [5], while Terrain Guarding in both discrete and continuous domains is solvable in time $n^{O(\sqrt{k})}$.

Our algorithm is based on a suitable definition of a planar graph. A notion of a planar graph was recently used in designing a local search based PTAS for the Terrain Guarding problem [18]. We were inspired by this construction and tried to use an appropriate notion of a planar graph in our settings. Towards this we define a planar graph that has a small domination number and which captures both the manner in which a hypothetical solution guards the terrain and some information on the layout of the terrain itself. Having this planar graph, we are able to “guess” separators whose exploitation, which involves additional guesses guided by the structure of the graph, essentially results in a divide-and-conquer algorithm. The design of the divide-and-conquer algorithm is also nontrivial since given our guesses, it is not possible to divide the problem into two simpler subproblems in the obvious way—that is, we cannot divide the terrain into two disjoint subterrains that can be handled separately. We overcome this difficulty by dividing not the terrain itself, but a set of points of interest on the terrain.

We also shed light on the second question by showing that Orthogonal Terrain Guarding of vertices of the orthogonal terrain with vertices is FPT with respect to the parameter $k$. More precisely, we obtain the following result.

Theorem 1.2. Orthogonal Terrain Guarding of vertices of the terrain with vertices is solvable in time $k^{O(k)} \cdot n^{O(1)}$.

Our algorithm is based on new insights into the structure of orthogonal terrains, particularly into the relations between their left and right reflex and convex vertices. We integrate these insights in the design of an algorithm that is based on the proof that one can ignore “exposed vertices”, which are vertices seen by too many vertices of a specific type, greedy localization, and a non-trivial branching strategy. We conclude the introduction by posing the following open problems: Are Terrain Guarding and Orthogonal Terrain Guarding in continuous domains FPT?

2 PRELIMINARIES

For a positive integer $k$, we use $[k]$ as a shorthand for $\{1, 2, \ldots, k\}$.

Graphs. We use standard notation and terminology from the book of Diestel [10] for graph-related terms which are not explicitly defined here. We only consider simple undirected graphs. Given a
Exact Algorithms for Terrain Guarding

Let $H$, $V(H)$, and $E(H)$ denote its vertex-set and edge-set, respectively. Given a subset $U \subseteq V(H)$, the subgraph of $H$ induced by $U$ is denoted by $H[U]$. A dominating set of $H$ is a subset $S \subseteq V(H)$ such that each vertex in $V(H)$ either belongs to $S$ or has a neighbor in $S$. The domination number of $H$, denoted by $\gamma(H)$, is the minimum size of a dominating set of $H$. A clique cover of $H$ is a partition $(V_i, V_2, \ldots, V_t)$ of $V(H)$ for some $t \in \mathbb{N}$ such that for any $i \in [t]$, $H[V_i]$ is a clique. The size of the clique cover is $t$. The clique cover number of $H$, denoted by $\kappa(H)$, is the minimum size of a clique cover of $H$. An independent set of $H$ is a subset $U \subseteq V(H)$ such that there do not exist two vertices in $U$ that are neighbors in $H$. The independence number of $H$, denoted by $\alpha(H)$, is the maximum size of an independent set of $H$. Given a graph $G$, a cycle $C = \{v_1, v_2, \ldots, v_n, v_{n+1} = v_1\}$ is called an induced cycle if the induced subgraph on the vertex set $V(C)$ is the cycle $C$ itself. A chordal graph is a graph that has no induced cycle on more than three vertices. In the context of chordal graphs, we will need to rely on the following well-known results.

**Theorem 2.1** ([19]). Let $H$ be a chordal graph. Then
- A clique cover of $H$ of minimum size can be found in linear time.
- An independent set of $H$ of maximum size can be found in linear time.
- $\kappa(H) = \alpha(H)$.

**Terrains.** A 1.5-dimensional terrain $T = (V, E)$, or terrain for short, is a graph on vertex-set $V = \{v_1, v_2, \ldots, v_n\}$ where each vertex $v_i$ is associated with a point $(x_i, y_i)$ on the two-dimensional Euclidean plane such that $x_1 < x_2 < \ldots < x_n$ and the edge-set is $E = \{(v_i, v_{i+1}) : i \in [n-1]\}$. We say that a point $p$ sees a point $q$ if every point of the line segment $pq$ is either on or above $T$. Note that if a point $p$ sees a point $q$, then the point $q$ sees the point $p$ as well. More generally, we say that a set of points $P$ sees a set of points $Q$ if each point in $Q$ is seen by at least one point in $P$.

An orthogonal terrain, also known as a rectilinear terrain, is a terrain $T = (V, E)$ where for each vertex $v_i$, $2 \leq i \leq n - 1$, either both $x_{i-1} = x_i$ and $y_i = y_{i+1}$ or both $y_{i-1} = y_i$ and $x_i = x_{i+1}$. In other words, an orthogonal terrain is a terrain where each edge is either a horizontal line segment or vertical line segment, and each vertex is incident to at most one horizontal edge and at most one vertical edge. A vertex $v_i$, $2 \leq i \leq n - 1$ belongs to one of the following four categories: its $x_i = x_{i+1}$ and $y_i > y_{i+1}$, it is a right reflex vertex; if $x_i = x_{i+1}$ and $y_i < y_{i+1}$, it is a right convex vertex; if $x_i = x_{i-1}$ and $y_i > y_{i-1}$, it is a left reflex vertex; if $x_i = x_{i-1}$ and $y_i < y_{i-1}$, it is a left convex vertex. Moreover, if $x_1 = x_2$ and $y_1 > y_2$, $v_1$ is a right reflex vertex; if $x_1 = x_2$ and $y_1 < y_2$, it is a right convex vertex; otherwise it is a left convex vertex. Symmetrically, if $x_n = x_{n-1}$ and $y_n > y_{n-1}$, $v_n$ is a left reflex vertex; if $x_n = x_{n-1}$ and $y_n < y_{n-1}$, it is a left convex vertex; otherwise it is a right convex vertex. We also say that a vertex is a reflex vertex if it is either a left reflex vertex or a right reflex vertex, and otherwise it is a convex vertex. Furthermore, we say that left reflex/convex vertices are opposite to right reflex/convex vertices. An illustrative example of these notions is given in Fig. 1.

Let $T = (V, E)$ be a terrain and let $U$ be a subset of $V$. We use $\text{vis}(U)$ to denote the set containing every vertex in $V$ that is seen by at least one vertex in $U$. In case $U = \{u\}$, we abuse notation and write $\text{vis}(u)$ to refer to $\text{vis}(U)$. We use $\text{cut}(U)$ to denote the set of (maximal) subterrains of $T$ that result from the removal of the vertices in $U$. That is, $\text{cut}(U)$ is the set of each subterrain $T' = (V', E')$ for which there exist $i < j$ such that $V' = \{v_i, v_{i+1}, \ldots, v_j\} \subseteq V \setminus U$, either $i = 1$ or $v_{i-1} \in U$, and either $j = n$ or $v_{j+1} \in U$. An illustrative example of this notation is given in Fig. 2. Given a subset $X \subseteq V$ and subterrain $T' = (V', E')$, we define $X[T'] = X \cap V'$. Moreover, given a set of terrains $T$, we let $X[T]$ be set of vertices that is the union of the sets in $\{X[T'] : T' \in T\}$.

**Terrain Guarding Problems.** For the sake of clarity, we would like to state the several variants of Terrain Guarding that we consider in this paper. The decision version of the (Discrete) Terrain
The Terrain Guarding problem is also defined in the context of continuous domains, in which case it is called the Continuous Terrain Guarding problem. The input for the Continuous Terrain Guarding problem is the same as the input for the Discrete Terrain Guarding problem. We say that a point lies on the terrain \( T \) if it is either a vertex in \( V \) or a point on an edge between
two adjacent vertices. The objective is to determine whether there is a subset of points of size at most $k$ that lie on $T$ and which see every point that lies on $T$.

To develop our algorithms for Discrete Terrain Guarding, it will be more convenient to solve a problem generalizing Discrete Terrain Guarding, that we call Annotated Terrain Guarding. Roughly speaking, Annotated Terrain Guarding is the variant of Discrete Terrain Guarding where one cannot place a “guard” on any vertex, but only on vertices from a given set $G$, and where it is not necessary to “cover” all of the vertices in $V$, but only those belonging to a given set $C$. Formally, the input consists of a terrain $T = (V, E)$ on $n$ vertices, a positive integer $k \leq n$, and subsets $G, C \subseteq V$. The objective is to determine whether there is a subset $S \subseteq G$ of size at most $k$ that sees $C$. We say that such a subset $S$ is a solution. Clearly, Discrete Terrain Guarding is the special case of Annotated Terrain Guarding where $G = C = V$. We will refer to the special case where the input terrain is an orthogonal terrain as the Annotated Orthogonal Terrain Guarding problem.

**Treewidth.** A tree decomposition of a graph $H$ is a pair $(D, \beta)$, where $D$ is a rooted tree and $\beta : V(D) \to 2^{V(H)}$ is a mapping that satisfies the following conditions.

- For each vertex $v \in V(H)$, the set $\{d \in V(D) : v \in \beta(d)\}$ induces a nonempty and connected subtree of $D$.
- For each edge $\{v, u\} \in E(H)$, there exists $d \in V(D)$ such that $\{v, u\} \subseteq \beta(d)$.

A vertex $d$ in $V(D)$ is called a node, and the set $\beta(d)$ is called the bag at $d$. We let $\text{descendants}(d)$ denote the set of descendants of $d$ in $D$. The width of $(D, \beta)$ is the size of the largest bag minus one (i.e., $\max_{d \in V(D)} |\beta(d)| - 1$). The treewidth of $H$, denoted by $\text{tw}(H)$, is the minimum width among all possible tree decompositions of $H$.

Standard arguments on trees, see e.g. [8, Lemma 7.20], imply the correctness of the following observation.

**Observation 1.** Let $(D, \beta)$ be a tree decomposition of a graph $H$ where $D$ is a binary tree, and let $S$ be a subset of $V(H)$. Then, there exists a node $d \in V(D)$ such that $|S|/3 \leq |\bigcup_{d' \in \text{descendants}(d)} \beta(d') \cap S|$ and $|\bigcup_{d' \in \text{descendants}(d) \setminus \{d\}} \beta(d') \cap S| \leq 2|S|/3$.

**Parameterized Complexity.** In Parameterized Complexity each problem instance is accompanied by a parameter $k$. A central notion in this field is the one of fixed-parameter tractability (FPT). This means, for a given instance $(I, k)$, solvability in time $f(k)|I|^{O(1)}$ where $f$ is some function of $k$. For more information on Parameterized Complexity we refer the reader to monographs such as [8, 11].

**Bit Vectors.** A $t$-length bit vector is a vector $\overline{v} = (v_1, v_2, \ldots, v_t)$ such that for any $i \in [t]$, $v_i \in \{0, 1\}$. Given two $t$-length bit vectors $\overline{v}$ and $\overline{u}$, the Hamming distance between them, denoted by $H(\overline{v}, \overline{u})$, is the number of indices $i \in [t]$ such that $v_i \neq u_i$.

### 3. SUBEXPONENTIAL ALGORITHM

Recall that the Annotated Terrain Guarding problem takes as input a terrain $T = (V, E)$ on $n$ vertices, a positive integer $k \leq n$, and subsets $G, C \subseteq V$. The objective is to determine whether there is a subset $S \subseteq G$ of size at most $k$ that sees $C$. In this section, we prove that Annotated Terrain Guarding can be solved in time $n^{O(\sqrt{n})}$. In fact, we obtain a somewhat stronger result:

**Theorem 3.1.** Annotated Terrain Guarding is solvable in time $n^{O(\sqrt{n})}$.

Since Discrete Terrain Guarding is a special case of Annotated Terrain Guarding, we derive the following result.
Corollary 2. Discrete Terrain Guarding is solvable in time $n^{O(\sqrt{k})}$.

We also derive the following result.

**Corollary 3.** Continuous Terrain Guarding is solvable in time $n^{O(\sqrt{k})}$.

Throughout this section, we let $(T = (V, E), n, k, G, C)$ denote the input instance of ANNOTATED TERRAIN GUARDING. First, in Section 3.1, we carefully define a planar graph $P_S$ that captures relations between a hypothetical solution and the set $C$. Then, in Section 3.2, we rely on properties of $P_S$ to show that there exists a partition of $G \cup C$ into two sets which rarely “alternate” along the terrain and which are both relatively “small”. In Section 3.3 we show how the existence of this partition allows us to design an algorithm for ANNOTATED TERRAIN GUARDING. The algorithm is based on the method of divide-and-conquer, although the subproblems we obtain are not associated with subterrains smaller than the original one. Finally, in Subsection 3.4 we deduce that Continuous Terrain Guarding is also solvable in time $n^{O(\sqrt{k})}$.

### 3.1 The Planar Graph $P_S$

In this section we assume that the input instance is a yes-instance. Let $S$ be some hypothetical solution, that is, a subset of $G$ of size at most $k$ that sees $C$. We define three sets of edges:

- The set $E_1$ contains an edge $\{v_i, v_j\}$ between any two vertices $v_i, v_j \in S \cup C$ such that there is no $v_t \in S \cup C$ with $i < t < j$.
- The set $E_2$ contains an edge $\{v_i, v_j\}$ between any two vertices $v_i \in S$ and $v_j \in C \cap \text{vis}(v_i)$ such that $i < j$ and there is no $v_t \in S$ with $t < i$ and $v_j \in \text{vis}(v_t)$.
- The set $E_3$ contains an edge $\{v_i, v_j\}$ between any two vertices $v_i \in S$ and $v_j \in C \cap \text{vis}(v_i)$ such that $j < i$ and there is no $v_t \in S$ with $i < t$ and $v_j \in \text{vis}(v_t)$.

We define $P_S$ as the graph on the vertex set $V(P_S) = S \cup C$ and the edge set $E(P_S) = E_1 \cup E_2 \cup E_3$. Denote the vertices in $V(P_S)$ by $u_1, u_2, \ldots, u_{|V(P_S)|}$ with respect to the order (from left to right) in which they appear on the terrain $T$. An illustrative example is given in Fig. 3. We show that $P_S$ is a planar graph using techniques similar to that in [18]. To show that $P_S$ is a planar graph, we will need the following result, known as the Order Claim, which was proved in [3].

**Lemma 3.2 ([3]).** Let $v_i, v_j, v_k, v_r$ be four vertices in $V$ such that $i < t < j < r$. If $v_i$ sees $v_j$ and $v_t$ sees $v_r$, then $v_i$ sees $v_r$.

Our proof also relies on the following result.

**Lemma 3.3.** There is no pair of edges $\{u_i, u_j\}, \{u_t, u_r\} \in E_2$ such that $i < t < j < r$. Symmetrically, there is no pair of edges $\{u_i, u_j\}, \{u_t, u_r\} \in E_3$ such that $i < t < j < r$.

**Proof.** Since the proofs of the two claims are symmetric, we only prove the first one. Targeting towards a contradiction, suppose that there exist two edges $\{u_i, u_j\}, \{u_t, u_r\} \in E_2$ such that $i < t < j < r$. Since $u_i$ sees $u_j$ and $u_t$ sees $u_r$, by Lemma 3.2, we have that $u_i$ and $u_r$ see each other. By the definition of $E_2$, $u_i, u_r \in S$ and $u_t, u_r \in C$, and it holds that $r$ is the smallest index of a vertex in $S$ that sees $u_r$. However, we deduced that $u_i$ and $u_r$ see each other, which results in a contradiction. □

**Lemma 3.4.** The graph $P_S$ is a planar graph.

**Proof.** For each $i \in [|V(P_S)|]$, place $u_i$ on the point $(i, 0)$ of the two-dimensional Euclidean plane. Draw each edge $\{u_i, u_{i+1}\} \in E_1$ on the line segment between $(i, 0)$ and $(i + 1, 0)$ (see Fig. 4). Since $E(P_S) = E_1 \cup E_2 \cup E_3$, to prove that $P_S$ is a planar graph it is now sufficient to show that all of the edges in $E_2 \setminus E_1$ can be drawn above the line segment $u_i u_{|V(P_S)|}$ without crossing, while all of the
The proof of the following result is given in [2], (see also [9]).

Recall that we use \( tw(H) \) to denote the treewidth and \( \gamma(H) \) the dominating number of a graph \( H \). The proof of the following result is given in [2], (see also [9]).

**Lemma 3.5 ([2]):** There exists a constant \( c \) such that for any planar graph \( H \), \( tw(H) \leq c\sqrt{\gamma(H)} \).
Fig. 4. A precise description of the embedding of the planar graph $P_S$ in Fig. 3 where the edges are colored blue.

From now on, we let $c$ denote the constant mentioned in this lemma. We are now ready to bound the treewidth of $P_S$.

**Lemma 3.6.** $\text{tw}(P_S) \leq c\sqrt{k}$.

**Proof.** We claim that the set $S$ is a dominating set of $P_S$. Indeed, consider some vertex $v_j \in V(P_S) \setminus S$. By the definition of $V(P_S)$, we have that $v_j \in C \setminus S$. The set $S$ sees $C$ and therefore it contains a vertex $v_i$ that sees $v_j$. Without loss of generality, assume that $i < j$, and let $i$ be the smallest index of a vertex in $S$ that sees $v_j$. Observe that $\{v_i, v_j\} \in E_2$, and therefore $v_i$ dominates $v_j$. We have that $\gamma(P_S) \leq |S| \leq k$. By Lemmata 3.4 and 3.5, we conclude that $\text{tw}(P_S) \leq c\sqrt{k}$. $\square$

### 3.2 The Existence of Exploitable Partitions

In this section we continue to assume that the input instance is a yes-instance, and again we let $S$ be some hypothetical solution. Given a subset $U \subseteq G \cup C$ and a mapping $f : \text{cut}(U) \to \{0, 1\}$, denote $\text{cut}(f, 0) = \{T' \in \text{cut}(U) : f(T') = 0\}$ and $\text{cut}(f, 1) = \{T' \in \text{cut}(U) : f(T') = 1\}$. Thus $\text{cut}(f, 0)$ and $\text{cut}(f, 1)$ form a partition of the set of subterrains $\text{cut}(U)$. Moreover, given a subset $X \subseteq V$, denote $X[f, 0] = X[\text{cut}(f, 0)]$ and $X[f, 1] = X[\text{cut}(f, 1)]$. Roughly speaking, we will use such a carefully chosen set $U$ and a function $f$ to achieve the following goal. The set $U$ will partition the terrain $T$ into subterrains, but these subterrains do not necessarily correspond to independent subproblems. Yet, the function $f : \text{cut}(U) \to \{0, 1\}$ will partition $\text{cut}(U)$ into two sets of subterrains.
such that each of them will be independent (in a certain exploitable sense) and relatively small. In our algorithm, the terrain itself will not change—the partition is only meant to control the annotations associated with it.

To make the above mentioned divide-and-conquer approach work, we need the following definition. Each of its properties will be exploited in the following section.

**Definition 3.7.** Let \( U \subseteq G \cup C \), and let \( f \) be a mapping \( f : \text{cut}(U) \to \{0, 1\} \). We say that the pair \((U, f)\) is good if the following conditions are satisfied.

1. \(|U| \leq 2c\sqrt{K}\).
2. \(S \cap U \) sees \( U\).
3. \(|S[f, 0]|, |S[f, 1]| \leq \frac{2}{3}|S|\).
4. \(S \cap (U \cup G[f, 0]) \) sees \(C[f, 0]\); \(S \cap (U \cup G[f, 1]) \) sees \(C[f, 1]\).

Roughly speaking, the motivation behind the introduction of each property is the following. The first property implies that the set \( U \) is small, and therefore it will be possible to “guess” it. The second property implies that by placing guards in the set \( S \cap U \), we can see all of the vertices of \( U \).

The third property implies that the two subproblems are small in a narrow yet exploitable sense: each subproblem will include the entire terrain and therefore its size will be roughly the same as the size of the original problem, yet the number of guards one should place to solve it will be much smaller than the number of guards needed to solve to the original problem. We briefly note that each subproblem will be associated with the entire terrain, including vertices on which we cannot place guards and which are already covered/may not be covered, because such vertices play a role in blocking the lines of sights between other vertices on which we can place guards and vertices that should be covered. The last property implies that the subproblems are independent in the sense that we do not need to cover a vertex of one subproblem using a guard that we place when we solve the other subproblem.

The rest of this section focuses on the proof of the existence of a good pair.

**Lemma 3.8.** There exists a good pair \((U, f)\).

**Proof.** By Lemma 3.6, \( \text{tw}(P_S) \leq c\sqrt{K} \). Therefore, we can let \((D, \beta)\) denote a tree decomposition of \( P_S \) whose width \( w \) is at most \( c\sqrt{K} \). It is known that given a tree decomposition \((D, \beta)\), one can obtain a tree decomposition of the same width where \( D \) is a binary tree [8], and therefore we can assume that \( D \) is a binary tree. By Observation 1 and since \( V(P_S) = G \cup C \), we can let \( d \) be a node in \( V(D) \) such that \(|S|/3 \leq |\bigcup_{d' \in \text{descendants}(d)} \beta(d') \cap S|\) and \(|\bigcup_{d' \in \text{descendants}(d)} \beta(d') \cap S| \leq 2|S|/3\). Denote \( A = (\bigcup_{d' \in \text{descendants}(d)} \beta(d')) \) and \( B = (G \cup C) \setminus (A \cup \beta(d)) \). Then, \(|A \cap S|, |B \cap S| \leq 2|S|/3\). Observe that by the definition of a tree decomposition \( A \cap B = \emptyset \). For each vertex \( v \in \beta(d) \), if \( v \in S \), define \( s(v) = v \), and otherwise let \( s(v) \) denote a vertex in \( S \) that sees \( v \) (if there are several vertices in \( S \) that see \( v \), choose one arbitrarily).

We define \( U = \beta(d) \cup \{s(v) : v \in \beta(d)\} \). At this point, we can already verify that the first two properties in Definition 3.7 are satisfied. First, since \(|\beta(d)| \leq c\sqrt{K} \) and \(|\{s(v) : v \in \beta(d)\}| \leq |\beta(d)| \), we have that \(|U| \leq 2c\sqrt{K}\). Second, since \( \{s(v) : v \in \beta(d)\} \) sees \( \beta(d) \) and \( \{s(v) : v \in \beta(d)\} \subseteq S \), it holds that \( S \cap U \) sees \( U \).

Next, we define the function \( f : \text{cut}(U) \to \{0, 1\} \). To this end, consider some subterrain \( T' \in \text{cut}(U) \). Since \( \beta(d) \subseteq U \), it holds that \( T' \) is a subterrain of some subterrain \( \hat{T} \in \text{cut}(\beta(d)) \). By the definition of a tree decomposition and since \( E_1 \subseteq E(P_S) \), we deduce that either \((G \cup C)[\hat{T}] \subseteq A \) or \((G \cup C)[\hat{T}] \subseteq B \), and therefore either \((S \cup C)[T'] \subseteq A \) or \((S \cup C)[T'] \subseteq B \). This means that \( f \) can classify \( T' \) as follows: \( f(T') = 0 \) if \((S \cup C)[T'] \subseteq A \), and \( f(T') = 1 \) if \((S \cup C)[T'] \subseteq B \). Since
3.3 Divide-and-Conquer

In this section we rely on Lemma 3.8 to design an algorithm, based on the method of divide-and-conquer, that solves Annotated Terrain Guarding in time $n^{O(V)}$.

We start by presenting an algorithmic interpretation of Lemma 3.8. To this end, we need the following definition.

**Definition 3.9.** A tuple $(U, U', f, k_0, k_1)$ is relevant if the following conditions are satisfied.

1. $U \subseteq G \cup C$ satisfies $|U| \leq 2c\sqrt{k}$.
2. $U' \subseteq U \cap G$ sees $U$.
3. $f: \text{cut}(U) \rightarrow \{0, 1\}$.
4. $k_0, k_1 \in \{0\} \cup \lfloor 2k/3 \rfloor; k_0 + k_1 + |U'| = k$.

**Lemma 3.10.** One can compute in time $n^{O(V)}$ a collection $Q$ of relevant tuples whose size is bounded by $n^{O(V)}$ such that if the input instance is a yes-instance, then there exists a solution $S$ of size $k$ and at least one tuple in $Q$ having the following properties.

1. $(U, f)$ is a good pair (with respect to $S$).
2. $U' = U \cap S$.
3. $|S[f, 0]| \leq k_0; |S[f, 1]| \leq k_1$.

**Proof.** Initially, we let $Q = \emptyset$. Then, we execute an exhaustive search which inserts every relevant tuple into $Q$. There are at most $\sum_{i=1}^{2c\sqrt{V}} \left| \frac{|G \cap C|}{i} \right| = 2^{O(V \log n)}$ choices for $U$, where for each set $U$, there are at most $2^{|U|} \leq 2^{c\sqrt{V}}$ choices for $U'$. For each pair $(U, U')$, the domain of $f$, cut$(U)$, is of size $|U| + 1$. Thus, for each pair $(U, U')$, there are at most $2^{|U'|+1} \leq 2^{c\sqrt{V}+1}$ choices for $f$, and for each triple $(U, U', f)$, there are at most $k + 1$ choices for the pair $(k_0, k_1)$. Thus, our procedure runs in time $n^{O(V)}$ and outputs a set $Q$ of size $n^{O(V)}$.

To complete the proof, suppose that the input instance is a yes-instance, and let $S$ be a solution. Let $(U, f)$ be a good pair with respect to $S$ whose existence is guaranteed by Lemma 3.8. Define $U' = U \cap S, k_0 = |S[f, 0]|$ and $k_1 = k - |S[f, 0]| - |U'|$. Then, by Definition 3.7, we deduce that $(U, U', f, k_0, k_1)$ is a relevant tuple and therefore it belongs to $Q$. Clearly, this tuple satisfies the three properties mentioned in the statement of this lemma. □
Let \( Q \) be a collection of tuples given by Lemma 3.10. With each tuple \((U, U', f, k_0, k_1) \in Q\), we associate a pair of instances of Annotated Terrain Guarding, \((I_0(U, U', f, k_0), I_1(U, U', f, k_1))\), as follows: \(I_0(U, U', f, k_0) = (T, k_0, G_0, C_0)\) where \(G_0 = G[f, 0]\) and \(C_0 = C[f, 0] \setminus \text{vis}(U')\); \(I_1(U, U', f, k_1) = (T, k_1, G_1, C_1)\) where \(G_1 = G[f, 1]\) and \(C_1 = C[f, 1] \setminus \text{vis}(U')\). We set \(I(Q) = \{(I_0(U, U', f, k_0), I_1(U, U', f, k_0, k_1)) : (U, U', f, k_0, k_1) \in Q\}\).

**Lemma 3.11.** The input instance is a yes-instance if and only if there exists a pair \((I_0, I_1)\) in \(I(Q)\) such that both \(I_0\) and \(I_1\) are yes-instances.

**Proof.** First, suppose that the input instance is a yes-instance, and let \(S\) be a solution. Then, by Lemma 3.10, there exists \((U, U', f, k_0, k_1) \in Q\) such that \((f, U)\) is a good pair with respect to \(S\), \(U' = U \cap S\), \(|S[f, 0]| \leq k_0\), \(|S[f, 1]| \leq k_1\). Denote \(I_0(U, U', f, k_0) = (T, k_0, G_0, C_0)\) and \(I_1(U, U', f, k_1) = (T, k_1, G_1, C_1)\). By the fourth property in Definition 3.7, \(S \cap (U \cup G[f, 0])\) sees \(G[f, 0]\) and \(S \cap (U \cup G[f, 1])\) sees \(G[f, 1]\), and therefore \(S[f, 0] \subseteq G_0\) and \(S[f, 1] \subseteq G_1\). Clearly, \(S[f, 0] \subseteq G_0\) and \(S[f, 1] \subseteq G_1\), and therefore \(S[f, 0]\) is a solution to \((T, k_0, G_0, C_0)\) and \(S[f, 1]\) is a solution to \((T, k_1, G_1, C_1)\).

For the opposite direction, suppose that there exists a pair \((I_0(U, U', f, k_0), I_1(U, U', f, k_1))\) in \(I(Q)\) such that both \(I_0(U, U', f, k_0)\) and \(I_1(U, U', f, k_1)\) are yes-instances. Denote \(I_0(U, U', f, k_0) = (T, k_0, G_0, C_0)\) and \(I_1(U, U', f, k_1) = (T, k_1, G_1, C_1)\). Let \(S_0\) and \(S_1\) be solutions to \((T, k_0, G_0, C_0)\) and \((T, k_1, G_1, C_1)\), respectively. Define \(S = S_0 \cup S_1 \cup U'\). By the fourth property in Definition 3.9, \(|S| = |S_0| + |S_1| + |U'| \leq k_0 + k_1 + |U'| = k\). By the second property in Definition 3.9 and since \(S_0 \subseteq G_0 \subseteq G\) and \(S_1 \subseteq G_1 \subseteq G\), we have that \(S \subseteq G\). Moreover, by the second property in Definition 3.9, \(U'\) sees \(U \cap C\), and since \(S_0\) sees \(C_0\) and \(S_1\) sees \(C_1\), we have that \(S\) sees \(C\). We conclude that \(S\) is a solution to the input instance. \(\Box\)

We are now ready to prove Theorem 3.1.

**Proof of Theorem 3.1.** We present a recursive algorithm that solves Annotated Terrain Guarding in the desired time. At each stage, if \(k \leq 10c\), it uses brute-force to solve the instance in polynomial time. Otherwise, it computes the set \(I(Q)\) where \(Q\) is given by Lemma 3.10. For each pair \((I_0, I_1)\) in \(I(Q)\), it calls itself recursively twice: once with the input \(I_0\) and once with the input \(I_1\). If the answers to both inputs \(I_0\) and \(I_1\) are positive, it returns a positive answer. At the end, if no pair resulted in two positive answers, it returns a negative answer. By Lemma 3.11, the algorithm returns the correct answer.

By Lemma 3.10, we have that \(|I(Q)| = |Q| = n^{O(\sqrt{n})}\). Consider some pair \((I_0(U, U', f, k_0), I_1(U, U', f, k_1))\) in \(I(Q)\). By the fourth property in Definition 3.9, \(k_0, k_1 \leq 2k/3\). Let \(t(k, n)\) denote the running time of our algorithm. Then, there exists a constant \(d\) such that \(t(k, n) \leq n^{d\sqrt{k}} \cdot t(2k/3)\). Let \(p\) be the largest number smaller than \(10c\) such that there exists \(q\) for which \(\sqrt{(2/3)^{2+p}} = p\). Thus, \(t(n, k) = n^{d\sqrt{k} + \sqrt{(2/3)^{2+p}}} = n^{d\sqrt{k} + \sqrt{(2/3)^{2+p}}} = n^{O(\sqrt{n})}\), since \((1 + \sqrt{(2/3)^{2+p}} + \sqrt{(2/3)^{2+p}} + \cdots + \sqrt{(2/3)^{2+p}} + n^{O(\sqrt{n})})\) is a geometric series that always converges to a constant. \(\Box\)

### 3.4 Continuous Terrain Guarding

Let \((T = (V, E), k)\) be an instance of the Continuous Terrain Guarding problem. Friedrichs et al. [15] proved that one can construct in polynomial time an equivalent instance \((T' = (V', E'), k')\) of Discrete Terrain Guarding such that \(n' = O(n^3)\) and \(k' = k\). By Corollary 2, we have that \((T' = (V', E'), k')\) is solvable in time \(n^{O(\sqrt{n^3})} = n^{O(\sqrt{n})}\). That is, we derive Corollary 3.

We remark that the correctness of this result, and in particular of our conclusion that Continuous Terrain Guarding admits a subexponential-time algorithm, crucially relies on the fact that we solved Discrete Terrain Guarding in time \(n^{O(\sqrt{n^3})}\) rather than \(n^{O(\sqrt{n})}\).
4 PARAMETERIZED ALGORITHM FOR ORTHOGONAL TERRAIN GUARDING

In this section we prove that Discrete Orthogonal Terrain Guarding is FPT:

**Theorem 4.1.** Discrete Orthogonal Terrain Guarding is solvable in time $k^{O(k)} \cdot n^{O(1)}$.

Throughout this section, we let $\mathcal{R}_f$, $\mathcal{R}_r$, $\mathcal{C}_f$ and $\mathcal{C}_r$ denote the sets of left reflex vertices, right reflex vertices, left convex vertices and right convex vertices, respectively. We further let $\mathcal{R} = \mathcal{R}_f \cup \mathcal{R}_r$ and $\mathcal{C} = \mathcal{C}_f \cup \mathcal{C}_r$ denote the sets of reflex vertices and convex vertices, respectively. Katz and Roisman [21] showed that an instance $(T = (V, E), n, k)$ of Discrete Orthogonal Terrain Guarding is a yes-instance if and only if the instance $(T = (V, E), n, k, \mathcal{R}, \mathcal{C})$ of Annotated Orthogonal Terrain Guarding is a yes-instance. In other words, it is sufficient to place guards only on reflex vertices and to guard only convex vertices. Therefore, we say that an instance $(T = (V, E), n, k, G, C)$ of Annotated Orthogonal Terrain Guarding is *relevant* if $\mathcal{R} = G$ and $\mathcal{C} = C$, and in the rest of this section, we focus on the proof of the following result.

**Lemma 4.2.** Relevant instances of Annotated Orthogonal Terrain Guarding are solvable in time $k^{O(k)} \cdot n^{O(1)}$.

First, in Section 4.1, we show that vertices seen by too many vertices of the opposite type can actually be ignored as they will be guarded even if we do not explicitly demand it. In Section 4.2, we describe solutions via clique covers in chordal graphs. This description will allow us to find a set of size at most $k'$, for any $k' \leq k$, that guards a subset of left convex vertices of interest via left reflex vertices, or provide a witness for the non-existence of such a set. Next, in Section 4.3, we examine the Hamming distance between vectors that describe the way in which convex vertices can be guarded, and show that this distance cannot be too large. Finally, in Section 4.4, we integrate the results obtained in the three previous sections into the design of our branching parameterized algorithm for relevant instances of Annotated Orthogonal Terrain Guarding.

4.1 Ignoring Exposed Vertices

In this section, we handle seemingly problematic vertices, which comply with the following definition.

**Definition 4.3.** A vertex $v \in V(T)$ is *exposed* if it is a convex vertex seen by more than $k + 2$ opposite reflex vertices.

We let $E$ denote the set of exposed vertices, $E_f = \mathcal{C}_f \cap E$ and $E_r = \mathcal{C}_r \cap E$. The efficiency of the second phase of our branching procedure, presented in Section 4.4, relies on the assumption that $C$ does not contain exposed vertices. However, $C = \mathcal{C}$, and the set $\mathcal{C}$ may very well contain exposed vertices. We circumvent this difficulty by showing that vertices in $E$ can actually be ignored. To prove this claim, we need the following notation. Given a vertex $v \in E_f$, we let $u^v_1, u^v_2, \ldots, u^v_{k+3}$ denote the $k + 3$ leftmost right reflex vertices that see $v$, sorted from left-to-right by the order in which they lie on $T$ (see Fig. 5(A)). Symmetrically, given a vertex $v \in E_r$, we let $u^v_1, u^v_2, \ldots, u^v_{k+3}$ denote the $k + 3$ rightmost left reflex vertices that see $v$, sorted from right-to-left by the order in which they lie on $T$ (see Fig. 5(B)). By the definition of an orthogonal terrain, we have the following observation.

**Observation 2.** For each vertex $v \in E$, the $x$-coordinate of $u^v_1$ is the same as the one of $v$ and the $y$-coordinate of $u^v_i$ is larger than the one of $v$. For any $2 \leq i \leq k + 3$, the $y$-coordinate of $u^v_i$ is the same as the one of $v$, and if $v \in E_f$ ($v \in E_r$), the $x$-coordinate of $u^v_i$ is larger (resp. smaller) than the one of $v$.

In the two following lemmata, we continue to examine vertices in $E$. 

---

*P. Ashok et. al.*
LEMMA 4.4. For each vertex \( v \in E_f \) and index \( 2 \leq i \leq k + 2 \), there exists a vertex in \( C_f \setminus E_f \) that lies between \( u_i^v \) and \( u_{i+1}^v \). Symmetrically, for each vertex \( v \in E_f \) and index \( 2 \leq i \leq k + 2 \), there exists a vertex in \( C_f \setminus E_f \) that lies between \( u_i^v \) and \( u_{i+1}^v \).

Proof. We only prove the first claim, since the proof of the second claim is symmetric. Consider a vertex \( v \in E_f \) and an index \( 2 \leq i \leq k + 2 \). Since \( u_i^v \) and \( u_{i+1}^v \) are right reflex vertices, there exists at least one left convex vertex that lies between them. Let \( w \) be such a vertex which has the smallest \( y \)-coordinate among all such vertices. To show that \( w \notin E_f \), it is sufficient to show that \( w \) is not seen by more than two right reflex vertices. Suppose, by way of contradiction, that this claim is false. Then, by Observation 2, there exist two right reflex vertices to the right of \( w \) that see \( w \) and whose \( y \)-coordinates are the same as the one of \( w \). In particular, these two right reflex vertices see each other, and they lie between \( u_i^v \) and \( u_{i+1}^v \). However, between two right reflex vertices that see each other and which have the same \( y \)-coordinate, there exists a left convex vertex whose \( y \)-coordinate is smaller than theirs. The existence of such a vertex contradicts the choice of \( w \).

LEMMA 4.5. Let \( S \) be a solution to \((T, n, k, R, C \setminus E)\). For each vertex \( v \in E_f \), there exists an index \( 2 \leq i \leq k + 2 \) and a vertex that lies strictly between \( u_i^v \) and \( u_{i+1}^v \) which is seen by a vertex in \( S \) to the right of \( u_{i+1}^v \). Symmetrically, for each vertex \( v \in E_f \), there exists an index \( 2 \leq i \leq k + 2 \) and a vertex that lies strictly between \( u_i^v \) and \( u_{i+1}^v \) which is seen by a vertex in \( S \) to the left of \( u_{i+1}^v \).

Proof. We only prove the first claim, since the proof of the second claim is symmetric. By Observation 2 and since \( u_{k+3}^v \) sees \( v \), no vertex that lies between \( v \) and \( u_{k+3}^v \) has a \( y \)-coordinate larger than the one of \( v \). In particular, for any pair of distinct indices \( 2 \leq i, j \leq k + 2 \), vertices that lie between \( u_i^v \) and \( u_{i+1}^v \), excluding \( u_{i+1}^v \), cannot see any vertex that lies between \( u_j^v \) and \( u_{j+1}^v \) whose \( y \)-coordinate is smaller than the one of \( v \). Moreover, the \( y \)-coordinate of any convex vertex that lies strictly between \( v \) and \( u_{k+3}^v \) is smaller than the one of \( v \). Therefore, since \(|S| \leq k\), Lemma 4.4 implies that there exists an index \( 2 \leq i \leq k + 2 \) and a vertex that lies strictly between \( u_i^v \) and \( u_{i+1}^v \) and is seen by a vertex \( s \in S \) that does not lie between \( u_i^v \) and \( u_{i+1}^v \). Since no vertex that lies between \( v \) and \( u_{k+3}^v \) has a \( y \)-coordinate larger than the one of \( v \), the vertex \( s \) is a left reflex vertex that lies to the right of \( u_{i+1}^v \).

We are now ready to show that vertices in \( E \) can be ignored.

Fig. 5. Exposed vertices are black, and reflex vertices of the opposite type that see them are white. The parameter \( k \) is 2.
LEMMA 4.6. \((T, n, k, \mathcal{R}, \mathcal{C})\) is a yes-instance if and only if \((T, n, k, \mathcal{R}, \mathcal{C} \setminus E)\) is a yes-instance.

Proof. Since \(\mathcal{C} \setminus E \subseteq \mathcal{C}\), it is clear that the forward direction is true. To prove the reverse direction, suppose that \((T, n, k, \mathcal{R}, \mathcal{C} \setminus E)\) is a yes-instance, and let \(S\) denote a solution. To show that \((T, n, k, \mathcal{R}, \mathcal{C})\) is a yes-instance, it is sufficient to show that \(S\) sees \(E\). We will only show that \(S\) sees \(E_r\), since the proof that \(S\) sees \(E_r\) is symmetric.

Let \(v\) be some left convex vertex in \(E_r\). By Lemma 4.5, there exists an index \(2 \leq i \leq k + 2\) and a vertex that lies strictly between \(w_i^R\) and \(w_{i+1}^R\), which is seen by a vertex \(s \in S\) to the right of \(u_{i+1}^C\). Since \(u_{i+1}^C\) sees \(v\), Lemma 3.2 implies that \(s\) sees \(v\). Since the choice of \(v\) was arbitrary, we conclude that the lemma is correct.

4.2 Describing Solutions via Clique Covers in Chordal Graphs

Katz and Roisman [21] defined two graphs that aim to capture relations between convex vertices. The first graph, \(G_L\), is defined as follows: \(V(G_L) = \mathcal{C}_L\) and \(E(G_L) = \{(u, v) : \text{there exists a vertex in} \ \mathcal{R}_L \text{that sees both} \ u \text{and} \ v\}\). The second one, \(G_R\), is defined symmetrically: \(V(G_R) = \mathcal{C}_R\) and \(E(G_R) = \{(u, v) : \text{there exists a vertex in} \ \mathcal{R}_R \text{that sees both} \ u \text{and} \ v\}\). For these graphs, Katz and Roisman [21] proved the following useful result.

LEMMA 4.7 ([21]). The graph \(G_L\) satisfies the following properties.
- The graph \(G_L\) is a chordal graph.
- For any subset \(U \subseteq V(G_L)\), \(G_L[U]\) is a clique if and only if there exists a left reflex vertex that sees all of the vertices in \(U\).

The symmetric claim holds for the graph \(G_R\).

By relying on Lemma 4.7, Katz and Roisman [21] showed that one can decide in polynomial time whether there exists a subset \(S \subseteq \mathcal{R}_L\) of size \(k'\) that sees \(\mathcal{C}_L\). To design our branching procedure (in Section 4.4), we will need the following stronger claim.

LEMMA 4.8. Let \(U \subseteq \mathcal{C}_L\) and \(k' \in \mathbb{N}\). Then, one can decide in polynomial time whether there exists a subset \(S \subseteq \mathcal{R}_L\) of size \(k'\) that sees \(U\). If such a subset does not exist, one can find in polynomial time a subset \(U' \subseteq U\) of size \(k' + 1\) such that there does not exist a subset \(S \subseteq \mathcal{R}_L\) of size \(k'\) that sees \(U'\).

Proof. Since the family of chordal graphs is hereditary, Property 1 of Lemma 4.7 implies that \(G_L[U]\) is a chordal graph. Thus, by Theorem 2.1, \(\kappa(G_L[U])\) can be computed in polynomial time. Moreover, Property 2 of Lemma 4.7 implies that there exists a subset \(S \subseteq \mathcal{R}_L\) of size \(k'\) that sees \(U\) if and only if \(\kappa(G_L[U]) \leq k'\). We deduce that one can decide in polynomial time whether there exists a subset \(S \subseteq \mathcal{R}_L\) of size \(k'\) that sees \(U\).

Now, suppose that there does not exist a subset \(S \subseteq \mathcal{R}_L\) of size \(k'\) that sees \(U\). Then, \(\kappa(G_L[U]) > k'\). By Theorem 2.1, it also holds that \(\alpha(G_L[U]) > k'\), and we can thus obtain (in polynomial time) an independent set \(U'\) of \(G_L[U]\) of size \(k' + 1\). If \(U'\) could have been seen by at most \(k'\) vertices in \(\mathcal{R}_L\), by the pigeonhole principle, there should have been a vertex in \(\mathcal{R}_L\) that sees at least two vertices in \(U'\). This would have implied that there exists an edge in \(G_L[U]\) between two vertices in \(U'\), contradicting the fact that \(U'\) is an independent set of \(G_L[U]\).

Symmetrically, we obtain the following claim.

LEMMA 4.9. Let \(U \subseteq \mathcal{C}_R\) and \(k' \in \mathbb{N}\). Then, one can decide in polynomial time whether there exists a subset \(S \subseteq \mathcal{R}_R\) of size \(k'\) that sees \(U\). If such a subset does not exist, one can find in polynomial time a subset \(U' \subseteq U\) of size \(k' + 1\) such that there does not exist a subset \(S \subseteq \mathcal{R}_R\) of size \(k'\) that sees \(U'\).
4.3 Hamming Distance

In this section, we associate vectors with subsets of $R$, and then examine the Hamming distance between these vectors and a special vector. We start with the definition of the association. Here, we set $m = |C \setminus E|$, and let $u_1, u_2, \ldots, u_m$ denote the vertices in $C \setminus E$ sorted from left-to-right by the order in which they lie on $T$.

**Definition 4.10.** Let $S \subseteq \mathcal{R}$ be a set that sees $C$. Then, the vector associated with $S$ is the $m$-length bit vector $(b_1, b_2, \ldots, b_m)$ such that $b_i = 0$ if and only if $u_i$ is seen by a vertex in $S$ that is a left reflex vertex.

In other words, the vector associated with a subset $S \subseteq \mathcal{R}$ that sees $C$ indicates, for each vertex that we would like to guard, whether it is guarded by at least one left reflex vertex or only by right reflex vertices. Observe that by the definition of an orthogonal terrain, a reflex vertex can see at most two vertices of the opposite type:

**Observation 3.** Any reflex vertex $v$ sees at most two convex vertices of the opposite type: one has the same $x$-coordinate as $v$ and the other has the same $y$-coordinate as $v$.

Next, we examine the Hamming distance between a vector associated with a solution and a special vector.

**Lemma 4.11.** Let $S^*$ be a solution to $(T, n, k, \mathcal{R}, C \setminus E)$, and let $\vec{b}^*$ be the $m$-length bit vector associated with $S^*$. Let $\vec{b}$ be the $m$-length bit vector $(b_1, b_2, \ldots, b_m)$ such that $b_i = 0$ if and only if $u_i$ is a left convex vertex. Then, $H(\vec{b}^*, \vec{b}) \leq 2k$.

**Proof.** Since $|S^*| \leq k$, Observation 3 implies that at most $2k$ vertices in $C \setminus E$ are seen by vertices in $S^*$ of the opposite type. Therefore, apart from at most $2k$ vertices, each vertex in $C \setminus E$ is seen only by vertices in $S^*$ of the same type. This observation implies that $H(\vec{b}^*, \vec{b}) \leq 2k$. \hfill \Box

The study of vectors associated with subsets of $\mathcal{R}$, and the Hamming distance between these vectors and the vector $\vec{b}$ helps in the analysis of the branching algorithm that will be described in the following Section. The Hamming distance serves as a measure for the branching algorithm and this is how Lemma 4.11 will be useful to us to bound the running time of the branching algorithm.

4.4 Branching

We are now ready to present ALG$(T = (V, E), n, \mathcal{R}, C, \delta, k_F, k_r)$, our algorithm for relevant instances of** Annotated Orthogonal Terrain Guarding.** Initially, it is called with the arguments $C = C \setminus E$, $\delta = 2k$, and every choice of $k_F, k_r \in [k]$ such that $k_F + k_r = k$. As the execution of the algorithm progresses, vertices are removed from $C$, and the values of $k_F, k_r$, and $\delta$ decrease. Note that there are only $k$ choices of $k_F$ and $k_r$, and there exists a choice of $k_F$ and $k_r$ such that if there exists a solution $S$, it holds that $|S \cap \mathcal{R}_f| = k_F$ and $|S \cap \mathcal{R}_r| = k_r$. Accordingly, and in light of Lemma 4.11, we say that the input instance (in the context of a pair $(k_F, k_r)$) is identifiable if there exists a solution $S$ such that $|S \cap \mathcal{R}_f| = k_F$, $|S \cap \mathcal{R}_r| = k_r$ and the Hamming distance between $\vec{b}$ and the vector associated with $S$ is at most $\delta$. Thus, to prove Lemma 4.2, it is sufficient to prove the following result.

**Lemma 4.12.** ALG$(T = (V, E), n, \mathcal{R}, C, \delta, k_F, k_r)$ runs in time $k^{O(k)} \cdot n^{O(1)}$, and returns YES if and only if the input instance is identifiable.

**Proof.** The pseudocode of our algorithm is given in Algorithm 1. First, we argue that if the input instance is identifiable, then the algorithm returns YES. In this argument, we follow the pseudocode
we also decrement

We decrement

YES only if

any set of at most

lossofgenerality,

, Vol. 1, No. 1, Article . Publication date: November 2017.

Algorithm 1: ALG\((T = (V, E), n, R, C, \delta, k_\ell, k_r)\)

1: if \(\delta < 0\) then
2: Return NO.
3: else if \(C \cap C_L\) cannot be seen by any set of at most \(k_\ell\) vertices from \(R_L\) then
4: Compute a set \(U \subseteq C \cap C_L\) of size \(k_\ell + 1\) that cannot be seen by any set of at most \(k_\ell\) vertices from \(R_L\).
5: for all \(v \in U\) do
6: for all \(u \in R_R \cap \text{vis}(v)\) do Return ALG\((T = (V, E), n, R, C \setminus \text{vis}(u), \delta - 1, k_\ell, k_r - 1)\).
7: end for
8: end if
9: Compute a set \(U \subseteq C \cap C_R\) of size \(k_r + 1\) that cannot be seen by any set of at most \(k_r\) vertices from \(R_R\).
10: for all \(v \in U\) do
11: for all \(u \in R_L \cap \text{vis}(v)\) do Return ALG\((T = (V, E), n, R, C \setminus \text{vis}(u), \delta - 1, k_\ell - 1, k_r)\).
13: end for
14: else
15: Return YES.
end if

line-by-line, and also highlight the phases of our branching. In case \(\delta < 0\), we return NO, since the Hamming distance between any two vectors is nonnegative. Next, suppose that \(\delta \geq 0\). By Lemma 4.8, we may proceed by deciding in polynomial time whether \(C \cap C_L\) cannot be seen by any set of at most \(k_\ell\) vertices from \(R_L\). If this is the case, then by Lemma 4.8 we can compute in polynomial time a set \(U \subseteq C \cap C_L\) of size \(k_\ell + 1\) that cannot be seen by any set of at most \(k_\ell\) vertices from \(R_L\). In case the input instance is identifiable, there exists a vertex \(v \in U\) that should be seen by a right reflex vertex. We try every option to identify the vertex \(v\); this is the first phase of our branching. Then, we try every option to identify a vertex \(u \in R_R \cap \text{vis}(v)\) that should both see \(v\) and belong to a solution; this is the second phase of our branching. Since \(v\) is not exposed, there are at most \(k + 2\) such options to consider. For each such option, we place a guard on \(u\). Therefore, we decrement \(k_r\) by 1, remove the vertices in \(\text{vis}(u)\) from \(C\), and since at least one bit is flipped in \(\overline{T}\), we also decrement \(\delta\) by 1. For an identifiable input instance, we will have made correct choices in at least one of the paths in the branch-tree. Now, if the answer is negative, by performing the symmetric test with respect to the set \(C \cap C_R\), we can safely conclude that an identifiable instance is detected correctly.

Suppose the algorithm returns YES for \((T = (V, E), n, R, C, \delta, k_\ell, k_r)\). In this case we prove, by induction on \(\delta\), that \(C\) can be guarded by \(k_\ell\) vertices of \(R_L\) and \(k_r\) vertices of \(R_R\). If \(\delta < 0\) then the algorithm returns NO. Therefore, the base case is when \(\delta = 0\). In this case, the algorithm returns YES only if \(C \cap C_L\) can be seen by any set of at most \(k_\ell\) vertices from \(R_L\) and \(C \cap C_R\) can be seen by any set of at most \(k_r\) vertices from \(R_R\). This also means that the relevant instance is a yes-instance, and therefore the algorithm returns the correct answer. Let us assume the induction hypothesis that the algorithm returns the correct answer for all recursive calls with \(\delta < d\). Suppose the input instance of the algorithm has \(\delta = d\). It is possible that the algorithm returns YES because \(C \cap C_L\) can be seen by any set of at most \(k_\ell\) vertices from \(R_L\) and \(C \cap C_R\) can be seen by any set of at most \(k_r\) vertices from \(R_R\). In this case the algorithm correctly detects a yes-instance. Otherwise, without loss of generality, \(C \cap C_L\) cannot be seen by any set of at most \(k_\ell\) vertices from \(R_L\). This means that the algorithm computes a set \(U \subseteq C \cap C_L\) of size \(k_\ell + 1\) that cannot be seen by any set of at most \(k_\ell\) vertices from \(R_L\).
We studied the well-known Terrain Guarding problem, addressing two fundamental questions:

1. Does Terrain Guarding admit a polynomial kernel?
2. Is Terrain Guarding FPT with respect to $k$?

We have resolved the first question: both Discrete Terrain Guarding and Continuous Terrain Guarding admit subexponential-time algorithms. For discrete orthogonal domains we have also resolved the second question: Discrete Orthogonal Terrain Guarding is FPT.

We would like to conclude our paper by suggesting several directions for further research. First and foremost, it remains to establish the fixed-parameter (in)tractability of Terrain Guarding in general (discrete and continuous) domains, as well as to determine whether Discrete Orthogonal Terrain Guarding is NP-hard or not. In case Terrain Guarding is FPT, one can further ask whether it admits a polynomial kernel. An affirmative answer to this question, combined with our subexponential-time algorithm, would imply that Terrain Guarding admits a subexponential-time parameterized algorithm. Finally, it would also be interesting to investigate whether the running time of our subexponential-time algorithm can be substantially improved, or whether it is essentially tight under reasonable complexity assumptions. We remark that the proof given by King and Krohn [24] to show that Terrain Guarding is NP-hard only implies that unless ETH fails, Terrain Guarding cannot be solved in time $2^{o(n^{1/2})}$.

In a recent manuscript, Bonnet and Giannopoulos [4] showed that Discrete Orthogonal Terrain Guarding is NP-complete. They also showed that the reduction of King and Krohn [24], as well as their reduction for Discrete Orthogonal Terrain Guarding, can actually be made linear (instead of quadratic). Using these new reductions, it is proved that unless ETH fails, Terrain Guarding and Discrete Orthogonal Terrain Guarding cannot be solved in time $2^{o(\sqrt{n})}$. Thus, the results of Bonnet and Giannopoulos [4] resolve two of our open problems.

5 CONCLUSION AND SUBSEQUENT WORK

We studied the well-known Terrain Guarding problem, addressing two fundamental questions relating to its complexity:

- Does Terrain Guarding admit a subexponential-time algorithm?
- Is Terrain Guarding FPT with respect to $k$?

REFERENCES
