Orthogonal Terrain Guarding is NP-complete

Édouard Bonnet
ENS Lyon, LIP
Lyon, France
edouard.bonnet@dauphine.fr

Panos Giannopoulos
Department of Computer Science, Middlesex University
London, UK
p.giannopoulos@mdx.ac.uk

Abstract

A terrain is an x-monotone polygonal curve, i.e., successive vertices have increasing x-coordinates. Terrain Guarding can be seen as a special case of the famous art gallery problem where one has to place at most k guards on a terrain made of n vertices in order to fully see it. In 2010, King and Krohn showed that Terrain Guarding is NP-complete [SODA '10, SIAM J. Comput. '11] thereby solving a long-standing open question. They observe that their proof does not settle the complexity of Orthogonal Terrain Guarding where the terrain only consists of horizontal or vertical segments; those terrains are called rectilinear or orthogonal. Recently, Ashok et al. [SoCG'17] presented an FPT algorithm running in time $k^{O(k)}n^{O(1)}$ for Dominating Set in the visibility graphs of rectilinear terrains without 180-degree vertices. They ask if Orthogonal Terrain Guarding is in P or NP-hard. In the same paper, they give a subexponential-time algorithm running in $n^{O(\sqrt{n})}$ (actually even $n^{O(\sqrt{k})}$) for the general Terrain Guarding and notice that the hardness proof of King and Krohn only disproves a running time $2^{o(n^{1/4})}$ under the ETH. Hence, there is a significant gap between their $2^{O(n^{1/2} \log n)}$-algorithm and the no $2^{o(n^{1/4})}$ ETH-hardness implied by King and Krohn's result.

In this paper, we answer those two remaining questions. We adapt the gadgets of King and Krohn to rectilinear terrains in order to prove that even Orthogonal Terrain Guarding is NP-complete. Then, we show how their reduction from Planar 3-SAT (as well as our adaptation for rectilinear terrains) can actually be made linear (instead of quadratic).

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases terrain guarding, rectilinear terrain, computational complexity

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.11

Related Version A full version of this paper is available at `https://arxiv.org/abs/1710.00386`

Funding Supported by EPSRC grant FptGeom (EP/N029143/1)

1 Introduction

Terrain Guarding is a natural restriction of the well-known art gallery problem. It asks, given an integer k, and an *x-monotone polygonal chain* or *terrain*, to guard it by placing at most k guards at vertices of the terrain. An *x-monotone polygonal chain* is defined from a sequence of n points of the real plane $\mathbb{R}^2$ $p_1 = (x_1, y_1), p_2 = (x_2, y_2), \ldots, p_n = (x_n, y_n)$ such that $x_1 \leqslant x_2 \leqslant \ldots \leqslant x_n$ as the succession of straight-line edges $p_1p_2, p_2p_3, \ldots, p_{n-1}p_n$. Each point p_i is called a *vertex* of the terrain. We can make each coordinate of the vertices rational without changing the (non-)existence of a solution. We will therefore assume that the input is given as a list of n pairs of rational numbers, together with the integer k. A point p lying on the terrain is *guarded* (or seen) by a subset S of guards if there is at least one guard $g \in S$ such that the straight-line segment pg is entirely above the polygonal chain. The terrain is said *guarded* if every point lying on the terrain is guarded. The visibility graph of a terrain has as vertices the geometric vertices of the polygonal chain and as edges every pair which sees each other. Again two vertices (or points) see each other if the straight-line segment that they define is above the terrain.

The Orthogonal Terrain Guarding is the same problem restricted to *rectilinear* (also called *orthogonal*) terrains, that is every edge of the terrain is either horizontal or vertical. In other words, p_i and p_{i+1} share the same x-coordinate or the same y-coordinate. We say that a rectilinear terrain is *strictly rectilinear* (or *strictly orthogonal*) if the horizontal and vertical edges alternate, that is, there are no two consecutive horizontal (resp. vertical) edges. Both problems come with two other variants: the *continuous variant*, where the guards can be placed anywhere on the edges of the terrain (and not only at the vertices), and the *graphic variant*, which consists of Dominating Set in the visibility graphs of (strictly rectilinear) terrains. The original problem is sometimes called the *discrete variant.*

It is a folklore observation that for rectilinear terrains, the discrete and continuous variants coincide. Indeed, it is an easy exercise to show that from any feasible solution using guards in the interior of edges, one can move those guards to vertices and obtain a feasible solution of equal cardinality. The only rule to respect is that if an edge, whose interior contained a guard, links a reflex and a convex vertex, then the guard should be moved to the reflex vertex. We will therefore only consider Orthogonal Terrain Guarding and Dominating Set in the visibility graphs of strictly rectilinear terrains. By subdividing the edges of a strictly rectilinear terrain with an at most quadratic number of 180-degree vertices (i.e., vertices incident to two horizontal edges or to two vertical edges), one can make *guarding all the vertices* equivalent to *guarding the whole terrain.* Therefore, Orthogonal Terrain Guarding is not very different from Dominating Set in the visibility graph of (non necessarily strictly) rectilinear terrains (and Terrain Guarding is not very different from Dominating Set in the visibility graph of terrains).

Exponential Time Hypothesis. The *Exponential Time Hypothesis* (usually referred to as the ETH) is a stronger assumption than P≠NP formulated by Impagliazzo and Paturi [14]. A practical (and slightly weaker) statement of ETH is that 3-SAT with n variables cannot be solved in subexponential-time $2^{o(n)}$. Although this is not the original statement of the hypothesis, this version is most commonly used; see also Impagliazzo et al. [15]. The so-called *sparsification lemma* even brings the number of clauses in the exponent.

▶ **Theorem 1** (Impagliazzo and Paturi [14]). *Under the ETH, there is no algorithm solving every instance of* 3-SAT *with n variables and m clauses in time* $2^{o(n+m)}$.

As a direct consequence, unless the ETH fails, even instances with a linear number of clauses $m = \Theta(n)$ cannot be solved in $2^{o(n)}$. Unlike P≠NP, the ETH allows to rule out specific running times. We refer the reader to the survey by Lokshtanov et al. for more information about ETH and conditional lower bounds [23].

Planar satisfiability. Planar 3-SAT was introduced by Lichtenstein [22] who showed its NP-hardness. It is a special case of 3-SAT where the variable/clause incidence graph is planar even if one adds edges between two consecutive variables for a specified ordering of the variables: $x_1, x_2, \dots, x_n$; i.e., $x_i x_{i+1}$ is an edge (with index $i+1$ taken modulo n). This extra

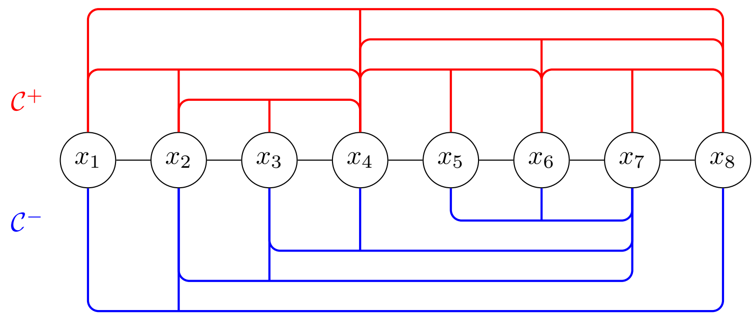

Figure 1 The bipartition $(\mathcal{C}^+,\mathcal{C}^-)$ of a PLANAR 3-SAT-instance. The three-legged arches represent the clauses. Here is a removal ordering for $\mathcal{C}^-$: remove the clause on x_5, x_6, x_7 and its middle variable x_6, remove the variable x_5, remove the clause on x_3, x_4, x_7 and its middle variable x_4, remove the clause on x_2, x_3, x_7 and its middle variable x_3, remove the variable x_7, remove the clause x_1, x_2, x_8 and its middle variable x_2.

structure makes this problem particularly suitable to reduce to planar or geometric problems. As a counterpart, the ETH lower bound that one gets from a linear reduction from PLANAR 3-SAT is worse than with a linear reduction from 3-SAT; it only rules out a running time $2^{o(\sqrt{n})}$. Indeed, PLANAR 3-SAT can be solved in time $2^{O(\sqrt{n})}$ and, unless the ETH fails, cannot be solved in time $2^{o(\sqrt{n})}$. A useful property of any PLANAR 3-SAT-instance is that its set of clauses $\mathcal{C}$ can be partitioned into $\mathcal{C}^+$ and $\mathcal{C}^-$ such that both $\mathcal{C}^+$ and $\mathcal{C}^-$ admit a *removal ordering*. A *removal ordering* is a sequence of the two following deletions:

- (a) removing a variable which is not present in any remaining clause
- (b) removing a clause on *three consecutive remaining variables* together with the *middle variable*

which ends up with an empty set of clauses. By *three consecutive remaining variables*, we mean three variables x_i, x_j, x_k, with $i < j < k$ such that $x_{i+1}, x_{i+2}, \ldots, x_{j-1}$ and $x_{j+1}, x_{j+2}, \ldots, x_{k-1}$ have all been removed already. The middle variable of the clause is x_j. For an example, see Figure 1.

Order claim. The following visibility property in a terrain made King and Krohn realize that they will crucially need the extra structure given by the planarity of 3-SAT-instances.

▶ **Lemma 2** (Order Claim, see Figure 2). *If a, b, c, d happen in this order from the left endpoint of the terrain to its right endpoint, a and c see each other, and b and d see each other, then a and d also see each other.*

In particular, this suggests that checking in the terrain *if a clause is satisfied* can only work if the encodings of the three variables contained in the clause are *consecutive*.

Related work and remaining open questions for terrain guarding. TERRAIN GUARDING was shown NP-hard [18] and can be solved in time $n^{O(\sqrt{k})}$ [1]. This contrasts with the

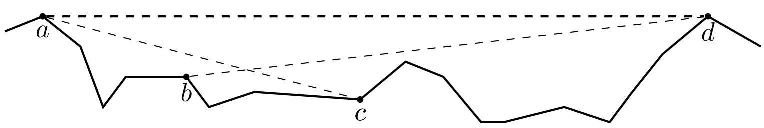

Figure 2 The order claim.

parameterized complexity of the more general art gallery problem where an algorithm running in time $f(k)n^{o(k/\log k)}$ for any computable function f would disprove the ETH, both for the variant Point Guard Art Gallery where the k guards can be placed anywhere inside the gallery (polygon with n vertices) and for the variant Vertex Guard Art Gallery where the k guards can only be placed at the vertices of the polygon [4], even when the gallery is a simple polygon (i.e., does not have holes). Dominating Set on the visibility graph of strictly rectilinear terrains can be solved in time $k^{O(k)}n^{O(1)}$ [1], while it is still not known if (Orthogonal) Terrain Guarding can be solved in FPT time $f(k)n^{O(1)}$ with respect to the number of guards.

There has been a succession of approximation algorithms with better and better constant ratios [16, 7, 2, 13]. Eventually, a PTAS was found for Terrain Guarding (hence for Orthogonal Terrain Guarding) [20] using local search and an idea developed by Chan and Har-Peled [6] and Mustafa and Ray [24] which consists of applying the planar separator theorem to a (planar) graph relating local and global optima. Interestingly, this planar graph is the starting point of the subexponential algorithm of Ashok et al. [1].

Again the situation is not nearly as good for the art gallery problem. If holes are allowed in the polygon, the main variants of the art gallery problem are as hard as the Set Cover problem; hence a $o(\log n)$-approximation cannot exist unless P=NP [11]. Eidenbenz also showed that a PTAS is unlikely in simple polygons [10]. For simple polygons, there is a $O(\log \log OPT)$-approximation [17, 19] for Vertex Guard Art Gallery, using the framework of Brönnimann and Goodrich to transform an ε-net finder into an approximation algorithm, and for Point Guard Art Gallery there is a randomized $O(\log OPT)$-approximation under some mild assumptions [5], building up on [9, 8]. If a small fraction of the polygon can be left unguarded there is again a $O(\log OPT)$-approximation [12]. A constant approximation is known for *monotone polygons* [21], where a monotone polygon is made of two terrains sharing the same left and right endpoints and except those two points the two terrains are never touching nor crossing.

The classical complexity of Orthogonal Terrain Guarding remains the most pressing open question [1].

▶ **Open question 1.** *Is* Orthogonal Terrain Guarding *in P or NP-hard?*

In the conclusion of the paper by Ashok et al. [1], the authors observe that the construction of King and Krohn [18] rules out for Terrain Guarding a running time of $2^{o(n^{1/4})}$, under the ETH. Indeed the reduction from Planar 3-SAT (which is not solvable in time $2^{o(\sqrt{n})}$ unless the ETH fails) and its adaptation for Orthogonal Terrain Guarding in the current paper have a quadratic blow-up: the terrain is made of $\Theta(m) = \Theta(n)$ chunks containing each $O(n)$ vertices. On the positive side, the subexponential algorithm of Ashok et al. runs in time $2^{O(\sqrt{n} \log n)}$ [1]. Therefore, there was still a significant gap between the algorithmic upper and lower bounds.

▶ **Open question 2.** *Assuming the ETH, what is the provably best asymptotic running time for* Terrain Guarding *and* Orthogonal Terrain Guarding*?*

We resolve both open questions. It is remarkable that within the last ten years, knowledge on the computational complexity of (Orthogonal) Terrain Guarding has gone from a handful of constant-approximation algorithms and no lower bound at all to tight approximability under P≠NP and almost tight ETH-hardness. Our paper provides the two missing pieces: the NP-hardness of Orthogonal Terrain Guarding and a versatile refinement of a quadratic reduction from Planar 3-SAT to a linear reduction.

Organization. In Section 2, we address Open question 1 by showing that Orthogonal Terrain Guarding is also NP-hard. We design a rectilinear subterrain with a constant number of vertices which simulates a triangular pocket surrounded by two horizontal segments. We can then adapt the reduction of King and Krohn [18] to rectilinear terrains. Our orthogonal gadgets make an extensive use of the triangular pockets.

In Section 3, we show how to make both reductions from Planar 3-SAT linear instead of quadratic. This shows that, under the ETH, Terrain Guarding and Orthogonal Terrain Guarding cannot be solved in $2^{o(\sqrt{n})}$, and thereby resolve Open question 2. Unless the ETH fails, the $2^{O(\sqrt{n}\log n)}$-time algorithm of Ashok et al. is optimal up to logarithmic factors in the exponent.

2 Orthogonal Terrain Guarding is NP-complete

King and Krohn give a reduction with a quadratic blow-up from Planar 3-SAT to Terrain Guarding [18]. They argue that the order claim entails some critical obstacle against straightforward hardness attempts. In some sense, the subexponential algorithm running in time $n^{O(\sqrt{n})}$ of Ashok et al. [1] proves them right: unless the ETH fails, there cannot be a linear reduction from 3-SAT to Terrain Guarding. It also justifies their idea of starting from the planar variant of 3-SAT. Indeed, this problem can be solved in time $2^{O(\sqrt{n})}$. However, we will see that the quadratic blow-up of their construction is avoidable. In the next section, we show how to make their reduction (and ours for the orthogonal case) linear. In this section, we focus on our main result: the NP-hardness of Orthogonal Terrain Guarding.

From far, King and Krohn's construction looks like a V-shaped terrain. If one zooms in, one perceives that the V is made of $\Theta(n)$ connected subterrains called *chunks*. If one zooms a bit more, one sees that the chunks are made of up to n variable encodings each. Let us order the chunks from bottom to top; in this order, the chunks alternate between the right and the left of the V (see Figure 3).

The construction is such that only two consecutive chunks interact. More precisely, a vertex of a given chunk T_i only sees bits of the terrain contained in T_{i-1}, T_i, and T_{i+1}. Half-way to the top is the chunk T_0 that can be seen as the *initial* one. It contains the

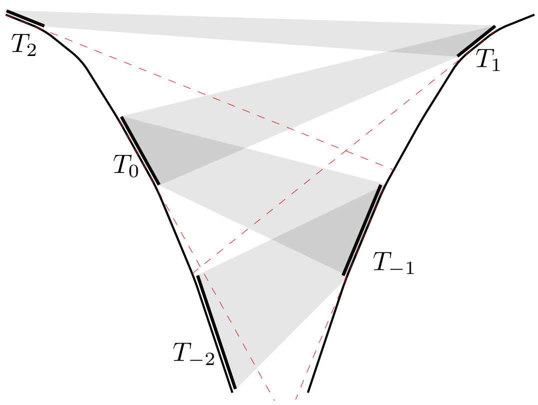

Figure 3 The V-shaped terrain and its ordered chunks. The chunk T_i only sees parts of chunks T_{i-1} and T_{i+1}. The *initial* chunk T_0 contains an encoding of each variable. Below this level (chunks with a negative index), we will check the clauses of $\mathcal{C}^-$. Above this level (chunks with a positive index), we will deal with the clauses of $\mathcal{C}^+$.

encoding of *all* the variables of the Planar 3-SAT-instance. Concretely, the reasonable choices to place guards on the chunk T_0 are interpreted as setting each variable to either *true* or *false*. Let $(\mathcal{C}^+, \mathcal{C}^-)$ be the bipartition of the clauses into two sets with a removal ordering for the variables ordered as $x_1, x_2, \dots, x_n$. Let $C_1^+, C_2^+, \dots, C_s^+$ (resp. $C_1^-, C_2^-, \dots, C_{m-s}^-$) be the order in which the clauses of $\mathcal{C}^+$ (resp. $\mathcal{C}^-$) disappear in this removal ordering. Every chunk below T_0, i.e., with a negative index, are dedicated to checking the clauses of $\mathcal{C}^-$ in the order $C_1^-, C_2^-, \dots, C_{m-s}^-$, while every chunk above T_0, i.e., with a positive index, will check *if the clauses of* C^+ *are satisfied* in the order $C_1^+, C_2^+, \dots, C_s^+$. The placement of the chunks will *propagate downward and upward* the truth assignment of T_0, and simulate the operations of a removal ordering: checking/removing a clause and its middle variable, removing a useless variable. Note that for those gadgets, we will have to distinguish if we are *going up* ($\mathcal{C}^+$) or *going down* ($\mathcal{C}^-$). In addition, the respective position of the positive and negative literals of a variable appearing in the next clause to check will matter. So, we will require a gadget to invert those two literals if needed.

To sum up, the reduction comprises the following gadgets: encoding a variable (variable gadget), propagation of its assignment from one chunk to a consecutive chunk (interaction of two variable gadgets), inverting its two literals (inverter), checking a clause *upward* and removing the henceforth useless middle variable (upward clause gadget), checking a clause *downward* and removing the henceforth useless middle variable (downward clause gadget), removing a variable (upward/downward deletion gadget). Although King and Krohn rather crucially rely on having different slopes in the terrain, we will mimic their construction gadget by gadget with an orthogonal terrain. We start by showing how to simulate a restricted form of a triangular pocket. This will prove to be a useful building block.

The simulation of a *right trapezoid pocket* giving rise to the *right triangular pocket* is depicted on Figure 4. The idea is that the vertex p at the bottom of the pit is only seen by four vertices (no vertex outside this gadget will be able to see p). Among those four vertices, u sees a strict superset of what the others see. Hence, we can assume with no loss of generality that a guard is placed on u. Now, u sees the part of the terrain represented in bold. Even if vertex u sees a part of the vertical edge incident to a (actually the construction could avoid it), this information can be discarded since the guard responsible for seeing a will see this edge entirely. Everything is therefore equivalent to guarding the terrain with the right trapezoid pocket drawn in the middle of Figure 4 with a budget of guards decreased

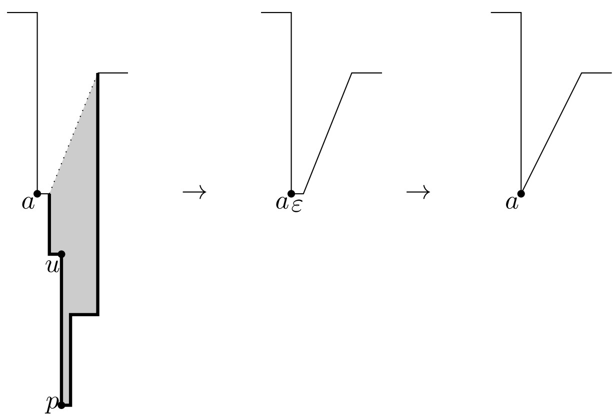

Figure 4 Simulation of a right trapezoid pocket and a right triangular pocket. The right triangular pocket is obtained from the right trapezoid by making the distance ε sufficiently small.

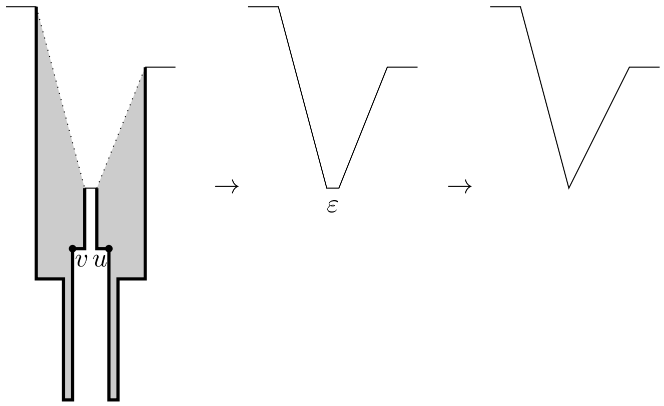

Figure 5 Simulation of a trapezoid pocket and a triangular pocket. The triangular pocket is obtained from the trapezoid by making the distance ε arbitrary small.

by one. If the length of the horizontal edge incident to a is made small enough, then every guard seeing a will see the whole edge, thereby simulating the right triangular pocket to the right of the figure.

The acute angle made by the right triangular pocket and the altitude of the leftmost and rightmost horizontal edge in this gadget can be set at our convenience. We will represent triangular pockets in the upcoming gadgets. The reader should keep in mind that they are actually a shorthand for a rectilinear subterrain.

With the same idea, one can simulate a general triangular pocket as depicted on Figure 5, with the budget decreased by two guards. Again, the non-reflex angle made by the triangular pocket and the altitude of the leftmost and rightmost horizontal edges can be freely chosen. The reason why those triangular pockets do not provide a straightforward reduction from the general Terrain Guarding problem is that the pocket has to be preceded and succeeded by horizontal edges.

The variable gadget is depicted on Figure 6. It is made of three right triangular pockets. Placing a guard on v_x (resp. $v_{\overline{x}}$) is interpreted as setting the variable x to *true* (resp. *false*).

The propagation of a variable assignment from one chunk to the next chunk is represented on Figure 7. On all the upcoming figures, we adopt the convention that red dashed lines materialize a blocked visibility (the vertex cannot see anything *below this line*) and black dashed lines highlight important visibility which sets apart the vertex from other vertices.

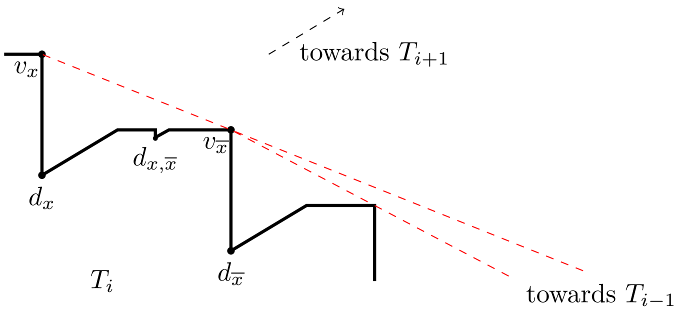

Figure 6 A variable gadget. We omit superscript i on all the labels. Placing a guard at vertex v_x to see d_x corresponds to setting variable x to true, while placing it at vertex $v_{\overline{x}}$ to see $d_{\overline{x}}$ corresponds to setting x to false. Both v_x^{i+1} and $v_{\overline{x}}^{i+1}$ of T_{i+1} (not represented on this picture) see $d_{x,\overline{x}}$ of T_i.

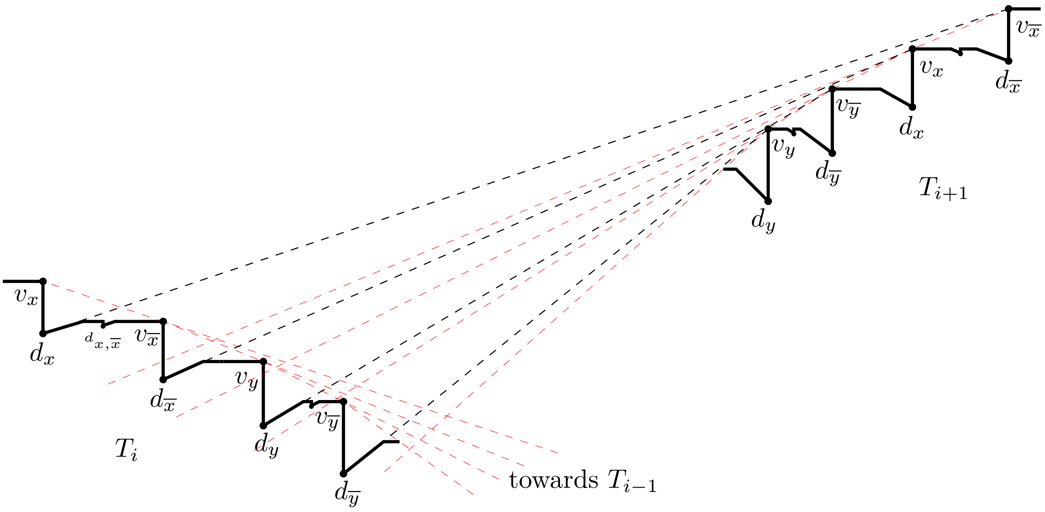

Figure 7 Propagating variable assignments upward and downward. Note that the positive literal alternates being above or below the negative literal. We represent two variables x and y to illustrate how the corresponding gadgets are not interfering.

Say, one places a guard at vertex $v^i_{\overline{x}}$ to see (among other things) the vertex $d^i_{\overline{x}}$. Now, d^i_x and $d^i_{x,\overline{x}}$ remain to be seen. The only way of guarding them with one guard is to place it at vertex $v^{i+1}_{\overline{x}}$. Indeed, only vertices on the chunk T_{i+1} can possibly see both. But the vertices higher than $v^{i+1}_{\overline{x}}$ cannot see them because their visibility is blocked by $v^{i+1}_{\overline{x}}$ or a vertex to its right, while the vertices lower than $v^{i+1}_{\overline{x}}$ are too low to see the very bottom of those two triangular pockets. The same mechanism (too high $\rightarrow$ blocked visibility, too low $\rightarrow$ too flat angle) is used to ensure that the different variables do not interfere.

Symmetrically, the only vertex seeing both $d^i_{x,\overline{x}}$ and $d^i_{\overline{x}}$ is v^{i+1}_x. So, placing a guard at v^i_x forces to place the other guard at v^{i+1}_x. The chosen literal goes from being above (resp. below) in chunk T_i to being below (resp. above) in chunk T_{i+1}. Each d-vertex (i.e., vertex of the form $d^{\bullet}_{\bullet}$) has its visibility dominated in the one of a v-vertex (of the form $v^{\bullet}_{\bullet}$). Indeed, the visibility of d^i_x and $d^i_{\overline{x}}$ is contained in the visibilty of v^i_x and $v^i_{\overline{x}}$, respectively, while $d^i_{x,\overline{x}}$ has its visibility dominated by the one of v^{i+1}_x or $v^{i+1}_{\overline{x}}$ (what a vertex sees from below in a rectilinear terrain is irrelevant). Each non v-vertex has its visibility contained in the one of a v-vertex. Seeing the d-vertices with v-vertices is enough to see the entire subterrain/chunk. Thus, the problem can be seen as a red-blue domination: taking v-vertices (red) to dominate the d-vertices (blue). The red-blue visibility graph corresponding to the propagation of variable assignments is shown on Figure 8. The only way of guarding the $3z$ d-vertices on chunk T^i (corresponding to z variables) with a budget of z guards on T^i and z guards on T^{i+1} is to place z guards on v-vertices of chunk T_i and z guards on v-vertices of chunk T_{i+1} in a consistent way: the assignment of each variable is preserved.

We also need an alternative way of propagating truth assignments such that the chosen literal stays above or stays below on its respective chunk. This gadget is called *inverter*. It requires an extra guard compared to the usual propagation. The inverter gadget allows us to position the three literals of the clause to check and delete at the right spots.

It consists of a right triangular pocket whose bottom vertex is $d^i_{x,\overline{x}}$ surrounded by two rectangular pockets whose bottom vertices e^i_x, f^i_x and $e^i_{\overline{x}}, f^i_{\overline{x}}$ are only seen among the v-vertices by v^{i+1}_x, v^i_x and $v^{i+1}_{\overline{x}}, v^i_{\overline{x}}$, respectively. On top of the rectangular pockets, g^i_x sees both $e^i_{\overline{x}}$ and

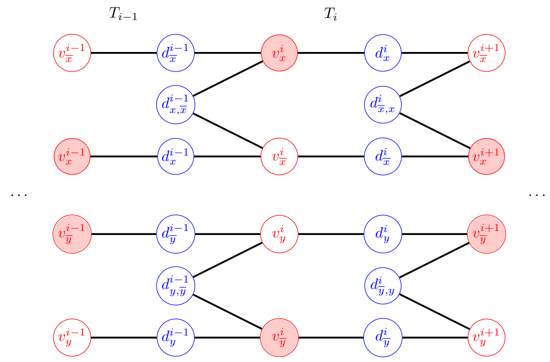

Figure 8 The red-blue domination graph for variable-assignment propagation.

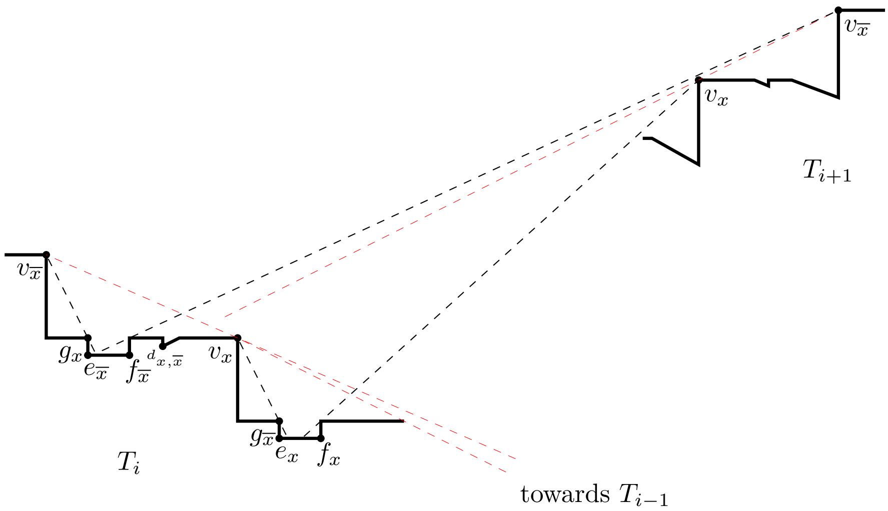

Figure 9 The inverter gadget. We omit the superscripts i and $i+1$. If a guard should be placed on at least one vertex among $v_{\overline{x}}^{\ell}$ and v_x^{ℓ} (for $\ell \in \{i, i+1\}$), then the two ways of seeing the four vertices $e_{\overline{x}}^i$, $f_{\overline{x}}^i$, e_x^i, f_x^i with three guards are $\{v_{\overline{x}}^i, g_{\overline{x}}^i, v_{\overline{x}}^{i+1}\}$ and $\{v_x^i, g_x^i, v_x^{i+1}\}$.

$f_{\overline{x}}^i$, whereas $g_{\overline{x}}^i$ sees both e_x^i and f_x^i. Actually, g_ℓ^i is only one of the four vertices seeing both e_ℓ^i and f_ℓ^i (which includes e_ℓ^i and f_ℓ^i themselves). We choose g_ℓ^i as a representative of this class. What matters to us is that the four vertices seeing both e_ℓ^i and f_ℓ^i do not see anything more than the rectangular pocket; the other parts of the terrain that they might guard are seen by any v-vertex on chunk T_{i+1} anyway.

The pockets are designed so that v_x^i and v_x^{i+1} (resp. $v_{\overline{x}}^i$ and $v_{\overline{x}}^{i+1}$) together see the whole edge $e_x^i f_x^i$ (resp. $e_{\overline{x}}^i f_{\overline{x}}^i$) and therefore the entire pocket. Again, the only two v-vertices to see $d_{x,\overline{x}}^i$ are v_x^{i+1} and $v_{\overline{x}}^{i+1}$. The e- and f-vertices are added to the blue vertices and the g-vertices are added to the red vertices, since the latter sees more than the former, and since seeing the e- and f-vertices are sufficient to also see the g-vertices. The red-blue domination graph is depicted on Figure 10.

Guarding $d_{x,\overline{x}}^{i-1}$ (resp. guarding $d_{x,\overline{x}}^i$) requires to take one v-vertex among v_x^i, $v_{\overline{x}}^i$ (resp. v_x^{i+1}, $v_{\overline{x}}^{i+1}$). Note that if one makes two inconsistent choices such as placing guards at v_x^i and $v_{\overline{x}}^{i+1}$ (or $v_{\overline{x}}^i$ and v_x^{i+1}), then it is not possible to see both rectangular pockets with one extra guard. Whereas, placing three guards at v_x^i, g_x^i, v_x^{i+1} or $v_{\overline{x}}^i, g_{\overline{x}}^i, v_{\overline{x}}^{i+1}$ would cover both rectangular pockets; hence the propagation of the truth assignment.

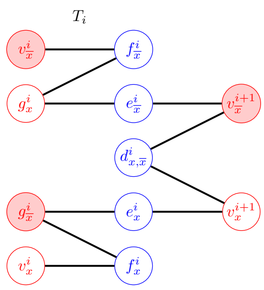

Figure 10 The red-blue domination graph for the inverter gadget.

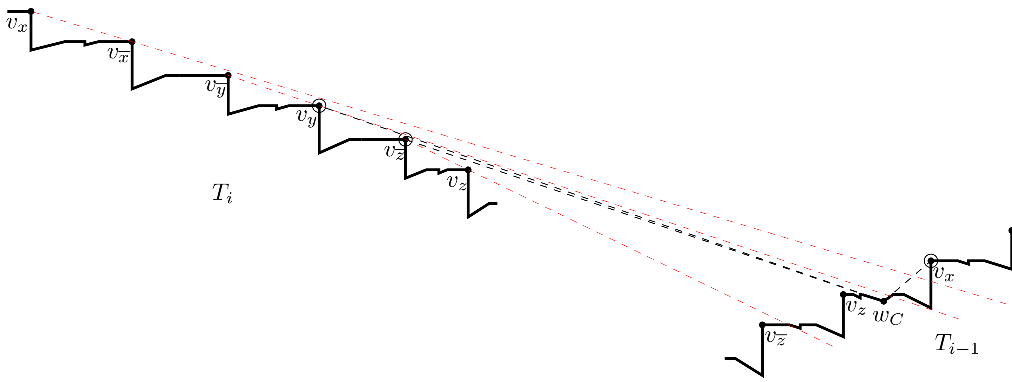

Figure 11 The downward clause gadget for $C = x \vee y \vee \neg z$. We use the usual propagation for variables x and z. Variable y disappears from T_{i-1} and downward. The inverters have been used to place, on T_i, the literals of C at positions 1, 4, and 5. Vertex w_C is seen only by v^i_y, $v^i_{\overline{z}}$, and v^{i-1}_x (circled); hence it is seen if and only if the chosen assignment satisfies C.

So far, the gadgets that we presented can be used *going up* along the chunks of positive index as well as *going down* along the chunks of negative index. For the clause gadgets, we will have to distinguish the *downward clause gadget* when we are below T_0 (and going down) and the *upward clause gadget* when we are above T_0 (and going up). The reason we cannot design a single gadget for both situations is that the middle variable which needs be deleted is in one case, in the lower chunk, and in the other case, in the higher chunk.

To check a clause downward on three consecutive variables x, y, z, we place on chunk T_i, thanks to a preliminary use of inverter gadgets, the three literals satisfying the clause at the relative positions 1, 4, and 5 when the six literals of x, y, z are read from top to bottom. Figure 11 shows the downward clause gadget for the clause $x \vee y \vee \neg z$. On chunk T_{i-1} just below, we find the usual encoding of variables x and z, which propagates the truth assignment of those two variables. The variable gadget of y is replaced by the right triangular pocket whose bottom is $d^{i-1}_{y,\overline{y}}$, and a general triangular pocket whose bottom w_C is only seen by the v-vertices $v^{i-1}_{\ell_1}$ (on chunk T_{i-1}), and $v^i_{\ell_2}$ and $v^i_{\ell_3}$ (on chunk T_i), with $C = \ell_1 \vee \ell_2 \vee \ell_3$. On chunk T_{i-1} and below, no v-vertex corresponding to variable y can be found.

Hence, vertex w_C is only guarded if the choices of the guards at the v-vertices correspond to an assignment satisfying C. The terrain visible to w_C is also covered by $v^{i-1}_{\ell_1}$, hence it is a blue vertex. The red-blue domination graph associated to a downward clause is represented on Figure 12.

To check a clause upward on three consecutive variables x, y, z, we place on chunk T_i, thanks to a preliminary use of inverter gadgets, the three literals satisfying the clause at the

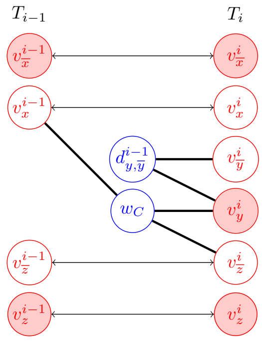

Figure 12 The red-blue domination graph for the downward clause gadget for $C = x \vee y \vee \neg z$. The double arcs symbolize that, due to the propagator, the variable-assignment of x and z should be the same between T_i and T_{i-1}. The only assignment that does not dominate w_C is $\overline{x}$, $\overline{y}$, z, as it should.

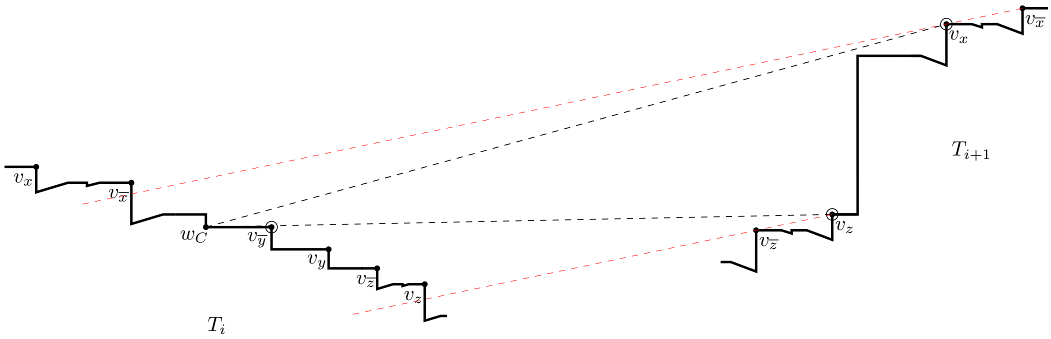

Figure 13 The upward clause gadget for $C = x \vee \neg y \vee z$. We use the usual propagation for variables x and z. Variable y disappears from T_{i+1} and upward. The inverters have been used to place, on T_i, the literals of C at positions 1, 3, and 6. Vertex w_C is seen only by $v_{\overline{y}}^i$, v_x^{i+1}, and v_z^{i+1} (circled); hence it is seen if and only if the chosen assignment satisfies C.

relative positions 1, 3, and 6 when the six literals of x, y, z are read from top to bottom. We exclude the three right triangular pockets for the encoding of the middle variable y. At the same altitude as the v-vertex corresponding to the literal of y satisfying the clause, we have a designated vertex w_C. On the chunk T_{i+1}, we find the usual encoding of variables x and z, which propagates the truth assignment of those two variables, but the encoding of variable y is no longer present (in this chunk and in all the chunks above). Figure 13 shows the upward clause gadget for the clause $x \vee \neg y \vee z$.

Vertex w_C is only seen by the v-vertices $v_{\ell_2}^i$ (on chunk T_i) and $v_{\ell_1}^{i+1}$ and $v_{\ell_3}^{i+1}$ (on chunk T_{i+1}), where $C = \ell_1 \vee \ell_2 \vee \ell_3$. The particularity of two consecutive chunks encoding an upward clause gadget is that T_i is not entirely below T_{i+1}. In fact, all the encodings of variables above y on chunk T_{i+1} are above all the encodings of variables above y on chunk T_i. The latter are above all the encodings of variables below y on chunk T_{i+1}, which are, in turn, above all the encodings of variables below y on chunk T_i. Again, vertex w_C is only guarded if the choices of the guards at the v-vertices correspond to an assignment satisfying C, as depicted in Figure 14.

Finally, we design upward and downward variable deletion gadgets; the description of these gadgets can be found in the full version [3]. The reader can just think of them as simplifications of the clause gadgets where vertex w_c is suppressed. This ends the list of gadgets.

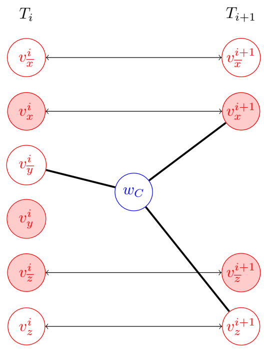

Figure 14 The red-blue domination graph for the upward clause gadget for $C = x \vee \neg y \vee z$. The double arcs symbolize that, due to the propagator, the variable-assignment of x and z should be the same between T_i and T_{i+1}. The only assignment that does not dominate w_C is $\overline{x}$, y, $\overline{z}$, as it should.

The gadgets are assembled as in the reduction of King and Krohn. From the initial chunk T_0 and going up (resp. going down), one realizes step by step (chunk by chunk) the elementary operations to check the clauses of $\mathcal{C}^+$ (resp. $\mathcal{C}^-$) in the order $C_1^+, C_2^+, \dots C_s^+$ (resp. $C_1^-, C_2^-, \dots C_{m-s}^-$) including propagation, inversion of literals, upward clause checking (resp. downward clause checking), and upward variable deletion (resp. downward variable deletion). Each chunk has $O(n)$ vertices. Each clause takes $O(1)$ chunks to be checked. So the total number of chunks is $O(m) = O(n)$ and the total number of vertices is $O(n^2)$.

We call *total budget* the total number of guards allowed. The total budget is fixed as one per right triangular pocket, two per general triangular pocket, one per variable encoding (including the slightly different one at inverters and the one just before an upward deletion), and one extra per inverter. Note that the lone $d^\bullet_{x,\overline{x}}$ in a downward clause gadget or a downward deletion gadget does not count as a variable gadget and does not increase the budget. To give an unambiguous definition of the number of variable encodings, we count the number of pairs i, x such that the vertices v^i_x and $v^i_{\overline{x}}$ exist.

We explained why the guards inside the triangular pockets can be placed (and the budget reduced). The correctness of the reduction is similar to King and Krohn's. The d-vertices force the placement of at least one guard in each variable encoding. We argued this to be sufficient to see all the right triangular and rectangular pockets if and only if the variable assignments are consistent between two consecutive chunks (by completing with guards g^i_ℓ at each inverter where ℓ is the literal chosen to be true). The terrain is entirely seen whenever the m general triangular pockets corresponding to the m clauses are all guarded, which happens if and only if the *truth assignment* chosen on chunk T_0 satisfies all the clauses.

This shows that Orthogonal Terrain Guarding and Dominating Set on visibility graph of rectilinear terrains are NP-hard. Recall that the continuous variant of Orthogonal Terrain Guarding is equivalent to its discrete counterpart. Membership in NP of all those variants is therefore trivial. What is left to prove is that Dominating Set on the visibility graph of *strictly* rectilinear terrains is NP-hard. Our reduction almost directly extends to this variant. The only issue is with the general triangular pocket gadget. Indeed, when the two guards are placed inside the pocket, all the internal vertices are guarded. In Orthogonal Terrain Guarding, one still needed to see the interior of the tiny top horizontal edge. This is no longer required in Dominating Set. Observe that the general triangular pocket is only used in the downward clause gadget. In the full version [3], we explain how we can make the downward clause gadget without the general triangular pocket.

3 ETH-Hardness of (Orthogonal) Terrain Guarding

We now explain how to turn those quadratic reductions into linear reductions by taking a step back. This step back is the reduction from 3-SAT to Planar 3-SAT by Lichtenstein [22], or rather, the instances of Planar 3-SAT it produces. The idea of Lichtenstein in this classic paper is to replace each intersection of a pair of edges in the incidence graph of the formula by a constant-size planar gadget, called crossover gadget. Using the sparsification of Impagliazzo et al. [15], even instances of 3-SAT with a linear number of clauses cannot be solved in subexponential time, under the ETH. Hence, the number of edges in the incidence graph of the formula can be assumed to be linear in the number N of variables. Thus there are at most a quadratic number $n = \Theta(N^2)$ of intersections; which implies a replacement of the intersections by a quadratic number n of constant-size crossover gadgets. We cannot expect to improve over the reduction of Lichtenstein since there is a matching algorithm solving Planar 3-SAT in time $2^{O(\sqrt{n})}$.

What we will do instead is to reduce the number of chunks that we actually need *and* to get a better upper bound of the number of vertices in a large fraction of the chunks. In the reduction by King and Krohn, each single clause incurs a constant number of chunks: to place the literals at the right position and to check the clause. The only requirement for a clause to be checked is that it operates on consecutive variables (discarding the deleted variables of the linear order). Therefore, nothing prevents us from checking several clauses *in parallel* if they happen to be on disjoint and consecutive variables.

A first observation is that the $\Theta(n)$ clauses of the crossover gadgets can be checked in parallel with only $O(1)$ chunks. Indeed, the constant number of clauses within each crossover gadget operates on pairwise-disjoint sets of variables. A second observation is that in all the remaining chunks only $\Theta(N)$ variables and $\Theta(N)$ clauses are left to be checked: they correspond to the original variables and clauses of the sparse 3-SAT instances. Therefore, the total number of vertices needed for the terrain is $O(1) \times \Theta(n) + \Theta(N) \times \Theta(N) = \Theta(n) + \Theta(N^2) = \Theta(n)$.

Thus, assuming the ETH, the $2^{O(\sqrt{n} \log n)}$ algorithm for guarding terrains [1] is optimal up to the logarithmic factor in the exponent for both Orthogonal Terrain Guarding and Terrain Guarding.

4 Perspectives

We have shown that Orthogonal Terrain Guarding is NP-complete, as well as its variants. We have presented a generic way of tightening quadratic reductions from Planar 3-SAT to linear. This applies to Terrain Guarding and Orthogonal Terrain Guarding and establishes that the existing $2^{\tilde{O}(\sqrt{n})}$-time algorithm is essentially optimal under the ETH, up to logarithmic factors in the exponent.

The principal remaining open questions concern the parameterized complexity of terrain guarding.

- (1) Is Terrain Guarding FPT parameterized by the number of guards?
- (2) Is Orthogonal Terrain Guarding FPT parameterized by the number of guards?

References

1 Pradeesha Ashok, Fedor V. Fomin, Sudeshna Kolay, Saket Saurabh, and Meirav Zehavi. Exact algorithms for terrain guarding. In *33rd International Symposium on Computational*

Geometry, SoCG 2017, July 4-7, 2017, Brisbane, Australia, pages 11:1–11:15, 2017. doi:10.4230/LIPIcs.SoCG.2017.11.

2 Boaz Ben-Moshe, Matthew J. Katz, and Joseph S. B. Mitchell. A constant-factor approximation algorithm for optimal 1.5d terrain guarding. *SIAM J. Comput.*, 36(6):1631–1647, 2007. doi:10.1137/S0097539704446384.

3 Édouard Bonnet and Panos Giannopoulos. Orthogonal terrain guarding is NP-complete. *CoRR*, abs/1710.00386, 2017. arXiv:1710.00386.

4 Édouard Bonnet and Tillmann Miltzow. Parameterized hardness of art gallery problems. In *24th Annual European Symposium on Algorithms, ESA 2016, August 22-24, 2016, Aarhus, Denmark*, pages 19:1–19:17, 2016. doi:10.4230/LIPIcs.ESA.2016.19.

5 Édouard Bonnet and Tillmann Miltzow. An approximation algorithm for the art gallery problem. In *33rd International Symposium on Computational Geometry, SoCG 2017, July 4-7, 2017, Brisbane, Australia*, pages 20:1–20:15, 2017. doi:10.4230/LIPIcs.SoCG.2017.20.

6 Timothy M. Chan and Sariel Har-Peled. Approximation algorithms for maximum independent set of pseudo-disks. *Discrete & Computational Geometry*, 48(2):373–392, 2012. doi:10.1007/s00454-012-9417-5.

7 Kenneth L. Clarkson and Kasturi R. Varadarajan. Improved approximation algorithms for geometric set cover. *Discrete & Computational Geometry*, 37(1):43–58, 2007. doi:10.1007/s00454-006-1273-8.

8 Ajay Deshpande, Taejung Kim, Erik D. Demaine, and Sanjay E. Sarma. A pseudopolynomial time O (log n)-approximation algorithm for art gallery problems. In *Algorithms and Data Structures, 10th International Workshop, WADS 2007, Halifax, Canada, August 15-17, 2007, Proceedings*, pages 163–174, 2007. doi:10.1007/978-3-540-73951-7_15.

9 Alon Efrat and Sariel Har-Peled. Guarding galleries and terrains. *Inf. Process. Lett.*, 100(6):238–245, 2006. doi:10.1016/j.ipl.2006.05.014.

10 Stephan Eidenbenz. Inapproximability results for guarding polygons without holes. In *Algorithms and Computation, 9th International Symposium, ISAAC '98, Taejon, Korea, December 14-16, 1998, Proceedings*, pages 427–436, 1998. doi:10.1007/3-540-49381-6_45.

11 Stephan Eidenbenz, Christoph Stamm, and Peter Widmayer. Inapproximability results for guarding polygons and terrains. *Algorithmica*, 31(1):79–113, 2001. doi:10.1007/s00453-001-0040-8.

12 Khaled Elbassioni. Finding small hitting sets in infinite range spaces of bounded vc-dimension. In *33rd International Symposium on Computational Geometry, SoCG 2017, July 4-7, 2017, Brisbane, Australia*, pages 40:1–40:15, 2017. doi:10.4230/LIPIcs.SoCG.2017.40.

13 Khaled M. Elbassioni, Erik Krohn, Domagoj Matijevic, Julián Mestre, and Domagoj Severdija. Improved approximations for guarding 1.5-dimensional terrains. *Algorithmica*, 60(2):451–463, 2011. doi:10.1007/s00453-009-9358-4.

14 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *Journal of Computer and System Sciences*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.

15 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001. doi:10.1006/jcss.2001.1774.

16 James King. A 4-approximation algorithm for guarding 1.5-dimensional terrains. In *LATIN 2006: Theoretical Informatics, 7th Latin American Symposium, Valdivia, Chile, March 20-24, 2006, Proceedings*, pages 629–640, 2006. doi:10.1007/11682462_58.

17 James King and David G. Kirkpatrick. Improved approximation for guarding simple galleries from the perimeter. *Discrete & Computational Geometry*, 46(2):252–269, 2011. `doi:10.1007/s00454-011-9352-x`.

18 James King and Erik Krohn. Terrain guarding is NP-hard. *SIAM J. Comput.*, 40(5):1316–1339, 2011. `doi:10.1137/100791506`.

19 David G. Kirkpatrick. An o(lg lg opt)-approximation algorithm for multi-guarding galleries. *Discrete & Computational Geometry*, 53(2):327–343, 2015. `doi:10.1007/s00454-014-9656-8`.

20 Erik Krohn, Matt Gibson, Gaurav Kanade, and Kasturi R. Varadarajan. Guarding terrains via local search. *JoCG*, 5(1):168–178, 2014. URL: `http://jocg.org/index.php/jocg/article/view/128`.

21 Erik Krohn and Bengt J. Nilsson. Approximate guarding of monotone and rectilinear polygons. *Algorithmica*, 66(3):564–594, 2013. `doi:10.1007/s00453-012-9653-3`.

22 David Lichtenstein. Planar formulae and their uses. *SIAM J. Comput.*, 11(2):329–343, 1982. `doi:10.1137/0211025`.

23 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Lower bounds based on the exponential time hypothesis. *Bulletin of the EATCS*, 105:41–72, 2011. URL: `http://albcom.lsi.upc.edu/ojs/index.php/beatcs/article/view/96`.

24 Nabil H. Mustafa and Saurabh Ray. Improved results on geometric hitting set problems. *Discrete & Computational Geometry*, 44(4):883–895, 2010. `doi:10.1007/s00454-010-9285-9`.