Orthogonal Terrain Guarding is NP-complete*

Édouard Bonnet† *and Panos Giannopoulos*‡

Abstract. A terrain is an x-monotone polygonal curve, that is, every vertical line crosses the curve at most once. In the Terrain Guarding problem, a special case of the famous art gallery problem, one has to place at most k guards on the vertices of a n-vertex terrain, in order to fully see it. In 2010, King and Krohn showed that Terrain Guarding is NP-hard [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 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 to obtain an improved ETH lower bound[1] of $2^{\Omega(n^{1/3})}$ by refining the quadratic reduction from Planar 3-SAT into a cubic reduction from 3-SAT. This works for both Orthogonal Terrain Guarding and Terrain Guarding.

1 Introduction

Given $p_1 = (x_1, y_1), p_2 = (x_2, y_2), \ldots, p_n = (x_n, y_n) \in \mathbb{R}^2$, n distinct points of the plane, such that $x_1 \leqslant x_2 \leqslant \ldots \leqslant x_n$, an *x-monotone polygonal chain* or *terrain* is defined as the sequence of vertices and edges $p_1, p_1p_2, p_2, \ldots, p_{n-1}, p_{n-1}p_n, p_n$. Each point p_i is called a *vertex* of the terrain. 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. Terrain Guarding, a natural restriction of the well-known art gallery problem,

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

† *Univ Lyon, CNRS, ENS de Lyon, Université Claude Bernard Lyon 1, LIP UMR5668, France*, edouard.bonnet@ens-lyon.fr

‡ *giCentre, Department of Computer Science, City University of London, EC1V 0HB, London, United Kingdom*, panos.giannopoulos@city.ac.uk

[1] In the conference version of this paper, we mistakenly claim a tighter lower bound.

asks, given an integer k, and a terrain $\mathcal{T}$, to guard it by placing at most k guards at vertices of $\mathcal{T}$. 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. In the literature, what we call (Orthogonal) Terrain Guarding is sometimes referred to as Discrete (Orthogonal) Terrain Guarding to emphasize the fact that guards can only placed at vertices. Henceforth, we consider it to be the main variant, and call it the *discrete variant*. Both problems (general and orthogonal) come with two other variants: the *continuous variant*, Continuous (Orthogonal) Terrain Guarding, where the guards can be placed anywhere on the edges of the terrain (and not only at the vertices), and the *graphic variant*, that is, Dominating Set in the visibility graphs of (strictly rectilinear) terrains. A *convex vertex* (resp. *reflex vertex*) of a terrain is a vertex p_i such that the angle formed by $p_{i-1}p_ip_{i+1}$ above the terrain is at most 180 degrees (resp. more than 180 degrees). The first and the last vertices of the terrain are convex by convention. A *convex edge* is an edge whose both endpoints are convex vertices. It is a folklore observation that for rectilinear terrains, the discrete and continuous variants coincide (see Observation 7 in Section 2). In particular, Observation 7 establishes the NP membership of the continuous variant (for orthogonal terrains). And, it allows us to 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 [16]. 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. [17]. The so-called *sparsification lemma* shows that, assuming the ETH, 3-SAT does not even admit a subexponential algorithm in the number of variables plus the number of clauses.

Theorem 1 (Impagliazzo and Paturi [16])**.** *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 [26].

Planar satisfiability. Planar 3-SAT was introduced by Lichtenstein [25] 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: $x_1, x_2, \ldots, x_n$; i.e., $x_i x_{i+1}$ is an edge (with index $i+1$ taken modulo n). This extra 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 the one 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 and
- (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.

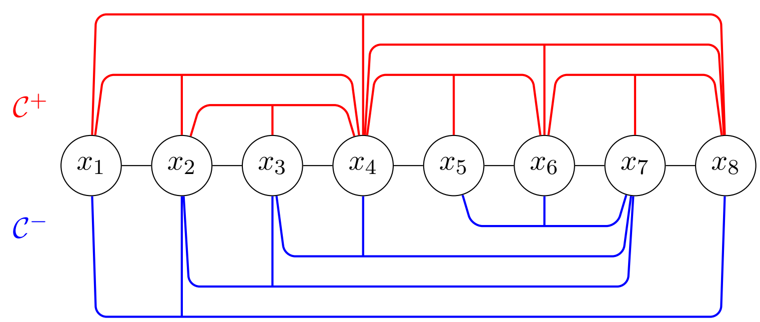

Figure 1: The bipartition $(\mathcal{C}^+, \mathcal{C}^-)$ of a Planar 3-SAT-instance. The three-legged arches represent the clauses. Here is a removal ordering for $\mathcal{C}^-$: remove the clause on x_5, x_6, x_7 and its middle variable x_6, remove the variable x_5, remove the clause on x_3, x_4,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

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. We mean the latter informal remark as a way to form an intuition on the difficulty of showing hardness for Terrain Guarding, but not as a formal barrier.

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.*

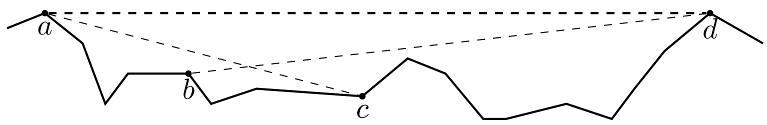

Figure 2: The order claim.

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 (together with its continuous and graphic variants) was shown NP-hard [21], and the membership in NP of Continuous Terrain Guarding was established [15]. Observe that the discrete and the graphic variants are immediately in NP. Terrain Guarding can be solved in time $n^{O(\sqrt{k})}$ [2]. This contrasts with the 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 [5], 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 $k^{O(k)}n^{O(1)}$ [2], 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 [19, 18, 8, 3, 14]. Eventually, a PTAS was found for Terrain Guarding (hence for Orthogonal Terrain Guarding) [23] using local search and an idea developed by Chan and Har-Peled [7] and Mustafa and Ray [27] 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. [2].

Again the situation is not nearly as good for the art gallery problem, whose point guard variant is $\exists\mathbb{R}$-complete [1] (hence unlikely to even be in NP). If holes are allowed in the polygon, the main variants of the art gallery problem are as hard as the Set Cover problem; hence an $o(\log n)$-approximation cannot exist unless P=NP [12]. Eidenbenz also showed that a PTAS is unlikely in simple polygons [11]. For simple polygons, there is an $O(\log\log OPT)$-approximation [20, 22] 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 [6], building up on [10, 9]. Recently, a constant-approximation for Vertex Guard Art Gallery was announced [4]. If a small fraction of the polygon can be left unguarded there is again an $O(\log OPT)$-approximation [13]. A constant-factor approximation is known for *monotone polygons* [24], 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 following open question is raised by Ashok et al. [2]:

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

In the conclusion of the same paper, the authors observe that the construction of King and Krohn [21] 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 $2^{O(\sqrt{n}\log n)}$ [2]. Therefore, there is 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*?*

Our contributions. In Section 2, we address Open question 1 by showing:

Theorem 3. Orthogonal Terrain Guarding *is NP-complete.*

To achieve that result, we design a rectilinear subterrain with a constant number of vertices which simulates a triangular pocket surrounded by two horizontal segments. This enables us to adapt the reduction of King and Krohn [21] to rectilinear terrains. Our orthogonal gadgets make an extensive use of the *triangular pockets*. A slight tuning of the construction also shows:

Theorem 4. Dominating Set *in the visibility graphs of strictly rectilinear terrains is NP-complete.*

In Section 3, we show how to make cubic reductions from 3-SAT by refining the quadratic reductions from Planar 3-SAT:

Theorem 5. Terrain Guarding *and* Orthogonal Terrain Guarding *both require time* $2^{\Omega(n^{1/3})}$*, unless the ETH fails.*

This improves the lower bound of $2^{\Omega(n^{1/4})}$ implicitly proven by King and Krohn [21], but does not quite resolve Open question 2.

2 Orthogonal Terrain Guarding is NP-complete

We start this section with generic observations on guarding orthogonal terrains.

Observation 6 (folklore)**.** *A guarding set remains so when we change the visibility notion to* guards only see at their y-coordinate and below. *In more intuitive terms, in an orthogonal terrain, what a guard sees strictly above itself is irrelevant.*

Proof. Strictly above itself, a guard g can only see a point on a vertical edge minus its lower endpoint v. The guard responsible for seeing v (possibly g) sees this vertical edge entirely. □

Observation 7 (folklore)**.** *For any n-vertex orthogonal terrain $\mathcal{T}$ and integer k, $(\mathcal{T}, k)$ is a positive instance for* Orthogonal Terrain Guarding *if and only if it is for* Continuous Orthogonal Terrain Guarding*. Furthermore, a solution for the latter can be transformed into a solution for the former in polynomial time.*

Proof. A solution for Orthogonal Terrain Guarding is obviously also a solution for Continuous Orthogonal Terrain Guarding. We thus show that a guarding set for Continuous Orthogonal Terrain Guarding can be moved to the vertices, without increasing its cardinality nor changing the fact that it guards the entire terrain. Every guard g lying on the interior of a vertical edge can be moved to the higher endpoint of the edge. Indeed, any point seen by g is also seen by this new and higher guard. Every guard h lying on the interior of a horizontal edge can be moved to any one of its endpoints. Indeed, any point seen by h and at the same level of h or below, is also seen by the new guard (in fact h sees only points with the same y-coordinate as itself). And by Observation 6, what a guard sees strictly above itself is irrelevant in orthogonal terrains. □

Consequently, Continuous Orthogonal Terrain Guarding is in NP. The discrete and graphic variants are in NP since one can guess which vertices to put a guard on. Then, checking if a set is guarding the whole terrain can be done in polynomial time, for instance, by building the following subdivision. We add a vertex at every intersection between an edge of the terrain and a line defined by two originally-present vertices. We then subdivide each edge once. By that, we mean that we create a new vertex at the midpoint of the edge, that we link to both endpoints (and delete the original edge). Now, guarding all the vertices with the original vertices is equivalent to guarding the whole terrain. Building the subdivision and checking the visibility of two vertices can be done in polynomial time in the description (number of vertices times number of digits to represent one of them).

King and Krohn's reduction. King and Krohn give a reduction with a quadratic blow-up from Planar 3-SAT to Terrain Guarding [21]. 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. [2] 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 $2^{O(\sqrt{n})}$.

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).

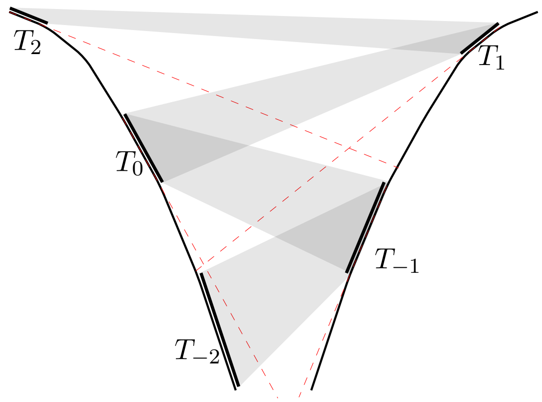

Figure 3: The V-shaped terrain and its ordered chunks. The chunk T_i only sees parts of chunks T_{i-1} and T_{i+1}. The *initial* chunk T_0 contains an encoding of each variable. Below this level (chunks with a negative index), we will check the clauses of $\mathcal{C}^-$. Above this level (chunks with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

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 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 $x_1, x_2, \ldots, x_n$. Let $C_1^+, C_2^+, \ldots, C_s^+$ (resp. $C_1^-, C_2^-, \ldots, C_{m-s}^-$) 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 $C_1^-, C_2^-, \ldots, C_{m-s}^-$, while every chunk above T_0, i.e., with a positive index, will check *if the clauses of* C^+ *are satisfied* in the order $C_1^+, C_2^+, \ldots, C_s^+$. 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.

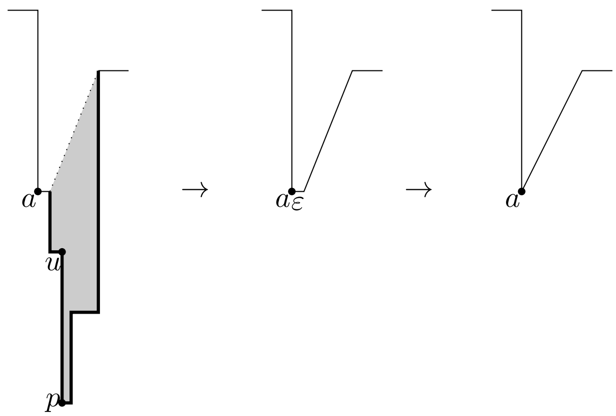

Figure 4: Simulation of a right trapezoid pocket and a right triangular pocket. The right triangular pocket is obtained from the right trapezoid by making the distance ε sufficiently small.

Triangular pocket simulation. The simulation of a *right trapezoid pocket* giving rise to the *right triangular pocket* is depicted in 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. More generally, by Observation 6, in rectilinear terrains, what a guard sees strictly above it can be safely ignored. 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 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. Let us precise that. We make the construction such that a is not aligned with two other vertices of the terrain. Therefore every vertex seeing a also sees a small interval to its right. We choose $\varepsilon > 0$ smaller than the minimum length of these intervals. The previous remarks show the following substitution lemma:

Lemma 8. *Guarding a terrain containing the orthogonal subterrain on the left of Figure 4 with $k+1$ guards is equivalent to guarding the same terrain in which the orthogonal subterrain is replaced by the right triangular pocket on the right of Figure 4 with k guards.*

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.

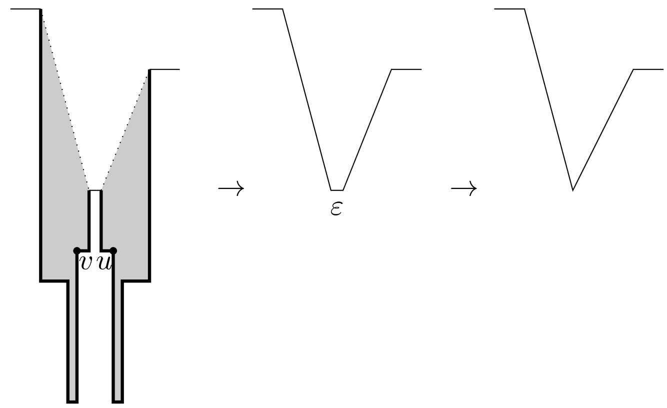

Figure 5: Simulation of a trapezoid pocket and a triangular pocket. The triangular pocket is obtained from the trapezoid by making the distance ε sufficiently small.

With the same idea, one can simulate a general triangular pocket as depicted in 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.

Lemma 9. *Guarding a terrain containing the orthogonal subterrain on the left of Figure 5 with $k+2$ guards is equivalent to guarding the same terrain in which the orthogonal subterrain is replaced by the triangular pocket on the right of Figure 5 with k guards.*

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.

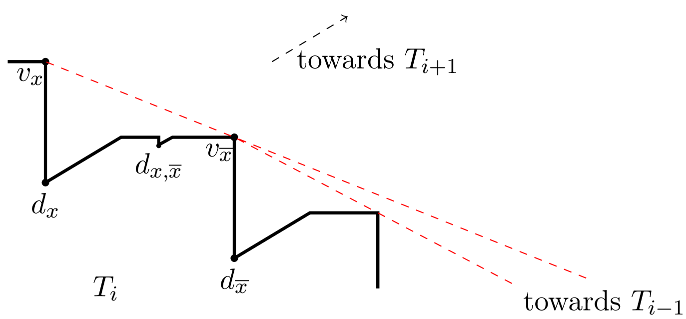

Figure 6: A variable gadget. We omit the superscript i on all the labels. Placing a guard at vertex v_x to see d_x corresponds to setting variable x to true, while placing it at vertex $v_{\overline{x}}$ to see $d_{\overline{x}}$ corresponds to setting x to false. Both v_x^{i+1} and $v_{\overline{x}}^{i+1}$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

Variable gadget and propagation along the chunks. The variable gadget is depicted in 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*).

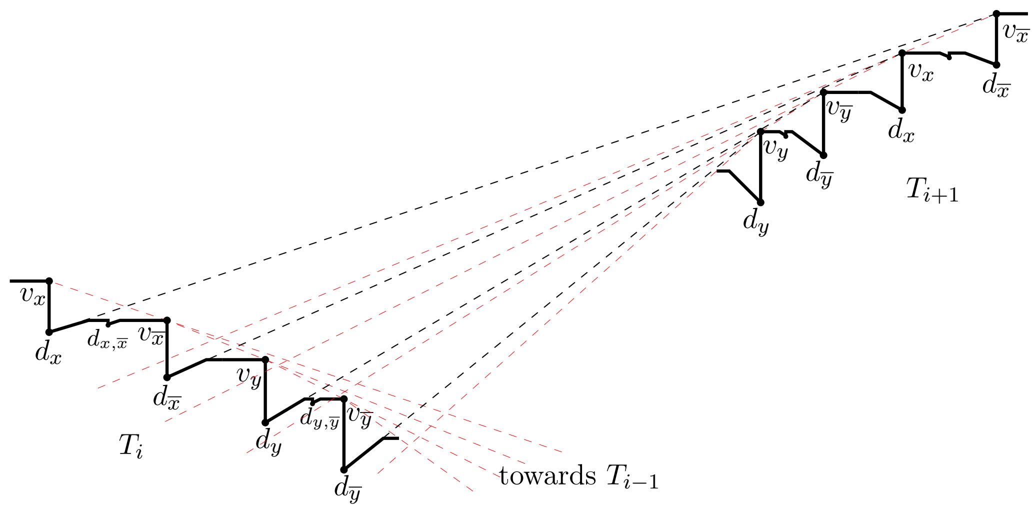

Figure 7: Propagating variable assignments upward and downward. Note that the positive literal alternates being above or below the negative literal. We represent two variables x and y to illustrate how the corresponding gadgets are not interfering. In all the figures, red (thinner) dashed lines delineate the visibility cone of a vertex (typically blocked by a close-by vertex), while the black (bolder) dashed lines show an important visibility relation.

Figure 7 represents the propagation of a variable assignment from one chunk to the next chunk. 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. 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. Observe that the chosen literal goes from being above (resp. below) in chunk T_i to being below (resp. above) in chunk T_{i+1}. We call a vertex of the form $d^{\bullet}_{\bullet}$, $e^{\bullet}_{\bullet}$, $f^{\bullet}_{\bullet}$, or $w_{\bullet}$, a *d-vertex*, and one of the form $v^{\bullet}_{\bullet}$ or $g^{\bullet}_{\bullet}$, a v-vertex. We will construct the terrain so that:

- each non v-vertex has its visibility essentially contained in the one of a v-vertex, and
- seeing all the d-vertices with v-vertices is enough to see the entire terrain.

By *essentially contained*, we mean that the potential regions that the non v-vertex sees and the v-vertex does not are irrelevant; in the sense that several other vertices (at least one of which is required in a solution) also see these regions. Hence, the problem will boil down to red-blue domination: selecting some v-vertices (red) to dominate all the d-vertices (blue). The red-blue visibility graph corresponding to the propagation of variable assignments is represented in Figure 8. It can be observed that:

Lemma 10. *The only way of guarding the $3z$ d-vertices on chunk T^i (corresponding to z variables) with $2z$ guards 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.*

Proof. The only red-blue dominating sets of size at most two of $\{d^i_x, d^i_{x,\overline{x}}, d^i_{\overline{x}}\}$ are $\{v^i_x, v^{i+1}_x\}$ and $\{v^i_{\overline{x}}, v^{i+1}_{\overline{x}}\}$, for every variable present on the chunks T_i and T_{i+1} (see Figure 8). This readily generalizes to z variables. Furthermore, this guards the entire subterrain associated to these variables. □

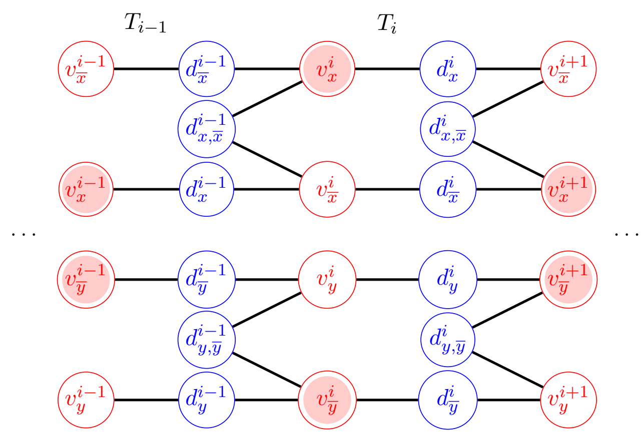

Figure 8: The red-blue domination graph for variable-assignment propagation. The v-vertices are in red, and the d-vertices, in blue. One should place guards at some red vertices to dominate all the blue vertices. In this example, the shaded vertices correspond to setting x to true and y to false, consistently performed over three consecutive chunks.

A set of guards is said *non-dominated* if for every i and x, the vertices v^i_x and $v^i_{\overline{x}}$ receive at least one guard. In other words, its intersection with $\{v^i_x, v^i_{\overline{x}}\}$ is of size at least one. The global construction will be such that for every vertices v^i_x and $v^i_{\overline{x}}$, there is a vertex $d^{i-1}_{x,\overline{x}}$. This will be helpful to get the following lemma:

Lemma 11. *Every guarding set can be turned into a no-larger non-dominated guarding set.*

Proof. Besides v^i_x and $v^i_{\overline{x}}$, only three other vertices guard $d^{i-1}_{x,\overline{x}}$: $d^{i-1}_{x,\overline{x}}$ itself and the two other vertices forming its right triangular pocket. The visibility of $d^{i-1}_{x,\overline{x}}$ is contained in the one of v^i_x or $v^i_{\overline{x}}$. The construction will be such that the two other vertices do not see anything important that v^i_x (or $v^i_{\overline{x}}$) does not see. They will see on the other side of the valley a fraction of a vertical edge minus its lower endpoint. So we can conclude by Observation 6. □

A set of guards is said *legal* if for every i and x, the vertices v_x^i and $v_{\overline{x}}^i$ receive exactly one guard. The number of guards allowed, together with Lemma 11, will enable us to focus on legal guard placements. We say that a legal placement is *consistent* if there is no pair of guards at v_x^i and $v_{\overline{x}}^{i'}$. Lemma 10 shows that legal consistent placements are the only ones that do not require more guards than the number of variable gadgets. Notice that a legal consistent placement naturally corresponds to a truth assignment of all the variables.

Inverter gadget. We also need an alternative way of propagating a variable assignment such that in the same-variable gadget of two consecutive chunks, the positive literal remains on top (or remains at the bottom). 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.

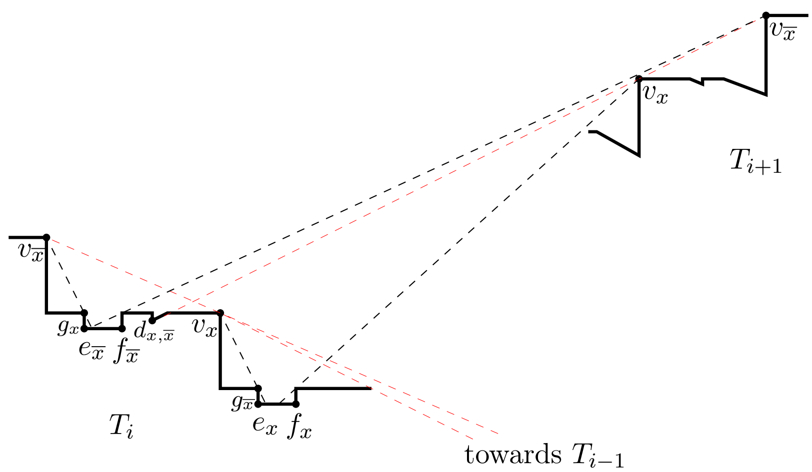

Figure 9: The inverter gadget. We omit the superscripts i and $i+1$. If a guard should be placed on at least one vertex among $v_{\overline{x}}^\ell$ and v_x^ℓ (for $\ell \in \{i, i+1\}$), then the two ways of seeing the four vertices $e_{\overline{x}}^i$, $f_{\overline{x}}^i$, e_x^i, f_x^i with three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

It consists of a right triangular pocket whose bottom vertex is $d_{x,\overline{x}}^i$ surrounded by two rectangular pockets whose bottom vertices e_x^i, f_x^i and $e_{\overline{x}}^i, f_{\overline{x}}^i$ are only seen among the v-vertices by v_x^{i+1}, v_x^i and $v_{\overline{x}}^{i+1}, v_{\overline{x}}^i$, respectively. On top of the rectangular pockets, g_x^i sees both $e_{\overline{x}}^i$ and $f_{\overline{x}}^i$, 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 in Figure 10.

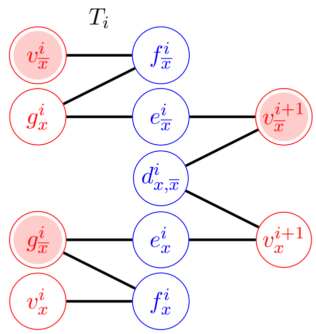

Figure 10: The red-blue domination graph for the inverter gadget.

Lemma 12. *The only legal ways to guard an inverter gadget of T_i with three guards is to place them consistently on v^i_x, g^i_x, v^{i+1}_x or $v^i_{\overline{x}}, g^i_{\overline{x}}, v^{i+1}_{\overline{x}}$.*

Proof. Recall that a legal guard placement is one that puts exactly one guard in $\{v^i_x, v^i_{\overline{x}}\}$ for every i and x. The two only ways of seeing both rectangular pockets with an extra guard is then to place the three guards at v^i_x, g^i_x, v^{i+1}_x or $v^i_{\overline{x}}, g^i_{\overline{x}}, v^{i+1}_{\overline{x}}$; hence the propagation of the truth assignment. □

We extend the notion of legal consistent assignment to the ones placing exactly one guard in $\{g^i_x, g^i_{\overline{x}}\}$ corresponding to the chosen literal.

Clause gadgets. 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.

Downward clause gadget. 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 the 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 among the v-vertices by $v^{i-1}_{\ell_1}$ (on chunk T_{i-1}) and $v^i_{\ell_2}$ and $v^i_{\ell_3}$ (on chunk T_i), where $C = \ell_1 \vee \ell_2 \vee \ell_3$. On chunk T_{i-1} and below, no v-vertex corresponding to variable y can be found.

Hence, the vertex w_C is only guarded if the choices of the guards at the v-vertices correspond to an assignment satisfying C. The vertex w_C has its visibility contained in the

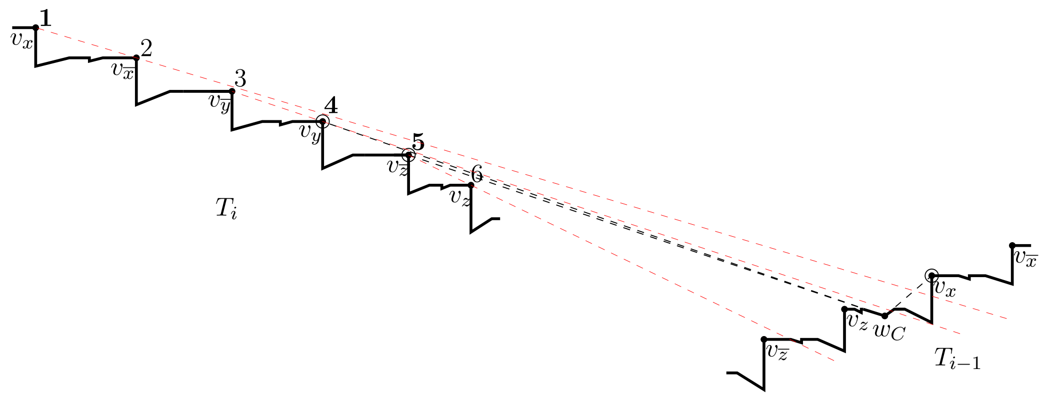

Figure 11: The downward clause gadget for $C = x \vee y \vee \neg z$. We use the usual propagation for variables x and z. The variable y disappears from T_{i-1} and downward. The inverters have been used to place, on T_i, the literals of C at positions 1, 4, and 5. The vertex w_C is seen only by v_y^i,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

one of a v-vertex, hence it is a blue vertex. The red-blue domination graph associated to a downward clause is represented in Figure 12.

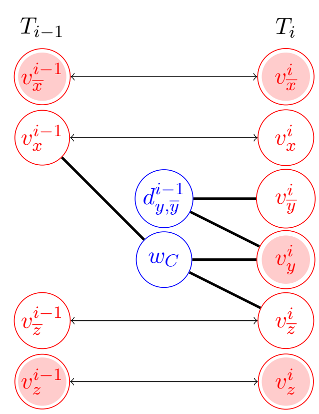

Figure 12: The red-blue domination graph for the downward clause gadget for $C = x \vee y \vee \neg z$. The double arcs symbolize that, due to the propagator, the variable-assignment of x and z should be the same between T_i and T_{i-1}. The only assignment that does not dominate w_C is $\overline{x}$,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

Lemma 13. *A legal consistent guard placement does not require an additional guard to see the whole downward clause gadget of C if and only if C is satisfied by the corresponding assignment.*

Proof. The only *new* vertex to see is w_C on chunk, say T_{i+1}. Let C be $\ell_1 \vee \ell_2 \vee \ell_3$. This will be done if at least one of $v_{\ell_1}^{i+1}, v_{\ell_2}^{i}, v_{\ell_3}^{i}$ contains a guard, which happens if and only if the legal consistent placement corresponds to an assignment satisfying C. $\square$

Upward clause gadget. 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 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 $x \vee \neg y \vee z$.

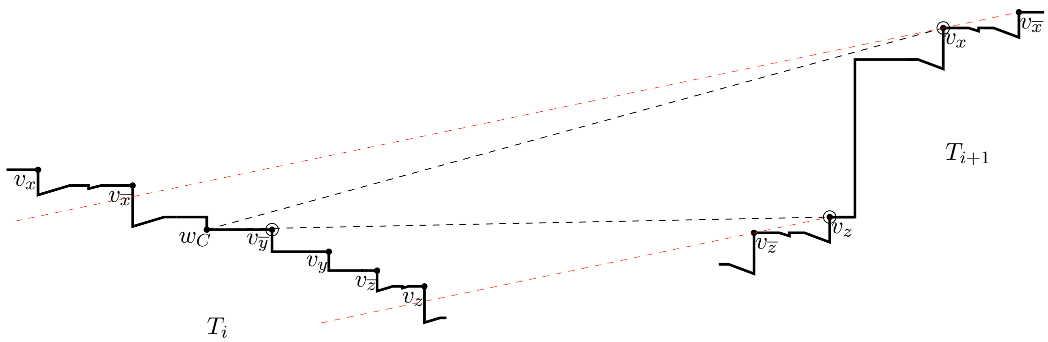

Figure 13: The upward clause gadget for $C = x \vee \neg y \vee z$. We use the usual propagation for variables x and z. The variable y disappears from T_{i+1} and upward. The inverters have been used to place, on T_i, the literals of C at positions 1, 3, and 6. The vertex w_C is seen only by $v^i_{\overline{y}}$,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

The vertex w_C is only seen among the v-vertices by $v^i_{\ell_2}$ (on chunk T_i) and $v^{i+1}_{\ell_1}$ and $v^{i+1}_{\ell_3}$ (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, the 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.

Lemma 14. *A legal consistent guard placement does not require an additional guard to see the whole upward clause gadget of C if and only if C is satisfied by the corresponding assignment.*

Proof. The only *new* vertex to see is w_C on chunk, say T_i. Let C be $\ell_1 \vee \ell_2 \vee \ell_3$. This will be done if at least one of $v^i_{\ell_2}, v^{i+1}_{\ell_1}, v^{i+1}_{\ell_3}$ contains a guard, which happens if and only if the legal consistent placement corresponds to an assignment satisfying C. □

Variable deletion gadgets. Finally, we design variable deletion gadgets. Recall that we sometimes need to remove a variable which does not appear in any clauses anymore (and was

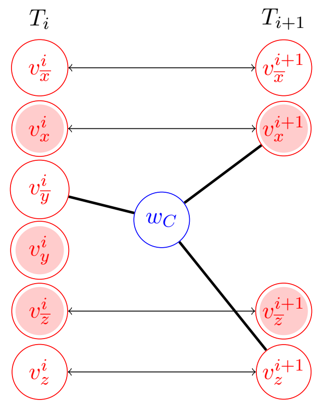

Figure 14: The red-blue domination graph for the upward clause gadget for $C = x \vee \neg y \vee z$. The double arcs symbolize that, due to the propagator, the variable-assignment of x and z should be the same between T_i and T_{i+1}. The only assignment that does not dominate w_C is $\overline{x}$,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

never a middle variable). As for clause gadgets, we have to distinguish *downward deletion gadget* and *upward deletion gadget*. Both gadgets can be thought as a simplification of the corresponding clause gadget where we flatten the region which should normally contain w_C.

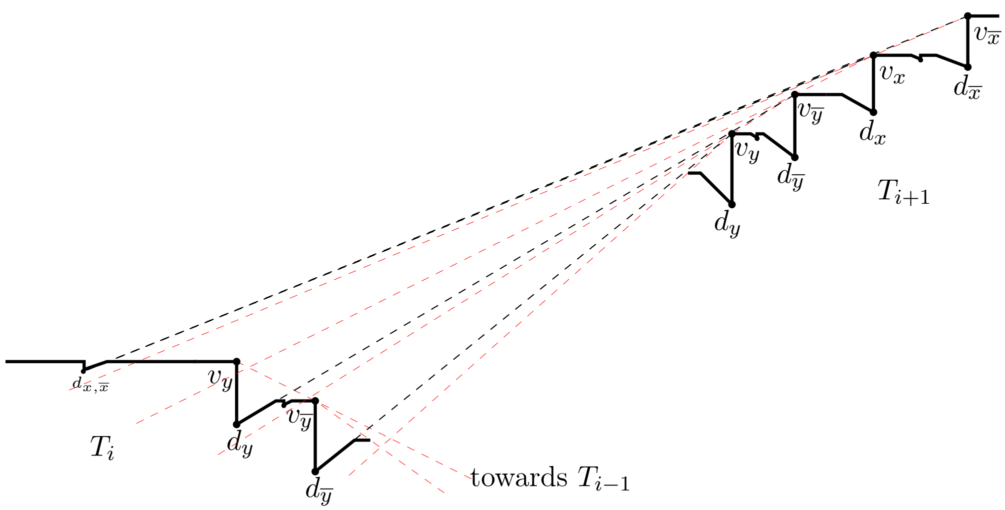

Figure 15: Downward deletion of the variable x (and propagation of the variable y). On chunk T_{i-1}, the encoding of variable x has totally disappeared: there is *not* even a $d^{i-1}_{x,\overline{x}}$.

On all the chunks below the downward deletion of a variable x, there is no encoding of variable x. And, on all the chunks above the upward deletion of a variable x, there is no encoding of variable x. The gadgets are represented in Figure 15 and Figure 16, respectively.

This ends the list of gadgets.

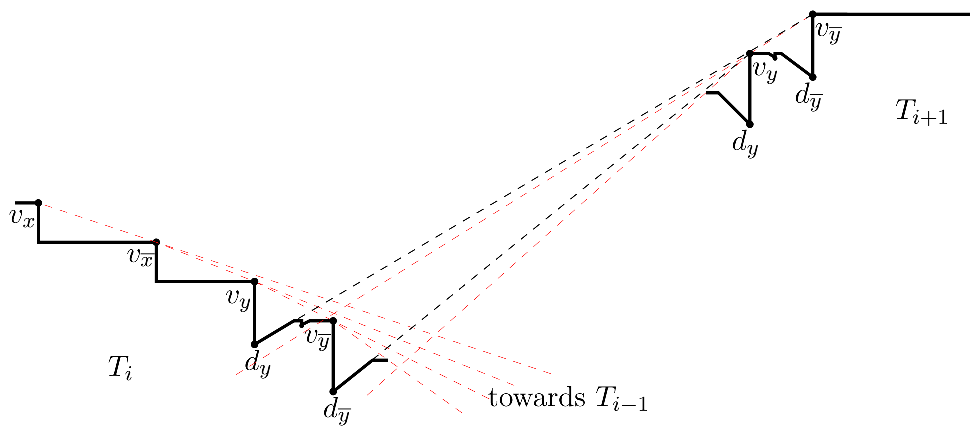

Figure 16: Upward deletion of the variable x (and propagation of the variable y). On chunk T_{i-1} is the usual encoding of variable x with three right triangular pockets.

Putting pieces together. 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 $C_1^+, C_2^+, \ldots C_s^+$ (resp. $C_1^-, C_2^-, \ldots C_{m-s}^-$) 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 $O(m) = O(n)$ (recall that by the sparsification lemma, we can assume that $m = \Theta(n)$) and the total number of vertices is $O(n^2)$.

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 (see encoding of variable x on chunk T_i in Figure 16), and one extra per inverter. Note that the lone $d^{\bullet}_{x,\overline{x}}$ at downward clause gadget and downward deletion do not count as variable gadget and they do 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_x^i and $v_{\overline{x}}^i$ exist.

So far, we did not mention the issue related to the encoding of the terrain. As in the construction of King and Krohn, we need flatter and flatter angles between two consecutive chunks as we go up the valley (see Figure 3). However, we only need these angles to be a bit flatter (*not*, for instance, twice as flat as the previous chunk interaction). Hence, as for their construction, we can place all the vertices of the terrain on an integer grid of polynomial size. This implies that we only need logarithmically long digits to represent each vertex.

Proof of Theorem 3. By Lemmas 8 and 9, we know that guards inside the triangular pockets can be placed (and the budget reduced accordingly). The correctness of the reduction is similar to the one by King and Krohn.

A solution to the Planar 3-SAT can be transformed into a solution for the Orthogonal Terrain Guarding-instance. We place a guard at every vertex v^i_ℓ and every vertex g^i_ℓ such that ℓ is a satisfied literal. This exactly fits the total budget. By Lemmas 10 and 12, all the variable and inverter gadgets are seen since we chose consistent literals. By Lemmas 13 and 14, all the vertices w_{C_j} are seen, since the assignment satisfies all the clauses. Thus, all the convex vertices of the terrain are seen, as well as all the convex edges $e^i_x f^i_x$ and $e^i_{\overline{x}} f^i_{\overline{x}}$. This implies that all the vertical edges and consequently all the reflex vertices are seen. One can then check that all the remaining horizontal edges are fully guarded.

We now focus on the reverse direction, namely: if there is a solution to the Orthogonal Terrain Guarding-instance, then there is an assignment satisfying all the clauses. By Lemma 11, we can assume that the solution is non-dominated, that is, each pair $\{v^i_x, v^i_{\overline{x}}\}$ contains at least one guard. By Lemma 12 and the budget allowing one extra guard by inverter gadgets, any non-dominated solution should actually be legal. Now, by Lemmas 10 and 12, the guarding set has further to be consistent. Therefore, on chunk T_0, one can read out a truth assignment, consistently kept along the chunks. This truth assignment satisfies all the clauses by Lemmas 13 and 14. Hence, the Planar 3-SAT-instance is satisfiable.

Proof of Theorem 4. This shows that Orthogonal Terrain Guarding and Dominating Set on the visibility graph of rectilinear terrains are NP-hard. Recall that the continuous variant of Orthogonal Terrain Guarding is equivalent to its discrete counterpart. The membership in NP of all those variants was established in the first paragraphs of this section. It remains to prove 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. But this is no longer required in Dominating Set. We observe that the general triangular pocket is only used in the downward clause gadget. We explain how we can make the downward clause gadget without the general triangular pocket. From the gadget depicted in Figure 11, we make the following modifications. The three literals of the clause are now at positions 2, 4, and 5 on chunk T_i. The *third* literal, that is, the one of the middle variable which does not satisfy the clause has its v-vertex slightly lowered in such a way that it does not see anything meaningful on chunk T_{i-1}. On chunk T_{i-1}, the right triangular pocket with bottom $d^{i-1}_{y,\overline{y}}$ is simply removed, and the triangular pocket with bottom w_C is replaced by a right triangular pocket which sees among the v-vertices $v^i_{\ell_1}, v^i_{\ell_2}, v^i_{\ell_3}$ and nothing else, for $C = \ell_1 \vee \ell_2 \vee \ell_3$ (see Figure 17).

The one drawback of this new construction is the removal of vertex $d^{i-1}_{y,\overline{y}}$ which forced to take one v-vertex between $v^i_y, v^i_{\overline{y}}$. We can now place no guard at those vertices, provided that we place two guards at v^{i+1}_y and $v^{i+1}_{\overline{y}}$. However, this can only help if there is also a downward clause gadget between chunks T_{i+1} and T_i. Therefore, we just have to observe the rule of not putting two downward clause gadgets in a row (for instance by separating them with some simple propagation).

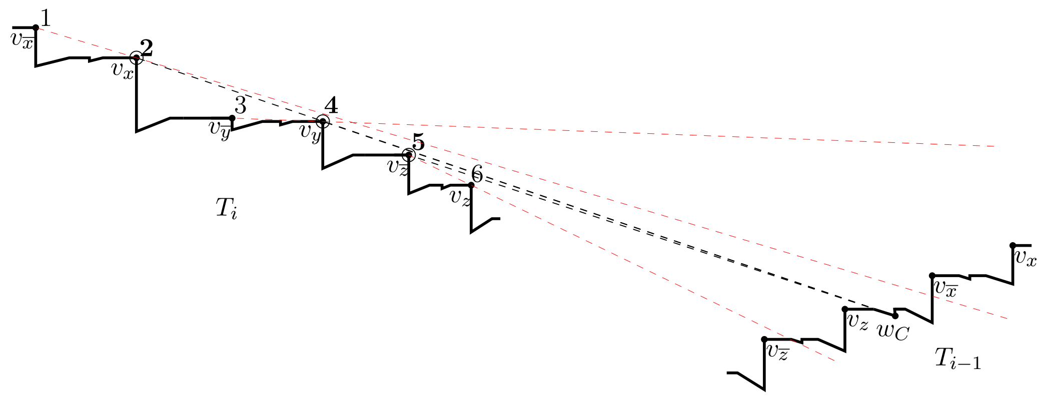

Figure 17: Variant of the downward clause gadget for $C = x \vee y \vee \neg z$ without general triangular pocket. We use the usual propagation for variables x and z. The variable y disappears from T_{i-1} and downward. The inverters have been used to place, on T_i, the literals of C at positions 2, 4, and 5. The vertex w_C is seen only by v^i_x, v^i_y, and $v^i_{\overline{z}}$ (circled); hence it is seen if and only if the chosen assignment satisfies C.

3 Improved ETH-Hardness for (Orthogonal) Terrain Guarding

This section is devoted to proving Theorem 5. We explain how to turn the quadratic reductions from Planar 3-SAT into cubic reductions from 3-SAT by taking a step back. This step back is the reduction from 3-SAT to Planar 3-SAT by Lichtenstein [25], or rather, the instances of Planar 3-SAT it produces. The idea of Lichtenstein in his 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 (see Figure 18).

Due to the sparsification of Impagliazzo et al. [17], 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 $\Theta(N^2)$ of intersections; which implies a replacement of the intersections by a quadratic number of constant-size crossover gadgets. More concretely, the original N variables (resp. $\Theta(N)$ clauses) are placed horizontally at the bottom of a $\Theta(N) \times \Theta(N)$ construction grid (resp. vertically at the left of that grid). Those original variables and clauses are joined in a rectilinear fashion. Crossover gadgets are placed on a superset of the edge intersections and subset of the grid (see Figure 19). There is a noose (blue closed curve on the figure) going through all the variables and defining the partition $(\mathcal{C}^+, \mathcal{C}^-)$. Let $\mathcal{C}^-$ be the part containing the original clauses and $\mathcal{C}^+$ be the other part.

We wish to reduce the number of chunks that we actually need to check all the clauses. 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. However, the only requirement for a clause to be checked is that it operates on consecutive variables. Therefore, one can check several clauses *in parallel* if they happen to be on disjoint and consecutive variables. *Checking a set of variable-disjoint clauses in parallel* means that we

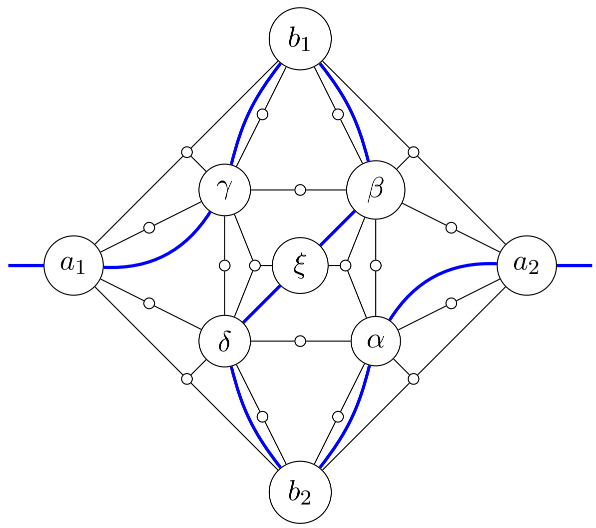

Figure 18: The crossover gadget of Lichtenstein for the crossing edges a_1a_2 and b_1b_2. The large labeled nodes represent variables, and the small unlabeled nodes represent clauses. The clauses ensure that the value of a_1 and a_2 (resp. b_1 and b_2) are the same. The thick blue curved line delimits on one side, the clauses of $\mathcal{C}^+$, and on the other side, the clauses of $\mathcal{C}^-$.

put the simple propagation/literal inverters/clause gadgets necessary to check a clause, on a constant number of chunks. In particular, between chunks, say, T_i and T_{i+1}, we may have multiple clause checker gadgets.

A first observation is that the $\Theta(N^2)$ 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. They are also consecutive within each gadget with the variable ordering $a_1, \gamma, b_1, \beta, \xi, \delta, b_2, \alpha, a_2$. We deal first with the remaining clauses of $\mathcal{C}^+$. At this point, there are still potentially $\Theta(N^2)$ equality constraints in $\mathcal{C}^+$. In Figure 19, the equality constraints are materialized by thick black edges going from one crossover to another. We say that an equality constraint is *vertical* if the corresponding edge contains a vertical section, and that it is *horizontal* otherwise. Hence a horizontal equality constraint is actually represented by a horizontal segment (without bend). The *column* of a vertical equality constraint is the column of its (unique) vertical section.

We first check in parallel all the vertical equality constraints of the first column (there are four in Figure 19, between rows 1 and 2, rows 3 and 4, rows 5 and 6, and rows 7 and 8). We can then check in parallel all the horizontal equality constraints whose segment ends to the left of the second column (there is just one in the figure, on row 2). Now, the vertical equality constraints of the second column can be checked in parallel (one in the figure, between rows 1 and 2). We then check at once all the horizontal equality constraints whose segment ends to the left of the third column (four, on rows 2, 4, 6, and 8 to 7), and so on. We therefore only need $\Theta(N)$ chunks for $\mathcal{C}^+$.

For $\mathcal{C}^-$, we do the same starting from the last column and going down column by

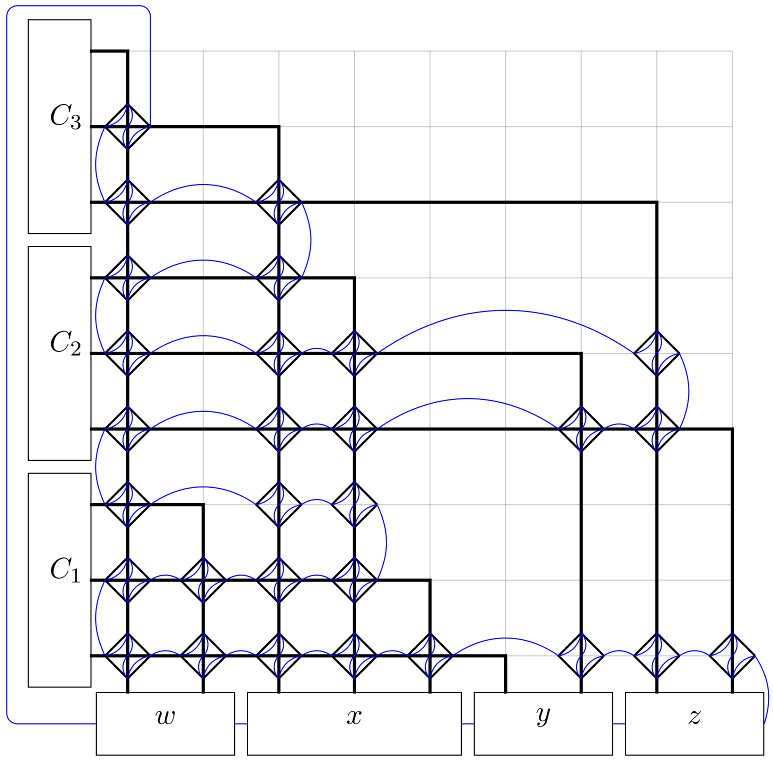

Figure 19: Reduction from 3-SAT to Planar 3-SAT, reproduction of Figure 4.3. in Tippenhauer's master thesis [28] which follows Lichtenstein's original paper. Notice that some crossover gadgets are used on places without edge intersection, in order to route the blue closed curve (indicating the separation $(\mathcal{C}^+, \mathcal{C}^-)$), with C^- containing the C_is and C^+ being on the *unbounded face*.

column. After $\Theta(N)$ chunks, we are left with the original variables and clauses which are only $O(N)$. Thus we finish with $O(N)$ additional chunks. A chunk contains $O(N^2)$ variable encodings, hence $O(N^2)$ vertices. So the total number of vertices of a terrain produced from a 3-SAT formula on N variables is $O(N^3)$. This implies that there is no algorithm running in time $2^{o(n^{1/3})}$ for (Orthogonal) Terrain Guarding on terrains with n vertices, unless the ETH fails.

4 Perspectives

We have proved that Orthogonal Terrain Guarding is NP-complete, as well as its variants. We showed how to get improved ETH-based lower bounds for Terrain Guarding and Orthogonal Terrain Guarding, by designing a cubic reduction from 3-SAT out of the quadratic reduction from Planar 3-SAT. This establishes that there is no $2^{o(n^{1/3})}$-time algorithm for those problems, unless the ETH fails.

Besides closing the gap between this lower bound and the existing $2^{O(\sqrt{n}\log n)}$-algorithm, 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?

A negative answer to the second question would come as a real surprise in light of the $k^{O(k)}n^{O(1)}$-time algorithm solving Dominating Set on the visibility graph of strictly orthogonal terrains.

References

[1] Mikkel Abrahamsen, Anna Adamaszek, and Tillmann Miltzow. The art gallery problem is ∃R-complete. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 65–73, 2018. URL: https://doi.org/10.1145/3188745.3188868, doi:10.1145/3188745.3188868.

[2] 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. URL: https://doi.org/10.4230/LIPIcs.SoCG.2017.11, doi:10.4230/LIPIcs.SoCG.2017.11.

[3] 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. URL: https://doi.org/10.1137/S0097539704446384, doi:10.1137/S0097539704446384.

[4] Pritam Bhattacharya, Subir Kumar Ghosh, and Sudebkumar Prasant Pal. Constant approximation algorithms for guarding simple polygons using vertex guards. *CoRR*, abs/1712.05492, 2017. URL: http://arxiv.org/abs/1712.05492, arXiv:1712.05492.

[5] 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. URL: https://doi.org/10.4230/LIPIcs.ESA.2016.19, doi:10.4230/LIPIcs.ESA.2016.19.

[6] 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. URL: https://doi.org/10.4230/LIPIcs.SoCG.2017.20, doi:10.4230/LIPIcs.SoCG.2017.20.

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

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

[9] 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. URL: https://doi.org/10.1007/978-3-540-73951-7_15, doi:10.1007/978-3-540-73951-7_15.

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

[11] 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. URL: https://doi.org/10.1007/3-540-49381-6_45, doi:10.1007/3-540-49381-6_45.

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

[13] 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. URL: https://doi.org/10.4230/LIPIcs.SoCG.2017.40, doi:10.4230/LIPIcs.SoCG.2017.40.

[14] 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. URL: https://doi.org/10.1007/s00453-009-9358-4, doi:10.1007/s00453-009-9358-4.

[15] Stephan Friedrichs, Michael Hemmer, James King, and Christiane Schmidt. The continuous 1.5D terrain guarding problem: Discretization, optimal solutions, and PTAS. *JoCG*, 7(1):256–284, 2016. URL: http://jocg.org/index.php/jocg/article/view/242.

[16] Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367 – 375, 2001. URL: http://www.sciencedirect.com/science/article/pii/S0022000000917276, doi:http://dx.doi.org/10.1006/jcss.2000.1727.

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

[18] James King. Errata on a 4-approximation for guarding 1.5-dimensional terrains. URL: http://www.cs.mcgill.ca/~jking/papers/4apx_latin.pdf.

[19] 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. URL: https://doi.org/10.1007/11682462_58, doi:10.1007/11682462_58.

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

[21] James King and Erik Krohn. Terrain guarding is NP-hard. *SIAM J. Comput.*, 40(5):1316–1339, 2011. URL: https://doi.org/10.1137/100791506, doi:10.1137/100791506.

[22] David G. Kirkpatrick. An $O(lg\ lg\ OPT)$-approximation algorithm for multi-guarding galleries. *Discrete & Computational Geometry*, 53(2):327–343, 2015. URL: https://doi.org/10.1007/s00454-014-9656-8, doi:10.1007/s00454-014-9656-8.

[23] 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.

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

[25] David Lichtenstein. Planar formulae and their uses. *SIAM J. Comput.*, 11(2):329–343, 1982. URL: https://doi.org/10.1137/0211025, doi:10.1137/0211025.

[26] 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.

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

[28] Simon Tippenhauer. On planar 3-SAT and its variants. Master's thesis, Freie Universität Berlin, 2016.