Computing Realistic Terrains from Imprecise Elevations

Anna Lubiw
School of Computer Science, University of Waterloo, Canada

Graeme Stroud
School of Computer Science, University of Waterloo, Canada

Abstract

In the imprecise 2.5D terrain model, each vertex of a triangulated terrain has precise x- and y-coordinates, but the elevation (z-coordinate) is an imprecise value only known to lie within some interval. The goal is to choose elevation values from the intervals so that the resulting precise terrain is "realistic" as captured by some objective function.

We consider four objectives: (1) minimizing local extrema; (2) optimizing coplanar features; (3) minimizing surface area; (4) minimizing maximum steepness.

We also consider the problems down a dimension in 1.5D, where a terrain is a poly-line with precise x-coordinates and imprecise y-coordinate elevations. In 1.5D we reduce three of the problems to a shortest path problem and show that Problem 2 (optimizing collinear features) can be 2-approximated via a minimum link path.

In 2.5D, Problem 1 (minimizing local extrema) was proved NP-hard by Gray et al. [Computational Geometry, 2012]. We give a polynomial time algorithm in the special case when the triangulation is the triangulation of a polygon. We prove that Problem 2 (optimizing coplanar features) is strongly NP-complete, but give a constant-factor approximation when the triangles form a path and lie in a strip. We show that Problems 3 and 4 (minimizing surface area and minimizing maximum steepness) can be solved efficiently via Second Order Cone Programming.

Keywords and phrases terrain, elevation, triangulation, imprecise points, smoothing, extrema, surface area, steepness, minimum link path

Digital Object Identifier 10.57717/cgt.v2i2.29

Related Version Master's thesis of Graeme Stroud [22]

Funding *Anna Lubiw*: Supported by Natural Sciences and Engineering Research Council of Canada

1 Introduction

A natural problem that arises in Geographic Information Systems is to compute a triangulated terrain in 3D space that is "nice" or "realistic". There is no single objective function to capture "niceness". In the study of erosion and hydrology, it is generally accepted that pits in a triangulated terrain are artifacts of imprecision, due to the unrealistic occurrence of water accumulation in flow simulations [12]. This motivates minimizing the number of extrema in the terrain. Since actual terrains tend to be smoothed by erosion, other natural objectives are to minimize the surface area, or to make the terrain as flat as possible.

A triangulated terrain is often computed from real elevation data. It is usually assumed that the data is accurate, however data acquisition can be complex and potentially prone to errors. It may be appropriate to model the input data as coming from a possible range of values to account for this uncertainty. Dealing with uncertainty or imprecision in the input data is a broad, well-studied area in computational geometry. Each input point may be represented by an uncertainty region, and the issue then is to find the best (or worst) placement of points, one in each region, for the problem at hand. For imprecise points in the

	extrema	coplanarity/collinearity	area/length	#4: steepness
1.5D	$\boldsymbol{O(n)}$ **[§2]**	**2-approx [§2]**	$\boldsymbol{O(n)}$ **[§2]**	$\boldsymbol{O(n)}$ **[§2]**
2.5D for triangulation of	$\boldsymbol{O(n^4)}$ **[§3] polygon**	**5-approx [§4.1] strip**		
2.5D general	NP-hard [12]	**NP-hard [§4.2]**	**SOCP [§5]**	**SOCP [§6]**

Table 1 Summary of results, with new results in bold.

plane, there is work on minimizing/maximizing the width, the area of the bounding box, or the diameter of the points [15, 19].

For the case of terrains, Gray and Evans [10], and Gray [11] formulated the ***imprecise 2.5D terrain model***. In this model, the x, y-coordinates of points are given as input, along with a triangulation defined on the points when projected to the xy plane, but the z-coordinate (elevation) of each point is only specified "imprecisely" via an interval of possible values. We obtain a ***precise 2.5D terrain*** (or a "realization" of the imprecise terrain) by choosing a precise elevation from each uncertainty interval and connecting the points together according to the input triangulation. A more detailed definition is given in Section 3. We allow triangulations where the outer face is not the convex hull of the points, for example, a triangulation of a simple polygon. Although the x, y data could also be considered to be imprecise (so that the vertex is somewhere within a 3D box), one justification for the current simpler model is that inaccuracy in the z-coordinate can compensate for inaccuracy in the x, y-coordinates.

Various "niceness" criteria for choosing a precise terrain have been considered in the past such as minimizing the number of local extrema [12], or minimizing the length of the shortest path along the terrain from one point to another [10, 16].

When these problems are NP-hard or have unknown computational complexity for 2.5D terrains, researchers (e.g., Gray et al. [13]) have considered ***imprecise 1.5D terrains***. Here, the x-coordinates are precise, and the elevations are the y-coordinates, each of which is given imprecisely via an interval.

We explore four objective functions that capture different "niceness" criteria for a terrain. To the best of our knowledge, only the first one (minimizing the number of extrema) has been considered before. See Table 1 for a summary of the results for these four problems.

Problem 1: Extrema. The objective is to minimize the number of local extrema. A local extremum is a local maximum or minimum compared to its neighbours in the triangulation. In a terrain these correspond to peaks or pits. To deal with equal elevations, define a ***plateau*** to be a maximal set of points of equal elevation that are connected by edges. A ***local minimum [maximum]*** is a plateau such that all neighbouring points have higher [lower] elevations. The problem of minimizing the number of local extrema was proved NP-hard in 2.5D by Gray et al. [12]. We give a polynomial time algorithm for the special case when the triangulation is the triangulation of a polygon, and solve the 1.5D version in linear time via a shortest path.

Problem 2: Coplanarity. The objective is to optimize coplanar features. To make a smooth terrain, we would like triangles to be coplanar with adjacent triangles if possible. This can be formalized as minimizing the number of ***patches***, where a patch is a maximal set of coplanar triangles that are connected edge-to-edge. An alternative is to minimize the number of ***bends***, where a bend is an edge whose two incident triangles are not coplanar. These objectives have different solutions in general, though they have the same solutions in 1.5D, where the goal is to optimize collinear features: a patch is a maximal set of connected collinear edges (a "link") and a bend is a point whose two incident edges are not collinear. We show that both the patch and the bend versions are NP-complete in 2.5D. We give an easy 2-approximation in 1.5D and extend this to a 5-approximation for 2.5D in the special case where the triangles form a path in a strip (i.e., there are only two y-values). This triangulation is a special case of a polygon triangulation.

Problem 3: Area. The objective is to minimize the surface area, or in 1.5D, the length. These are very natural objective functions. In 1.5D this becomes a shortest path problem. We formulate the 2.5D version as a Second Order Cone Program (SOCP). Second Order Cone Programming is a type of convex optimization problem that can be solved quite efficiently [20] (though not in polynomial time).

Problem 4: Steepness. The objective is to minimize the maximum steepness. The ***steepness*** of a segment in 2D is the absolute value of its slope, and ***steepness*** of a triangle in 3D is the norm of its gradient. Minimizing steepness gives a terrain that is as flat as possible, another reasonable objective.

We formulate the 2.5D version as a Second Order Cone Program, and show that the 1.5D version is solved via a shortest path—even for a lexicographic version where we minimize the maximum steepness, and subject to that, minimize the second maximum, etc.

Our paper is organized as follows: Section 2 deals with 1.5D terrains and the later sections deal with 2.5D terrains, specifically, Section 3 on Problem 1 (Extrema), Section 4 on Problem 2 (Coplanarity), Section 5 on Problem 3 (Area), and Section 6 on Problem 4 (Steepness). In the Conclusions section, we mention two very natural and practical special cases of the imprecise 2.5D terrain problem that are left for future work: (1) when the triangulation is the Delaunay triangulation; and (2) when all the imprecise elevation intervals have the same length.

1.1 Background

Gray [11] was the first to consider the imprecise terrain model. (See also Gray and Evans [10].) They considered the problem of finding the shortest path from one point to another over all precise realizations of the terrain. Recently, various problems have been explored for imprecise 1.5D and 2.5D terrains. The problem of minimizing the number of extrema was first explored by Gray et al. [12]. In the general 2.5D case, they show that minimizing the number of extrema is NP-hard, and there is no $O(\log \log n)$ approximation algorithm unless P = NP. Driemel et al. [9] considered the problem of determining whether water can flow between two points of an imprecise 2.5D terrain. Here, the assumption is that water flows down the path of steepest descent. Gray et al. [13] considered a few objectives that result in "smooth" 1.5D terrains, such as minimizing [maximizing] the total turning angle, and minimizing [maximizing] the largest [smallest] turning angle.

The problem of minimizing the number of links/bends for an imprecise 1.5D terrain is related to curve simplification. Imai and Iri [14] provide an algorithm that, given as input an x-monotone polygonal line and $\varepsilon > 0$, computes another polygonal line that is within vertical distance ε of the input polyline, while containing as few (bend) points as possible. Although there is no input polyline for the links problem, the output will also be a polyline, and it will consist of as few points as possible. However, in the links/bends problem, the points are limited to the n possible x-coordinates given as input, but Imai and Iri's problem does not require the bend points to be at any specific locations. Other polyline distance metrics for simplification have been considered, see [24] for a recent result for the Fréchet and Hausdorff metrics. For a recent result on curve simplification under uncertainty, see [3]. For a nice survey about curve simplification, see Section 4.1 of [1].

Imprecise points more generally. The imprecise points model has been explored more generally than for terrains. Given a class of objects in the plane as input, one can ask for a set of points (one per input object) that optimizes some objective function. For example, for classes such as line segments or squares, Löffler and Van Kreveld [18] give algorithms or hardness results for the objectives of minimizing [maximizing] the area or perimeter of the convex hull of the points. Other optimization objectives for imprecise points have been explored, including minimizing [maximizing] the width, the area of the bounding box, and the diameter of the points [15, 19]. Another optimization objective is to compute the minimum Euclidean weight of a spanning tree [8], over all possible placements of points and over all possible trees of the points. A non-optimization objective was considered by Löffler and Snoeyink [17], which asks to compute a data structure to efficiently compute the Delaunay triangulation for any possible precise realization of the imprecise points. Even more general is the study of how imprecision affects the accuracy of geometric computations, which is called "epsilon geometry" [21].

2 1.5D Terrains

The input for an ***imprecise 1.5D terrain problem*** consists of n x-coordinates, $x_1 < x_2 < \cdots < x_n$ given in sorted order, and n vertical segments $\ell_1, \ell_2, \ldots, \ell_n$ where ℓ_i has x-coordinate x_i. The problem is to choose a point p_i on ℓ_i so that the ***1.5D precise terrain*** $p_1, p_2, \ldots, p_n$ has the desired property. A precise terrain can be thought of as a "realization" of an imprecise terrain.

In this section we show that optimal 1.5D terrains for Problem 1 (Extrema), Problem 3 (Length), and Problem 4 (Steepness) can be computed in linear time by finding a shortest path in an appropriate polygon. For Problem 2 (Collinearity) we show that a minimum link path in the polygon provides a linear time 2-approximation.

Let P be the simple polygon whose vertices are the top and bottom endpoints of the segments ℓ_i, with a chain joining consecutive top endpoints, a chain joining consecutive bottom endpoints, plus the two edges ℓ_1 and ℓ_n. See Figure 1. Consider a shortest path π from ℓ_1 to ℓ_n in P, i.e., a shortest path in P from some point on ℓ_1 to some point on ℓ_n. Path π is unique unless it is a straight horizontal path that can shift up/down. Note that a shortest path in a polygon only bends at the polygon vertices. The vertices of polygon P are endpoints of segments, and therefore the path π provides a precise 1.5D terrain, which we will show is optimal for Problems 1, 3, and 4. (As discussed below, for Problem 2 (Collinearity) we need a minimum link path instead of a shortest path, and the fact that a minimum link path may bend at non-vertex points is why we can only achieve a 2-approximation.)

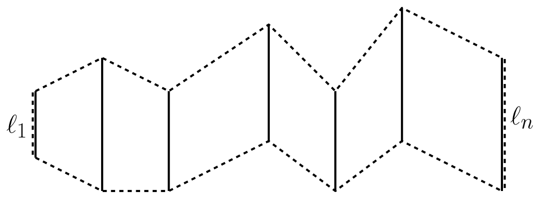


Figure 1 The input segments for the imprecise 1.5D terrain problem (solid) and the polygon P (dashed).

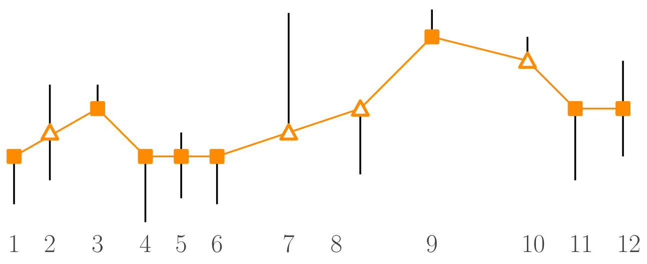


Figure 2 A shortest path terrain with five extreme plateaus (marked by squares).

▶ **Theorem 1.** *A shortest path from ℓ_1 to ℓ_n in polygon P can be found in linear time and provides an optimal 1.5D terrain for Problem 1 (Extrema), Problem 3 (Length), and Problem 4 (Steepness).*

Proof. The shortest path from one segment to another in a simple polygon can be found in linear time [5]. This algorithm needs a triangulation of the polygon. Thankfully, we do not need Chazelle's impractical linear time algorithm [4], since P is composed of trapezoids each of which can be cut into two triangles.

As discussed above, the shortest path provides a 1.5D terrain. We must show that the terrain is optimal. This is obvious for Problem 3 (Length). We next consider Problems 1 and 4.

Problem 1 (Extrema). Suppose the shortest path has k local extrema. We must prove that this is optimal, i.e., that any 1.5D terrain has at least k extrema. The plateaus of the leftmost point and rightmost point are extreme by definition. If $k = 1$ then there is a single plateau (the shortest path is horizontal) and this is clearly optimal. So suppose $k > 1$. Note that the extrema alternate between minima and maxima as we traverse the path. Let p_{i_j} be the rightmost point of the jth extreme plateau, lying on segment ℓ_{i_j} for $j = 1, \ldots, k-1$, and—since we want points where the path bends—let p_{i_k} be the leftmost point of the rightmost extreme plateau.

We will show that any 1.5D terrain must include at least k extrema, the leftmost and rightmost extrema plus at least $k-2$ others, one between segments $\ell_{i_{j-1}}$ and $\ell_{i_{j+1}}$ for each j, $2 \leq j \leq n-1$.

First, note that the points p_{i_j} zig-zag, i.e., if p_{i_j} is a minimum [maximum] then p_{i_j} is lower [higher] than $p_{i_{j-1}}$ and $p_{i_{j+1}}$. We see this in Figure 2, for instance, where the point on segment 6 is below the points on segments 3 and 9. Also, if p_{i_j} is part of a minimum [maximum] plateau, then the angle above [below] the path at p_{i_j} is strictly convex, so (because the path is shortest) p_{i_j} must be at the upper [lower] endpoint of its segment. Therefore, any point on segment ℓ_{i_j} is necessarily below [above] any point on segments $\ell_{i_{j-1}}$ and $\ell_{i_{j+1}}$ so there must always be a local minimum [maximum] between segments $\ell_{i_{j-1}}$ and $\ell_{i_{j+1}}$. Finally, note that since these extrema must alternate between minima and maxima, the extremum between $\ell_{i_{j-1}}$ and $\ell_{i_{j+1}}$ is distinct from the extremum between ℓ_{i_j} and $\ell_{i_{j+2}}$. Thus, any 1.5D terrain has at least k extrema, and the shortest path is an optimal solution for Problem 1.

Problem 4 (Steepness). We prove that the shortest path provides something stronger: it minimizes the maximum steepness, and, subject to that, minimizes the second maximum steepness, and so on. More formally, define the ***steepness vector*** of a 1.5D terrain to be the vector of $n-1$ steepness values obtained from the $n-1$ edges of the terrain, sorted in *decreasing* order (so that the maximum steepness value is first). Note that there might be duplicate values. We say that one steepness vector s is ***lexicographically less than***

another steepness vector s' if $s \neq s'$ and for the first index i where s differs from s', we have $s_i < s'_i$.

▶ **Proposition 2.** *A shortest path from ℓ_1 to ℓ_n in polygon P lexicographically minimizes the steepness vector.*

In order to prove Proposition 2, we first prove that a path that lexicographically minimizes the steepness vector is locally shortest, i.e., cannot be shortened by any small perturbation:

▷ Claim 3. If a path π in P from ℓ_1 to ℓ_n lexicographically minimizes the steepness vector, then it is locally shortest.

Proof. Suppose π bends at point p_i, and suppose the convex angle is above p_i (the other case is symmetric). If p_i is not at the top of its segment then moving p_i upward would change the steepness of the two incident edges, decreasing the larger steepness and (possibly) increasing the smaller steepness, and thus lexicographically decreasing the steepness vector. Therefore, π must be locally shortest at every bend. Furthermore, π must also be locally shortest at its endpoints, otherwise the steepness of the incident edge could be decreased. ◀

Returning to the proof of Proposition 2, it is well-known that between any two points in a simple polygon there is a unique locally shortest path, which is then the unique shortest path. The same is true if the points are replaced by segments, with one exception when the segments are parallel and the shortest path is a line segment orthogonal to the segments, then this path may be shifted parallel to itself. In our situation, segments ℓ_1 and ℓ_n are both vertical. If the shortest path from ℓ_1 to ℓ_n in P is unique, then it is the unique locally shortest path, and therefore the unique path that lexicographically minimizes the steepness vector by Claim 3. Otherwise, any shortest path from ℓ_1 to ℓ_n in P is a horizontal segment so its steepness vector is 0, which is lexicographically minimum.

This completes the proof of Theorem 1. ◀

We now turn to Problem 2 (Collinearity) where the goal is to minimize the number of links/bends. First note that the number of links is one more than the number of bends, so the two versions are equivalent (unlike in 2.5D). We make use of a minimum link path in polygon P from ℓ_1 to ℓ_n, which can be found in linear time using Suri's minimum link path algorithm [23]. (Suri's algorithm finds a minimum link path from a source point to a target point in a simple polygon, but, internally, it finds a minimum link path from a segment (a visibility window) to the target point, so it can easily be extended to deal with source and target segments.) This path may have bends that are not at the input line segments, but each such bend b can be replaced by two bends at the line segments just before and after b.

▶ **Theorem 4.** *A minimum link path π from ℓ_1 to ℓ_n in P can be found in linear time and the points where π intersects the segments provide a 1.5D terrain with at most twice the minimum number of bends.*

Proof. The number of bends in π is clearly a lower bound, and each bend in π is replaced by at most two bends in the terrain. ◀

3 Local extrema

We now turn to 2.5D terrains. The input is a set of 2D points, $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, a triangulation of these points, and a vertical range (b_i, t_i) for each $i = 1, \ldots, n$. The problem is to choose z_i with $b_i \leq z_i \leq t_i$ so that the resulting ***precise 2.5D terrain*** with vertices

(x_i, y_i, z_i) has the desired property. Note that we allow input triangulations that do not include all the convex hull edges of the input 2D points (to model triangulating a general shape). In this section we consider Problem 1, minimizing the number of local extrema.

Gray et al. [12] showed that this problem is NP-hard for 2.5D terrains. Therefore, we will examine a special case where we have a triangulation of a polygon, i.e., all points are on the boundary of the triangulation.

▶ **Theorem 5.** *There is an $O(n^4)$ time dynamic programming algorithm to minimize the number of extrema for imprecise 2.5D terrains when the input triangulation is a triangulation of a polygon.*

The following claim shows that we can restrict to a discrete set of elevation values.

▷ Claim 6. Let $E = \{b_1, t_1, \ldots, b_n, t_n\}$ denote the set z-values of the bottom and top endpoints of the input intervals. Then there exists an optimal solution z^* so that $z_i^* \in E$ for all $i = 1, \ldots, n$.

Proof. Consider any optimal solution z^*. Suppose there are $k > 0$ elevation values not in E (multiple points can share the same elevation value). Let v be the *largest* elevation value that is not in the set E and let V denote the set of points with elevation v. We will show that we can modify the solution to shift all the points of V upward to an elevation value from E (thus reducing k) while keeping the solution optimal and not changing any other elevation values. Therefore, we can apply induction on k in order to obtain an optimal solution with all elevations in E.

Let w denote the *smallest* value from E that is greater than v. This exists since if p_i is a point whose elevation is the chosen value v then $v < t_i$ and $t_i \in E$. Note that there are no points with elevations between v and w, by the definitions of v and w. We will move the points of V up to have elevation w. Since w is the smallest value from E that is greater than v, this new solution is feasible. Also, we have decreased k. Let W denote the set of points with elevation w (not including the moved points of V). First suppose that W is empty. Then the total ordering of the points (by z-value) has not changed, so the extrema have not changed, and the new solution is optimal.

Next, suppose that W is not empty. To show that the new solution is optimal, we will prove that any extremum in the new terrain resulted from at least one extremum from the original precise terrain (the terrain corresponding to z^*). The extrema that are not at elevation w in the new terrain are clearly extrema in the original terrain as well, so we will focus on the extrema at elevation w. Let $M \subseteq V \cup W$ be such an extremum in the new terrain. We will focus on the case that M is a local maximum, since the case of a local minimum can be argued similarly.

In the new terrain, every point that is adjacent to M has an elevation value less than v. If M only consists of points of V, then M is clearly a local maximum present in the original terrain as well. Otherwise, some points from $W \cap M$ form at least one local maximum in the original terrain.

Therefore, for every extremum M in the new terrain, there exists a subset of M representing an extremum in the original terrain. Therefore, the new terrain has at most as many extrema as the original one, and must be optimal. ◀

We will now describe the algorithm. Label the vertices around the polygon $p_1, \ldots, p_n$ in clockwise order. For each edge $p_i p_j$ of the triangulation with $i < j$, we define a subproblem $S_{i,j}(z_i, \alpha_i, \beta_i, z_j, \alpha_j, \beta_j)$. This records the minimum number of *internal* extreme plateaus for the subpolygon $p_i, \ldots, p_j$ where $z_i \in E$ is the elevation for p_i, $\alpha_i \in \{T, F\}$ records whether

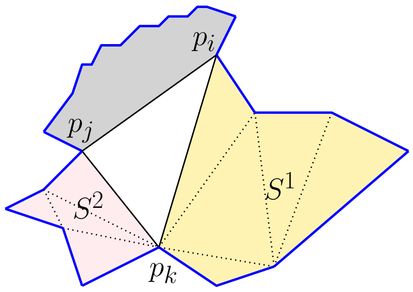


Figure 3 Splitting subproblem $S_{i,j}$ into $S^1 := S_{i,k}$ (yellow) and $S^2 := S_{k,j}$ (pink).

there are above (higher) elevations connected to p_i's plateau, $\beta_i \in \{T, F\}$ records whether there are below (lower) elevations connected to p_i's plateau, and similar for j. Here "internal" means that we do not count the plateau(s) of p_i and p_j. It is easy to add those plateaus into the count, since p_i's plateau is a local extremum in $S_{i,j}$ iff $\neg\alpha_i \vee \neg\beta_i$ (i.e., there are no higher elevations connected to its plateau or there are no lower elevations connected to its plateau) and similarly for p_j. Furthermore, they are in the same plateau iff $z_i = z_j$.

The algorithm computes all $S_{i,j}$ entries using dynamic programming. Initialize by setting $S_{i,j}(z_i, \alpha_i, \beta_i, z_j, \alpha_j, \beta_j)$ to ∞ when the parameters are *incompatible*, meaning that a z value is outside its interval, or the α, β values contradict the z values, e.g., $\alpha_i = F$ but $z_j > z_i$, etc.

We solve for $S_{i,j}(z_i, \alpha_i, \beta_i, z_j, \alpha_j, \beta_j)$ for compatible parameter values, starting with smaller values of $j - i$ before larger values. When $j = i + 1$, there are only two points (i.e., the subpolygon is an edge), and the number of internal extrema is zero.

For $j > i + 1$, there is a (unique) triangle p_i, p_k, p_j with $i < k < j$. Our goal is to combine solutions to the two subproblems $S_{i,k}$ and $S_{k,j}$ for various z, α, β values. See Figure 3. $S_{i,k}$ inherits z_i. $S_{k,j}$ inherits z_j. For z_k, we try all values in E (the same value in both subproblems). The above/below values are not simply inherited, since, for example, a T value for α_i in $S_{i,j}$ can come from a F value in $S_{i,k}$ if z_j provides the above elevation.

To simplify notation, let S^1 be $S_{i,k}$ and S^2 be $S_{k,j}$. Let α_i^1 be the α-value(s) of p_i in S^1, let α_k^1 be the α-value(s) of p_k in S^1, and similarly for the β values and for S^2. We have $\alpha_i \equiv \alpha_i^1 \vee (z_j > z_i)$. This tells us which values of α_i^1 to try. Similarly for β_i^1 and α_j^2, β_j^2.

We next specify which above/below values to try for p_k in the two subproblems. We will consider all possibilities for the final above/below values α_k, β_k of p_k in $S_{i,j}$. Namely, $(T, T), (T, F), (F, T), (F, F)$. We have $\alpha_k \equiv \alpha_k^1 \vee \alpha_k^2$, i.e., there are elevations above p_k's plateau in $S_{i,j}$ iff there are elevations above p_k's elevation in S^1 or in S^2. This tells us which values of α_k^1 and α_k^2 to try for a given choice of α_k. Similarly for β_k.

Finally, we set $S_{i,j}$ to be the minimum value, among all these choices, obtained as $S^1 + S^2 + \delta$ where $\delta \in \{0, 1\}$ is 1 iff p_k's plateau is an extremum distinct from the plateaus of p_i and p_j, i.e., iff $(\neg\alpha_k \vee \neg\beta_k) \wedge (z_k \neq z_i) \wedge (z_k \neq z_j)$.

The final minimum number of local extrema is obtained by taking the best of all the $S_{1,n}$ values, after adding 0, 1, or 2 extrema for p_1 and p_n as appropriate.

The algorithm is correct because we have considered all possibilities for the two subproblems.

Runtime. There are $O(n)$ edges $p_i p_j$ in the triangulation, and for each, we consider $O(n^2)$ elevations and above/below values, for a total of $O(n^3)$ subproblems. To solve a subproblem for $S_{i,j}$ we try $O(n)$ values for z_k and a constant number of combinations of above/below values. Thus the runtime of the algorithm is $O(n^4)$.

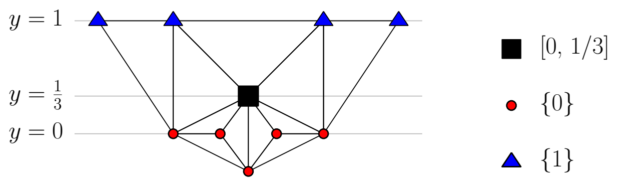


Figure 4 Proving that minimizing the number of bends is not always equivalent to minimizing the number of patches. The central point (drawn with a black square) is the only one with a non-trivial z-interval, viz. $[0, 1/3]$.

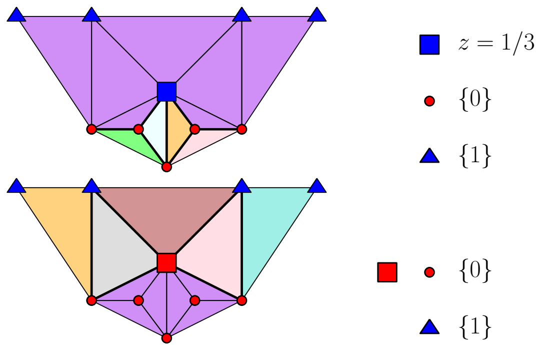


Figure 5 Solutions, with colours indicating the patches, and heavy lines indicating the bends: (top) if we place the center point (drawn with a square) at the top end of its elevation interval we get 5 patches (optimal) and 7 bends; (bottom) if we place the center point at the bottom end of its elevation interval, we get 6 patches and 6 bends (optimal).

4 Coplanar features

In this section, we explore Problem 2, minimizing the number of patches/bends in a 2.5D imprecise terrain. First, we give a 5-approximation algorithm for a triangulation in a strip (as shown in Figure 6). Then, we show that the general case is NP-complete for both objectives.

In general, these two objectives are not equivalent. Figure 4 shows an input in which all points have fixed elevations except for the central point p which has elevation range $[0, 1/3]$. As shown in Figure 5: if p is placed at the upper elevation, 1/3, then the resulting precise terrain has 5 patches and 7 bends; if p is placed at the lower elevation, 0, then the resulting precise terrain has 6 patches and 6 bends; and if p is placed at any other elevation in $(0, 1/3)$, then the resulting precise terrain has no co-planar triangles, thus 11 patches and 13 bends.

For a triangulation of a polygon, minimizing the number of patches is equivalent to minimizing the number of bends. More precisely, in any solution, the number of patches is one more than the number of bends: If a solution has B bends, then the bends, which are chords of the polygon, partition the polygon into $B + 1$ regions, and these regions are exactly the patches.

4.1 An algorithm for a strip triangulation

A strip triangulation is a special case of a polygon triangulation in which the polygon vertices lie on two horizontal lines, see Figure 6.

▶ **Theorem 7.** *There is a poly-time 5-approximation algorithm for the problem of minimizing the number of bends/patches when the input is restricted to a strip.*

The run-time of the algorithm depends on the run-time for linear programming, which we use as a subroutine. See the end of the section for further details.

Since a strip triangulation is a special case of a triangulation of a polygon, minimizing the number of bends is equivalent to minimizing the number of patches. We describe the algorithm in terms of bends.

Let the triangles along the strip be $T_1, \ldots, T_N$, where $N = n - 2$. We first greedily find the maximum index j such that triangles $T_1, \ldots, T_j$ can be coplanar. To test a given j, use a linear program whose variables are the z values of the imprecise points and the coefficients A, B, C of the plane $z = Ax + By + C$ that the triangles should lie in. Find the maximum j using binary search. The linear program gives precise elevations that make $T_1, \ldots, T_j$ coplanar.

Note that any precise terrain for $T_1, \ldots, T_{j+1}$ must have at least one bend. Let $k > j$ be the minimum index such that triangle T_k shares no vertices with T_j. The algorithm will recurse on triangles $T_k, \ldots, T_N$. It remains to fix the elevations of the vertices that lie between T_j and T_k. Observe that the situation is as shown in Figure 6: the last edge of T_j is ab; the first edge disjoint from ab is pq which is the first edge of T_k; and all intermediate triangles $T_{j+1}, \ldots, T_{k-1}$ include vertex a (without loss of generality, assume a and p lie on the top side of the strip). Note that the elevations of a and b have been fixed by the first greedy step, and the elevations of p and q will be fixed by the recursive call. By induction, it suffices to choose elevations for the remaining vertices, the ones that lie strictly between b and q along the bottom of the strip, so that the resulting precise terrain on $T_1, \ldots, T_{k-1}$ is a 5-approximation of the optimum.

Observe that triangles $T_{j+2}, \ldots, T_{k-3}$ form a ***fan*** F between apex a (with fixed elevation) and base edges (with imprecise elevations) on the bottom of the strip. Two adjacent triangles in this fan are coplanar iff their base edges are collinear. This reduces the problem to a 1.5D imprecise terrain problem in the xz-plane through the bottom of the strip. We use the linear time algorithm from Theorem 4 to find a 2-approximation for the minimum number of bends.

For an input I with n vertices, the algorithm runs in time $O(n \log n)$ times the run-time for linear programming with $O(n)$ variables, $O(n)$ constraints and coefficients that are the x, y-coordinates given in I.

Proof of correctness. Let OPT be an optimum solution and let B^* be the number of bends in OPT on edges up to and including pq. Let B be the number of bends on these edges produced by the above algorithm. Let s^* be the minimum number of bends for internal edges of the fan F. Then we have: $B^* \geq 1 + s^*$ since there is at least 1 bend before T_{j+1}, and s^* bends within F; and $B \leq 5 + 2s^*$, since there are five bends outside F (on the labelled edges in Figure 6) and at most $2s^*$ inside F by Theorem 4. Thus $5B^* \geq 5 + 5s^* \geq B$. Applying induction proves the approximation ratio is correct for the whole input.

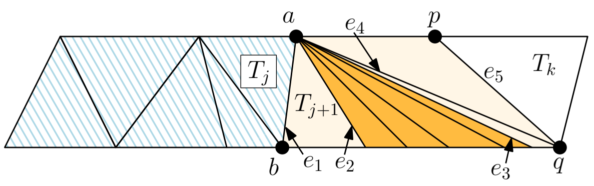


Figure 6 The first iteration of the algorithm. Fan F is colored dark orange and does not include the three pale orange triangles. The planar patch $T_1, \ldots, T_j$ is indicated with hatched blue lines.

4.2 NP-hardness for the general setting

We show that the objective of minimizing the number of patches is NP-complete for the case of a general triangulation without holes, using a reduction from Monotone Rectilinear Planar 3-SAT. The same reduction shows that minimizing the number of bends is NP-complete.

▶ **Theorem 8.** *The following problem is strongly NP-complete: Given an input to the imprecise 2.5D terrain problem in which the triangulation boundary is the convex hull of the points, and given a number k, is there a choice of elevations such that the resulting precise 2.5D terrain has at most k patches [or at most k bends].*

We first prove that the problem lies in the class NP by showing that a non-deterministic guess for the patches/bends can be verified in polynomial time using linear programming. For minimizing the number of bends, the procedure is as follows: Non-deterministically select k of the edges to be the ones with bends. The other edges will be called the ***non-bent edges***. For each non-bent edge e, define linear programming variables a_e, b_e, c_e, with the intended meaning that the triangles on both sides of e lie on the same plane $a_e x + b_e y + z + c_e = 0$. Also define linear programming variables z_i for all imprecise points i. A solution in the feasible region for the following linear program will be a 2.5D terrain with at most k bends.

- $b_i \leq z_i \leq t_i$ for all i
- for all non-bent edges e, let i, j, k, l be the indices of the imprecise points that form the two triangles sharing this edge. Then we want the points to be coplanar, so:
  - $a_e x_i + b_e y_i + z_i + c_e = 0$
  - $a_e x_j + b_e y_j + z_j + c_e = 0$
  - $a_e x_k + b_e y_k + z_k + c_e = 0$
  - $a_e x_l + b_e y_l + z_l + c_e = 0$

Since linear programming can be solved in polynomial time, this is a non-deterministic polynomial time algorithm. For the patches problem, we can similarly non-deterministically choose the disjoint patches of triangles, and then solve for the z-values using linear programming.

We next show that the problem is complete for the class NP.

Reduction details. The reduction will be from the NP-complete problem Monotone Rectilinear Planar 3-SAT [6]. In this variant of 3-SAT, each clause has either three positive literals or three negative literals, and the input includes a planar representation where each variable v is represented by a thin vertical rectangle along the line $x = 0$, each positive [negative] clause is represented by a thin vertical rectangle at a positive [negative, resp.] x-coordinate, and there are horizontal line segments joining each clause rectangle to the rectangles of the variables in the clause. We modify the representation by shrinking each clause rectangle to a square that is connected to its three literals via three "wires" created from the horizontal segments—the middle one remains horizontal, the bottom one bends to enter the clause rectangle from the bottom, and the top one bends twice to enter the clause rectangle from the far side. See Figure 7. For n variables and m clauses, the representation can be on an $O(m) \times O(n + m)$ grid.

Given an instance of Monotone Rectilinear Planar 3-SAT Φ, we will construct an imprecise 2.5D terrain.

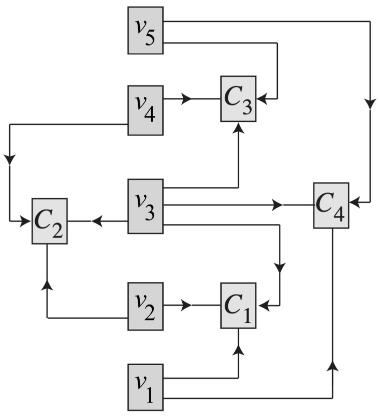
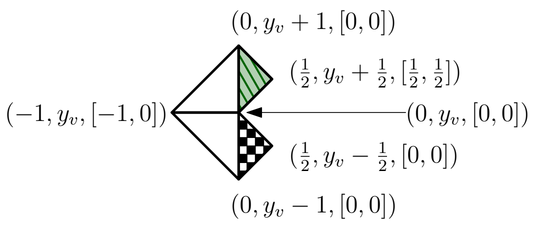
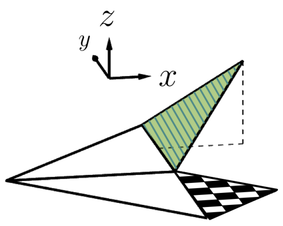


Figure 7 An instance of Monotone Rectilinear Planar 3-SAT, modified so the clauses have fixed height. The corresponding 3-SAT formula is $C_1 \wedge C_2 \wedge C_3 \wedge C_4$ where, for example, $C_1 = v_1 \vee v_2 \vee v_3$ and $C_2 = \neg v_2 \vee \neg v_3 \vee \neg v_4$.



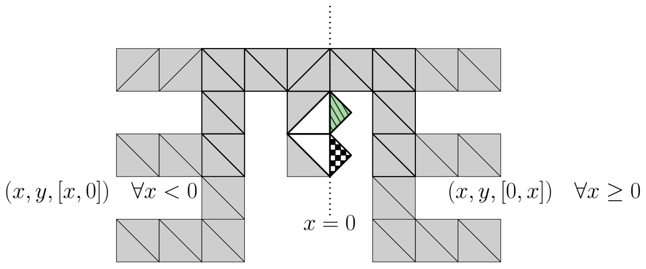
$(0, y_v + 1, [0, 0])$
$(\frac{1}{2}, y_v + \frac{1}{2}, [\frac{1}{2}, \frac{1}{2}])$
$(-1, y_v, [-1, 0])$
$(0, y_v, [0, 0])$
$(\frac{1}{2}, y_v - \frac{1}{2}, [0, 0])$
$(0, y_v - 1, [0, 0])$

(a) A top view of the variable gadget.



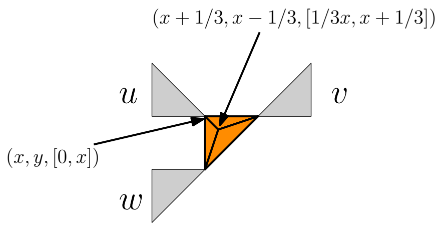
(b) A 3D depiction of the gadget (false setting).

Figure 8 The variable gadget.

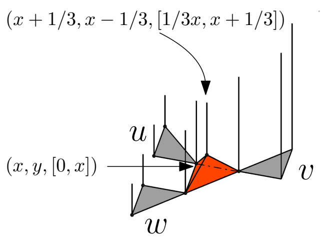


$(x, y, [x, 0])$ $\forall x < 0$
$(x, y, [0, x])$ $\forall x \geq 0$
$x = 0$

Figure 9 Variable component.



(a) A top view of the clause gadget.



(b) A 3D depiction of the clause gadget.

Figure 10 Clause gadget for positive literals.

Variable gadget and component. The ***variable gadget*** for variable v, shown in Figure 8, consists of four triangles: two ***selector*** triangles (in white); a *true* triangle (green striped), which we force onto the plane $x = z$; and a *false* triangle (checkered), which we force onto the horizontal plane $z = 0$. The z-interval of the leftmost vertex of the gadget extends between the two planes, which permits the two selector triangles to be coplanar with the *true* triangle or with the *false* triangle. Thus, if the variable gadget is limited to two patches [or to one bend], then the selector triangles "select" a true/false value for the variable. The gadget for variable v is placed inside v's input rectangle—see Figure 8a for the exact x, y-coordinates and z-intervals.

To model the wires in the input, we expand v's variable gadget to a ***variable component*** by constructing ***chains*** of ***path*** triangles as shown in Figure 9. The associated z-intervals are large enough to permit all path triangles to be coplanar with v's *true* triangle (lying in the plane $x = z$) or with v's *false* triangle (lying in the horizontal plane $z = 0$). If the variable component is limited to two patches [or to one bend], then the choice made by the selector triangles is transmitted to all the path triangles.

Clause gadget. The gadget for clause c, shown in Figure 10a and 10b, consists of three triangles sharing a ***centre vertex*** and joining the three final vertices of the chains corresponding to the variables in the clause. The z-interval of the central vertex is strictly above the $z = 0$ plane for a positive clause, and strictly above the $x = z$ plane for a negative clause. Therefore, for a positive [negative] clause, if all three chains are in the $z = 0$ plane [the $x = z$ plane] (corresponding to setting the literals false), then the three triangles of the clause gadget must form three patches. However, by making the z-interval of the central vertex large enough, we ensure that if at least one chain lies in the other plane (corresponding to setting the literal true), then the central vertex may be chosen to lie in the plane of the other three vertices, thus creating one coplanar patch out of the three clause triangles.

Completing the triangulation. The triangulation T constructed so far has holes and its outer boundary is not convex. Suppose T has bounding box B. Let E be the empty region inside B but outside T. We will add points and triangles to fill in E, thereby obtaining a triangulation whose boundary is the convex hull of its points. Add a new vertex at every integer xy-grid point inside E. Assign these vertices the precise elevation zero. Add the grid edges inside E. This partitions E into grid squares, plus empty pentagons near vertex gadgets, and empty triangles near clause gadgets. See Figure 11. Add one new ***spike*** vertex inside each such region, and triangulate the region into ***spike triangles*** formed by connecting the spike vertex to each vertex of its region. Each spike vertex is assigned a precise z-coordinate

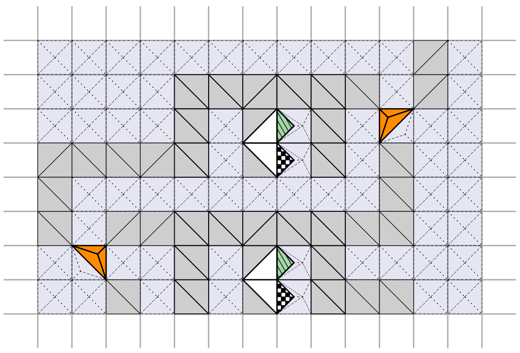


Figure 11 A portion of the final construction showing how spike triangles (in blue) fill in the triangulation.

at least four times lower than anything in T.

By this choice of z, each spike triangle must form one patch by itself, i.e., no spike triangle can be coplanar with an adjacent triangle. (A detailed proof of this can be found in the Master's thesis of the second author [22].) The full reduction from the 3-SAT instance in Figure 7 is shown in Figure 12.

Correctness. Note that this construction takes polynomial time. In particular, since the input for Monotone Rectilinear Planar 3-SAT lay on an $O(m) \times O(n+m)$ grid, the coordinates we construct are polynomially bounded.

▶ **Lemma 9.** *Let $k = 2n + m + s$, where s is the number of spike triangles. Then there is a satisfiable truth-value assignment for Φ if and only if there is a selection of elevations z that creates a terrain with at most k patches.*

Proof. Suppose there is a satisfiable truth-value assignment for Φ. Choose elevations that put the variable component of each true variable in the $x = z$ plane and the variable component of each false variable in the $z = 0$ plane. This creates $2n$ patches. Since each clause has at least one true literal, we can choose the elevation of the centre vertex of each clause gadget so that the clause gadget uses one patch. This creates m patches. Finally, each spike triangle is one patch, so the total number of patches is $2n + m + s = k$.

For the other direction, suppose there is a precise terrain with at most k patches. Each spike triangle forms one patch, each variable component forms at least two patches, and each clause gadget forms at least one patch (note that variable components do not share *edges* with clause gadgets). Thus each variable component must use two patches (thus forcing the three outer vertices of each clause gadget to respect the true/false choices), and each clause gadget must use one patch (thus requiring at least one of its literals to be true). ◀

A similar argument proves that this reduction also works for bends:

▶ **Lemma 10.** *Let $k = n + S$, where S is the number of edges shared between two triangles where at least one of them is a spike triangle. Then there is a satisfiable truth-value assignment for Φ if and only if there is a selection of elevations z that creates a terrain with at most k bends.*

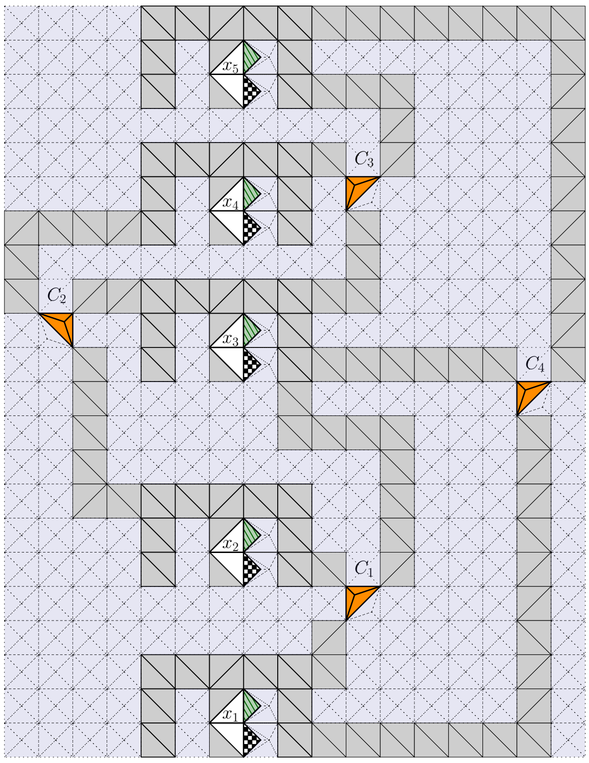


Figure 12 The imprecise 2.5D terrain constructed from the instance of 3-SAT in Figure 7.

5 Surface area

We show that the surface area of an imprecise 2.5D terrain can be minimized using Second Order Cone Programming (SOCP) [2, Section 4.4.2] which is an extension of Linear Programming, with additional constraints of the form $\|Ax+b\| \leq c^\top x + d$, where $\|\cdot\|$ represents the Euclidean (L_2) norm. More precisely:

▶ **Definition 11.** *A* ***Second Order Cone Program*** *(a SOCP) is an optimization problem of the following form:*

Input:

- $m, n, n_1, \ldots, n_m, k \in \mathbb{N}$
- $A_i \in \mathbb{R}^{n_i \times n}$, $b_i \in \mathbb{R}^{n_i}, c_i \in \mathbb{R}^n, d_i \in \mathbb{R}$ *for all* $i = 1, \ldots, m$.
- $E \in \mathbb{R}^{k \times n}, f \in \mathbb{R}^k, w \in \mathbb{R}^n$

Output: $x \in \mathbb{R}^n$
Objective:

$$\begin{array}{lll} \textit{minimize } w^\top x & & \\ \textit{subject to} & \|A_i x + b_i\| \leq c_i^\top x + d_i & i = 1, \ldots, m \\ & Ex = f & \end{array}$$

Second Order Cone Programs can be solved with additive error ε in time polynomial in the size of the input and $\log(\frac{1}{\varepsilon})$ using interior point methods [2, Section 4.4.2]. This is efficient, although not polynomial time.

To model minimizing the surface area of an imprecise 2.5D terrain as a SOCP, we use variables $z_i, i = 1, \ldots, n$ for the elevations, and the linear constraints $b_i \leq z_i \leq t_i$ to ensure that each elevation value is within its interval. For each triangle T, a variable s_T will upper bound the area of T, via the constraint $\text{area}(T) \leq s_T$. Then minimizing the linear objective function $\sum_{T \in \mathcal{T}} s_T$ guarantees that the total surface area is minimized.

We only need to show that $\text{area}(T) \leq s_T$ is a valid SOCP constraint. If T has imprecise vertices p_1, p_2, p_3, then $\text{area}(T)$ is $\frac{1}{2}\|(p_2 - p_1) \times (p_3 - p_1)\|$, where $\times$ is cross product. Because x- and y-coordinates of the vertices are fixed, $(p_2 - p_1) \times (p_3 - p_1)$ is a linear function of the z variables.

6 Min max steepness

We show that minimizing the maximum steepness of an imprecise 2.5D terrain can be formulated as a Second Order Cone Program (as defined in the previous section). The ***steepness*** of triangle T lying on the plane $z = A_T x + B_T y + C_T$ is the L_2 norm of the gradient vector (A_T, B_T), i.e., $\|(A_T, B_T)\|$.

As above, we use variables z_i for the elevations, together with the linear constraints $b_i \leq z_i \leq t_i$. For each triangle T, we introduce variables A_T, B_T, C_T representing the coefficients of the plane containing T, as captured by the constraints $z_i = A_T x_i + B_T y_i + C_T$ for each vertex (x_i, y_i, z_i) of T. Finally, we add constraints $\|(A_T, B_T)\| \leq F$ for one new variable F. Then minimizing F will minimize the maximum steepness.

7 Conclusion

For imprecise 1.5D terrains, we gave linear time exact algorithms for three objectives, but could only achieve a 2-approximation for minimizing the number of bends. We believe that minimizing the number of bends for an imprecise 1.5D terrain is weakly NP-hard. Is the problem fixed parameter tractable in the number of bends?

There are two very natural special cases of the 2.5D imprecise terrain problem that we have not addressed. The Delaunay triangulation is the triangulation of choice for many GIS applications. Do any of the problems become easier for a Delaunay triangulation? Our NP-hardness proof for minimizing the number of patches/bends does not apply to Delaunay triangulations, nor does the NP-hardness proof of Gray et al. [12] for minimizing local extrema.

Another special case of practical import is when all the imprecise elevation intervals have the same length. Again, our NP-hardness proof does not apply. The NP-hardness proof of Gray et al. [12] uses two interval lengths, one positive and one zero (i.e., some vertices are given precisely). Does the restriction to equal-length elevation intervals make any of the problems easier?

Another direction worth exploring is imprecise 2.5D terrains when the triangulation is not fixed, so the input consists only of imprecise points, and the problem is to find precise points and a triangulation for the given objective. Even if the points are given precisely, choosing the best triangulation can be NP-hard, as shown by De Kok et al. [7] for minimizing the number of extrema. Are any of the other objectives NP-hard when the triangulation is not fixed, either for precise or imprecise points? Minimizing the number of patches is closely related to the following terrain simplification problem: given a large set P of points with imprecise elevation intervals, select a subset S of the points, precise elevations for the points in S, and a triangulation of S—thus determining a precise terrain T on S—so that for any point in $P \setminus S$, its imprecise elevation interval intersects T. In other words, every point of $P \setminus S$ can be given a precise elevation and added to the terrain so that all its incident triangles are coplanar.

Acknowledgements. We thank Joseph O'Rourke and the other anonymous reviewers for careful reading and many helpful suggestions. We thank Therese Biedl for valuable comments on the second author's thesis [22].

References

1 Helmut Alt and Leonidas J. Guibas. Discrete geometric shapes: Matching, interpolation, and approximation. In *Handbook of Computational Geometry*, pages 121–153. Elsevier, 2000.

2 Stephen Boyd and Lieven Vandenberghe. *Convex Optimization.* Cambridge University Press, 2004.

3 Kevin Buchin, Maarten Löffler, Aleksandr Popov, and Marcel Roeloffzen. Uncertain Curve Simplification. In Filippo Bonchi and Simon J. Puglisi, editors, *46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021)*, volume 202 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 26:1–26:22, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: `https://drops.dagstuhl.de/opus/volltexte/2021/14466`, `doi:10.4230/LIPIcs.MFCS.2021.26`.

4 Bernard Chazelle. Triangulating a simple polygon in linear time. *Discrete & Computational Geometry*, 6(3):485–524, 1991.

5 Yi-Jen Chiang and Roberto Tamassia. Optimal shortest path and minimum-link path queries between two convex polygons inside a simple polygonal obstacle. *International Journal of Computational Geometry & Applications*, 7:85–121, 1997.

6 Mark de Berg and Amirali Khosravi. Optimal binary space partitions in the plane. In My T. Thai and Sartaj Sahni, editors, *Computing and Combinatorics*, volume 6196 of *Lecture Notes in Computer Science*, pages 216–225, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

7 Thierry De Kok, Marc Van Kreveld, and Maarten Löffler. Generating realistic terrains with higher-order delaunay triangulations. *Computational Geometry*, 36(1):52–65, 2007.

8 Reza Dorrigiv, Robert Fraser, Meng He, Shahin Kamali, Akitoshi Kawamura, Alejandro López-Ortiz, and Diego Seco. On minimum-and maximum-weight minimum spanning trees with neighborhoods. *Theory of Computing Systems*, 56(1):220–250, 2015.

9 Anne Driemel, Herman Haverkort, Maarten Löffler, and Rodrigo Silveira. Flow computations on imprecise terrains. *Journal of Computational Geometry*, 4(1):38–78, 2013.

10 C. Gray and W. Evans. Optimistic shortest paths on uncertain terrains. In *Proceedings of the 16th Canadian Conference on Computational Geometry (CCCG'04)*, pages 68–71, Montréal, Canada, 2004.

11 Chris Gray. Shortest paths on uncertain terrains. Master's thesis, University of British Columbia, 2004. URL: `https://open.library.ubc.ca/collections/ubctheses/831/items/1.0051736`, `doi:http://dx.doi.org/10.14288/1.0051736`.

12 Chris Gray, Frank Kammer, Maarten Löffler, and Rodrigo I. Silveira. Removing local extrema from imprecise terrains. *Computational Geometry*, 45(7):334–349, 2012. `doi:https://doi.org/10.1016/j.comgeo.2012.02.002`.

13 Chris Gray, Maarten Löffler, and Rodrigo I Silveira. Smoothing imprecise 1.5D terrains. *International Journal of Computational Geometry & Applications*, 20(04):381–414, 2010.

14 Hiroshi Imai and Masao Iri. Polygonal approximations of a curve — formulations and algorithms. In Godfried T. Toussaint, editor, *Computational Morphology*, volume 6 of *Machine Intelligence and Pattern Recognition*, pages 71–86. North-Holland, 1988. URL: `https://www.sciencedirect.com/science/article/pii/B9780444704672500114`, `doi:https://doi.org/10.1016/B978-0-444-70467-2.50011-4`.

15 Vahideh Keikha, Maarten Löffler, Ali Mohades, and Zahed Rahmati. Width and bounding box of imprecise points. In *Proceedings of the 30th Canadian Conference on Computational Geometry (CCCG'18)*, pages 142–148, Winnipeg, Canada, 2018.

16 Yury Kholondyrev. Optimistic and pessimistic shortest paths on uncertain terrains. Master's thesis, University of British Columbia, 2007.

17 Maarten Löffler and Jack Snoeyink. Delaunay triangulation of imprecise points in linear time after preprocessing. *Computational Geometry*, 43(3):234–242, 2010.

18 Maarten Löffler and Marc van Kreveld. Largest and smallest convex hulls for imprecise points. *Algorithmica*, 56(2):235–269, 2010.

19 Maarten Löffler and Marc van Kreveld. Largest bounding box, smallest diameter, and related problems on imprecise points. *Computational Geometry*, 43(4):419 – 433, 2010. doi:https://doi.org/10.1016/j.comgeo.2009.03.007.

20 Florian A. Potra and Stephen J. Wright. Interior-point methods. *Journal of Computational and Applied Mathematics*, 124(1):281–302, 2000. Numerical Analysis 2000. Vol. IV: Optimization and Nonlinear Equations. URL: https://www.sciencedirect.com/science/article/pii/S0377042700004337, doi:https://doi.org/10.1016/S0377-0427(00)00433-7.

21 David Salesin, Jorge Stolfi, and Leonidas Guibas. Epsilon geometry: building robust algorithms from imprecise computations. In *Proceedings of the Fifth Annual Symposium on Computational Geometry*, pages 208–217, 1989.

22 Graeme Stroud. Computing realistic terrains from imprecise elevations. Master's thesis, University of Waterloo, 2022. URL: https://uwspace.uwaterloo.ca/handle/10012/18155.

23 Subhash Suri. A linear time algorithm for minimum link paths inside a simple polygon. *Computer Vision, Graphics, and Image Processing*, 35(1):99–110, 1986. doi:https://doi.org/10.1016/0734-189X(86)90127-1.

24 Marc van Kreveld, Maarten Löffler, and Lionov Wiratma. On Optimal Polyline Simplification Using the Hausdorff and Fréchet Distance. In Bettina Speckmann and Csaba D. Tóth, editors, *34th International Symposium on Computational Geometry (SoCG 2018)*, volume 99 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 56:1–56:14, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. URL: http://drops.dagstuhl.de/opus/volltexte/2018/8769, doi:10.4230/LIPIcs.SoCG.2018.56.