Computing Realistic Terrains from Imprecise Elevations*

Anna Lubiw† Graeme Stroud†

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 problems #1, #3, and #4 to a shortest path problem, and show that problem #2 can be 2-approximated via a minimum link path.

In 2.5D, problem #1 was proved NP-hard by Gray et al. [Computational Geometry, 2012]. We give a polynomial time algorithm for a triangulation of a polygon. We prove that problem #2 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 can be solved efficiently via Second Order Cone Programming.

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 [9]. 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 plane, there is work on minimizing/maximizing the width, the area of the bounding box, or the diameter of the points [11, 13].

For the case of terrains, Gray and Evans [8], and Gray [7] formulated the ***imprecise 2.5D terrain model***. In this model, the x, y-coordinates of points are given as input, in addition to a triangulation defined on the points when projected to the xy plane, but the z-coordinate (elevation) of each point is only known "imprecisely" within some interval of possible values. We obtain a ***precise 2.5D terrain*** by choosing a precise elevation from each uncertainty interval and connecting the points together according to the input triangulation. Various "niceness" criteria for choosing a precise terrain have been considered in the past such as minimizing the number of local extrema [9], or minimizing the length of the shortest path along the terrain from one point to another [8, 12].

When these problems are NP-hard or have unknown computational complexity for 2.5D terrains, researchers (e.g., Gray et al. [10]) 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.

Objective #1: Minimizing 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. [9]. We give a polynomial time algorithm for the special case of a triangulation of a polygon, and solve the 1.5D version in

*Results from Master's thesis of the second author [16].

†School of Computer Science, University of Waterloo, alubiw@uwaterloo.ca, graeme.stroud@uwaterloo.ca

	#1: extrema	#2: coplanar	#3: area/length	#4: steepness
1.5D	**$O(n)$ [§2]**	**2-approx [§2]**	**$O(n)$ [§2]**	**$O(n)$ [§2]**
2.5D special	**$O(n^4)$ [§3] (polygon triangulation)**	**5-approx [§4.1] (strip triangulation)**		
2.5D general	NP-hard [9]	**NP-hard [§4.2]**	**SOCP [§5]**	**SOCP [§6]**

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

linear time via a shortest path.

Objective #2: Optimizing 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 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).

Objective #3: Minimizing surface area/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 [15] (though not in polynomial time).

Objective #4: Minimizing 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. We consider minimizing the maximum steepness. 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.

Background. Gray [7] was the first to consider the imprecise terrain model. (See also Gray and Evans [8].) The problem they considered was finding the shortest path from one point to another over all precise realizations of the terrain. Various other objective functions 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. [9], and they show there is no $O(\log \log n)$ approximation algorithm unless P = NP. Driemel et al. [6] 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. [10] 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.

2 1.5D Terrains

In this section we show that optimal 1.5D terrains for Objectives #1, #3, and #4 can be computed in linear time by finding a shortest path in an appropriate polygon, and that a minimum link path in the polygon provides a linear time 2-approximation for Objective #2.

Suppose the input to the problem has n points where, for $i = 1, \ldots, n$, point i must lie in segment ℓ_i at x-coordinate x_i, with $x_1 < x_2 < \cdots x_n$. Let P be the simple polygon whose vertices are the top and bottom endpoints of the segments, 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. We use a shortest path from ℓ_1 to ℓ_n, which is unique unless it is a straight horizontal path that can shift up/down.

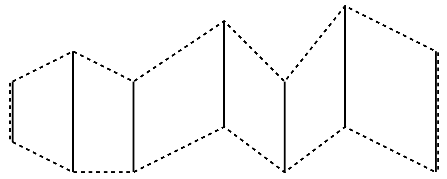


Figure 1: The input segments for the imprecise 1.5D terrain problem (solid) and the polygon P (dashed).

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 Objectives #1, #3, and #4.*

Proof. The shortest path from one segment to another in a simple polygon can be found in linear time [3]. This algorithm needs a triangulation of the polygon. Thankfully, we do not need Chazelle's impractical linear time algorithm [2], since P is composed of trapezoids each of which can be cut into two triangles.

Note that a shortest path in a polygon only bends at the polygon vertices. The vertices of polygon P are

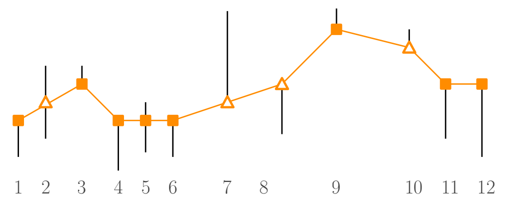


Figure 2: A shortest path terrain with five extreme plateaus (marked by squares).

endpoints of segments, and therefore a shortest path from ℓ_1 to ℓ_n provides a 1.5D terrain. (As discussed below, the fact that a minimum link path may bend at non-vertex points is why we can only achieve a 2-approximation for Objective #2.)

The theorem is obvious for Objective #3 (minimizing length). We next consider Objectives #1 and #4.

Objective #1. Suppose the shortest path has k 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, $2 \le j \le n-1$.

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 4 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] the points 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 $\ell_{i_{j+2}}$.

Objective #4. 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. We call this ***lexicographically minimizing the maximum steepness.*** In the full version [14] we prove the following.

Proposition 1 *A shortest path from ℓ_1 to ℓ_n in polygon P lexicographically minimizes the maximum steepness.*

This completes the proof of Theorem 1. □

We now turn to Objective #2, minimizing 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 [17]. (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 2 *Let π be a minimum link path from ℓ_1 to ℓ_n in P. Then 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, as defined in the Introduction. Note that we allow input triangulations that do not include all the convex hull edges of the projected 2D points (to model triangulating a general shape). In this section we consider Objective #1, minimizing the number of local extrema.

Gray et al. [9] 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 3 *There is an $O(n^4)$ dynamic programming algorithm to minimize the number of extrema for imprecise 2.5D terrains when the triangulation is of a polygon.*

The following claim (proved in the full version [14]) shows that we can restrict to a discrete set of elevation values.

Claim 4 *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 $i = 1, \ldots, n$.*

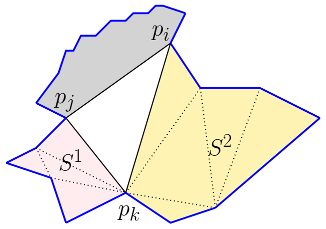


Figure 3: Splitting subproblem $S_{i,j}$ into $S^1 := S_{k,j}$ (pink) and $S^2 := S_{i,k}$ (yellow).

We will now describe the algorithm. Label the vertices around the polygon $p_1, \ldots, p_n$ in clockwise order. For each edge p_ip_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 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 etc. 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_ip_j in the triangulation, and for each, we try $O(n^2)$ 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 $O(n^4)$.

4 Coplanar Features

In this section, we explore the problem of minimizing the number of patches/bends. 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, see Figures 4 and 5. However, they are equivalent for a triangulation of a polygon—as we prove in the full version [14], the number of patches will be the number of bends plus one.

4.1 An algorithm for a strip triangulation

A strip triangulation is a special case of a triangulation of a polygon, so we can give an algorithm that works for both objectives.

Theorem 5 *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.*

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.

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

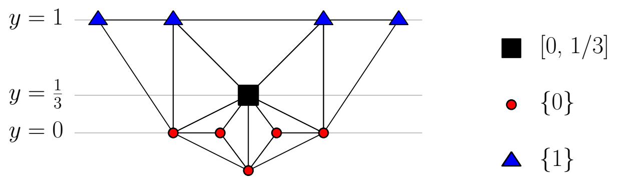


Figure 4: Proving that minimizing the number of bends is not always equivalent to minimizing the number of patches. The central point (drawn with a big black square) is the only one with a non-trivial z-interval, viz. $[0, 1/3]$.

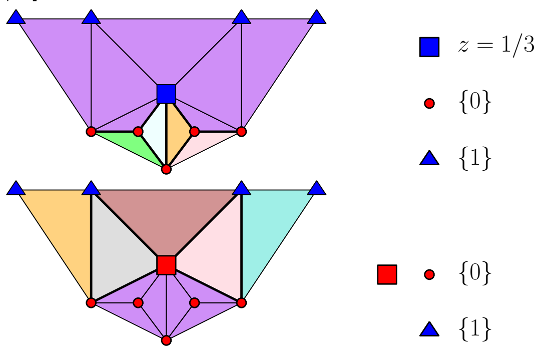


Figure 5: Solution (top) : If we use the top end of the black interval, we get 5 patches (optimal) and 7 bends. Solution (bottom) : If we use the bottom end of the black interval, we get 6 patches and 6 bends (optimal).

plan is to start the next greedy step from 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 second greedy step. 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 colinear. 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 2 to find a 2-approximation for the minimum number of bends.

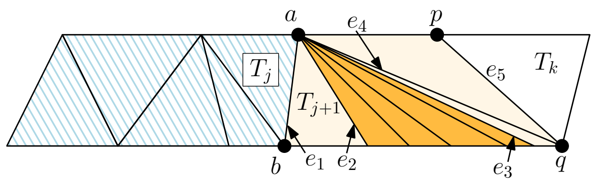


Figure 6: The first iteration of the algorithm. Fan F is colored in dark orange.

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 2. Thus $5B^* \geq 5 + 5s^* \geq B$. Applying induction proves the approximation ratio is correct for the whole input.

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—see [16].

Theorem 6 *Minimizing the number of patches [or bends] is strongly NP-complete in the general setting.*

Containment in NP is proved in the full version [14]—a non-deterministic guess for the patches/bends can be verified in polynomial time using linear programming.

Reduction details. The reduction will be from the NP-complete problem Monotone Rectilinear Planar 3-SAT [4]. 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 ("wires") 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 and adding vertical segments to the wires. 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.

Variable gadget and component. The ***variable gadget*** for variable v, shown in Figure 8a, 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

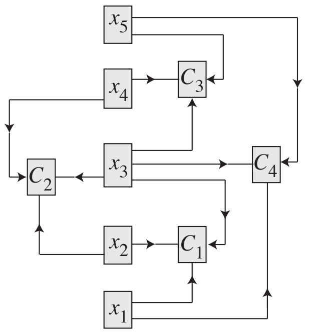


Figure 7: An instance of Monotone Rectilinear Planar 3-SAT, modified so the clauses have fixed height.

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, 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 8b. 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, 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 8c and 9, 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 variable components and clause gadgets can be completed to a triangulation by filling in the holes with ***spike*** triangles, each of which has one new vertex that is off the xy-integer grid and that we force to a z-coordinate at least four times lower than anything constructed so far. By this choice of z, each spike triangle must form one patch. See Figure 10. Figure 11 illustrates the full reduction from the 3-SAT instance in Figure 7.

Lemma 7 *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. (sketch) 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 $2n + m + s = k$.

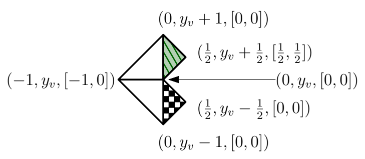
For the other direction, suppose there is a precise terrain with at most k patches. Each spike triangle forms one patch, each variable component forms at least two patches, and each clause gadget forms at least one patch (note that variable components do not share *edges* with clause gadgets). Thus each 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). $\square$

5 Surface Area

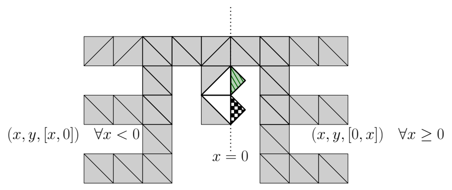
We show that the surface area of an imprecise 2.5D terrain can be minimized using Second Order Cone Programming [1, Section 4.4.2] which is an extension of Linear Programming, with additional constraints of the form $\|Ax + b\| \leq c^\top x + d$, where $\|\cdot\|$ represents the Euclidean (L_2) norm. 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. This is efficient, although not polynomial time.

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 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.

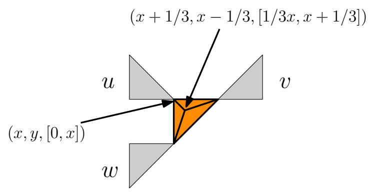
We only need to show that area$(T) \leq s_T$ is a valid SOCP constraint. If T has imprecise vertices p_1, p_2, p_3, then area(T) is $\frac{1}{2}\|(p_2 - p_1) \times (p_3 - p_1)\|$, where $\times$ is cross product. Because x and y are fixed, $(p_2 - p_1) \times (p_3 - p_1)$ is a linear function of the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


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

(a) Variable gadget


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

(b) Variable component



(c) Clause gadget (positive literals)

Figure 8: Gadgets for the NP-hardness reduction.

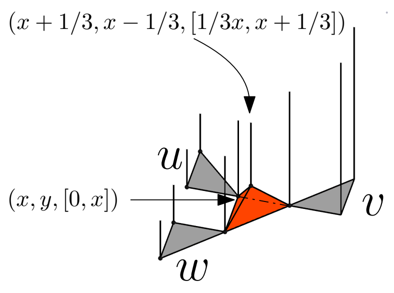


Figure 9: A 3D depiction of the clause gadget from Figure 8c.

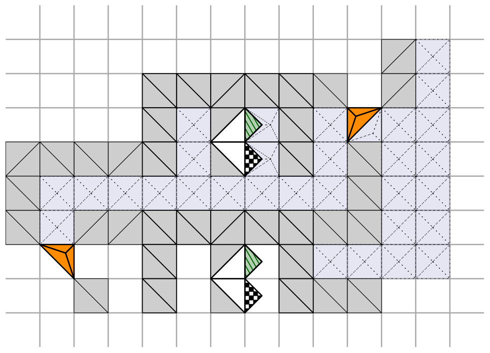


Figure 10: A portion of the final construction showing how spike triangles (in blue) fill in the triangulation.

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, 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?

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. [5] for minimizing extrema. Are any of the other objectives NP-hard when the triangulation is not fixed, either for precise or imprecise points?

References

[1] S. Boyd and L. Vandenberghe. *Convex Optimization.* Cambridge University Press, 2004.

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

[3] Y.-J. Chiang and R. 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.

[4] M. de Berg and A. Khosravi. Optimal binary space partitions in the plane. In M. T. Thai and S. Sahni, editors, *Computing and Combinatorics*, volume 6196 of *Lecture Notes in Computer Science*, pages 216–225, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[5] T. De Kok, M. Van Kreveld, and M. Löffler. Generating realistic terrains with higher-order delaunay triangulations. *Computational Geometry*, 36(1):52–65, 2007.

[6] A. Driemel, H. Haverkort, M. Löffler, and R. Silveira. Flow computations on imprecise terrains. *Journal of Computational Geometry*, 4(1):38–78, 2013.

[7] C. Gray. Shortest paths on uncertain terrains. Master's thesis, University of British Columbia, 2004.

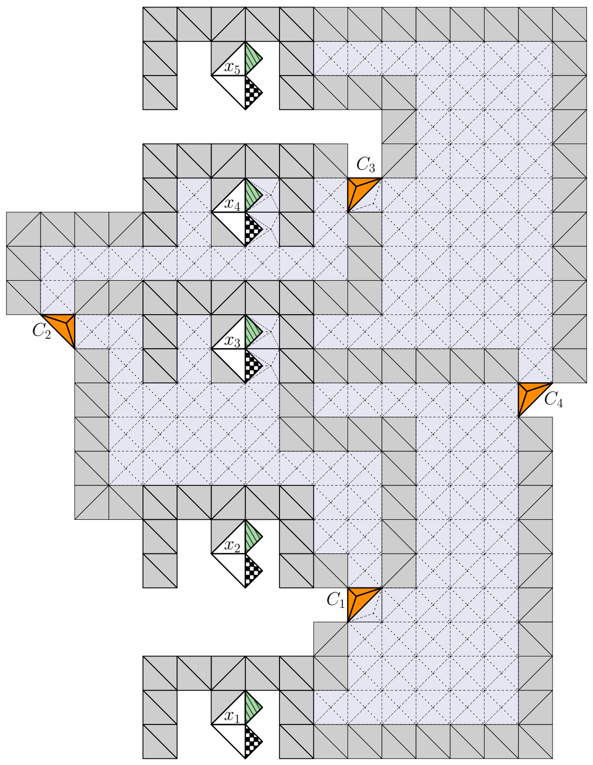


Figure 11: The resulting imprecise 2.5D terrain constructed from the instance of 3-SAT in Figure 7.

[8] 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.

[9] C. Gray, F. Kammer, M. Löffler, and R. I. Silveira. Removing local extrema from imprecise terrains. *Computational Geometry*, 45(7):334–349, 2012.

[10] C. Gray, M. Löffler, and R. I. Silveira. Smoothing imprecise 1.5D terrains. *International Journal of Computational Geometry & Applications*, 20(04):381–414, 2010.

[11] V. Keikha, M. Löffler, A. Mohades, and Z. 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.

[12] Y. Kholondyrev. Optimistic and pessimistic shortest paths on uncertain terrains. Master's thesis, University of British Columbia, 2007.

[13] M. Löffler and M. van Kreveld. Largest bounding box, smallest diameter, and related problems on imprecise points. *Computational Geometry*, 43(4):419 – 433, 2010.

[14] A. Lubiw and G. Stroud. Computing realistic terrains from imprecise elevations. *arxiv*, 2022. to appear.

[15] F. A. Potra and S. 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.

[16] G. Stroud. Computing realistic terrains from imprecise elevations. Master's thesis, University of Waterloo, 2022.

[17] S. Suri. A linear time algorithm for minimum link paths inside a simple polygon. *Computer Vision, Graphics, and Image Processing*, 35(1):99–110, 1986.