Removing Local Extrema from Imprecise Terrains

Chris Gray[*] Frank Kammer[†] Maarten Löffler[‡] Rodrigo I. Silveira[§]

Abstract

In this paper we consider imprecise terrains, that is, triangulated terrains with a vertical error interval in the vertices. In particular, we study the problem of *removing* as many local extrema (minima and maxima) as possible from the terrain; that is, finding an assignment of one height to each vertex, within its error interval, so that the resulting terrain has minimum number of local extrema. We show that removing only minima or only maxima can be done optimally in $O(n \log n)$ time, for a terrain with n vertices. Interestingly, however, the problem of finding a height assignment that minimizes the total number of local extrema (minima as well as maxima) is NP-hard, and is even hard to approximate within a factor of $O(\log \log n)$ unless $\mathrm{P} = \mathrm{NP}$. Moreover, we show that even a simplified version of the problem where we can have only three different types of intervals for the vertices is already NP-hard, a result we obtain by proving hardness of a special case of 2-Disjoint Connected Subgraphs, a problem that has lately received considerable attention from the graph-algorithms community.

1 Introduction

Digital terrain analysis is an important part of geographical information science, with applications in hydrology, geomorphology, visualization, and many other fields [7]. A popular structure for representing terrains is the *triangulated irregular network (TIN)*, also known as *polyhedral terrain*. In this model, a terrain is represented by a planar triangulation with a height associated with each vertex. If we linearly interpolate the heights of the vertices, we also obtain a height at every other point in the plane, resulting in a bivariate, piecewise linear and continuous function, defining the surface of the terrain. A terrain in this model is also often called a 2.5-dimensional (or 2.5D) terrain.

1.1 Imprecision in Terrains

In computational geometry it is usually assumed that the input data for any problem is correct and known exactly. In practice, this is unfortunately not the case. There are many sources of imprecision, the most prominent of which is the data acquisition itself. In terrain modeling, this is particularly relevant, because elevation data is collected by measuring devices that are ultimately error-prone. Often such devices produce heights with a known error bound or return a height interval rather than a fixed height value.

In order to handle the imprecision in terrains, we adopt the model used in [8, 9, 12], where the height of each terrain vertex is not precisely known, but only an interval of possible heights is available. This results in considerable freedom in the terrain, since the "real" terrain is unknown

[*]Department of Computer Science, TU Braunschweig, Germany, gray@ibr.cs.tu-bs.de
[†]Institut für Informatik, Universität Augsburg, Germany, kammer@informatik.uni-augsburg.de
[‡]Computer Science Department, University of California, Irvine, USA, mloffler@uci.edu
[§]Dept. de Matemàtica Aplicada II, Universitat Politècnica de Catalunya, Spain, rodrigo.silveira@upc.edu

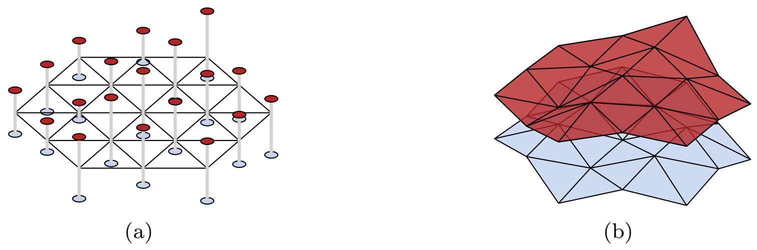


Fig. 1: (a) An example of an imprecise terrain. (b) The same terrain, shown by drawing the floor and the ceiling.

and any choice of a height for each vertex—as long as it is within its height interval—leads to a valid *realization* of the imprecise terrain. The large number of different realizations of an imprecise terrain leads naturally to the problem of finding one that is 'best' according to some criterion, or that avoids most or many instances of a certain type of unwanted feature (or artifact) from the terrain.

We note that, even though terrain data may contain error also in the x, y-coordinates, under this model we consider imprecision only in the z-coordinate. This simplifying assumption is justified by the fact that error in the x, y-coordinates will most likely produce elevation error. Moreover, often the data provided by commercial terrain data suppliers only reports the elevation error [6].

In the remainder of this paper, an *imprecise terrain* is a set of n vertical intervals in $\mathbb{R}^3$, together with a triangulation of the vertical projection. Figure 1(a) shows an example. A *realization* of an imprecise terrain is a triangulated terrain that has the same triangulation in the projection, and exactly one vertex on each interval. An alternative way to view an imprecise terrain is by connecting the tops of all intervals into a terrain, which we call the *ceiling*, and the bottoms into a second terrain, which we call the *floor*. Then, a realization is a terrain that lives in the space left open between the floor and the ceiling. Figure 1(b) shows this in the example.

1.2 Removing local extrema

A local minimum (or pit) is a location on a terrain that is surrounded by higher points. Similarly, a local maximum (or peak) is as a point surrounded by lower points. The term *local extrema* will be used to refer to both local minima and local maxima.

When terrains are used for studies of land erosion, landscape evolution, or hydrology, it is generally accepted that the majority of local extrema in the terrain model are spurious, caused by errors in the data or model production. A terrain model with many pits or peaks does not represent the terrain faithfully, and moreover, in the case of pits, it can create problems because water accumulates at them, affecting water flow routing simulations. For this reason the removal of local minima from terrain models is a standard preprocessing requirement for many uses of terrain models [22, 25]. However, existing preprocessing routines make no attempt to relate the removed minima to knowledge about the imprecision in the terrain model, possibly causing major alterations to the data under study.

In this paper we attempt to solve the problems of removing as many local minima, maxima, or extrema as possible by moving the vertices of an *imprecise* terrain within their allowed height intervals. The rationale behind this is that if a pit (or peak) can be removed in this way, it is likely to be an artifact of the data, whereas if it cannot, it is more certain to be a 'real' pit (or peak). We define the *minimizing-minima*, the *minimizing-maxima*, and the *minimizing-extrema* problems on imprecise terrains, where we attempt to find a realization of an imprecise terrain (by placing the imprecise points within their intervals) that minimizes the number of *local minima*, *local maxima*, and *local extrema*, respectively.

It is important to note that a group of k connected vertices at the same height without any lower neighbor is considered to be only *one* local minimum. This is reasonable from the point of view of the application, and follows the definitions used in previous work [21]. In Section 4 we discuss what the implications of this modeling choice are for our results.

Regarding previous work, a lot of research has been devoted to the problem of removing local minima from (precise) terrains, especially in the geographic information science community, but also from more algorithmic points of view; we only provide a few relevant references here. Most of the literature assumes a raster (grid) terrain (e.g. [17, 18, 25]), and employs methods that are some type of "pit filling" technique, which consist in filling in depressions until they disappear (e.g. [4, 17, 25]). Some of the few exceptions are the methods in [3, 18, 20]. A few algorithms have been proposed for triangulated terrains, see for example the ones in [1, 16]. The removal of local extrema has also been studied in the context of optimal higher order Delaunay triangulations [5, 10]. In particular, Gudmundsson *et al.* [10] show that an optimal first-order Delaunay triangulation, with respect to the number of local minima and maxima, can be found in $O(n \log n)$ time. More related to this paper, Silveira and Van Oostrum [21] study moving vertices vertically in order to remove all local minima with a minimum cost, but do not assume bounded intervals.

1.3 Results

In Section 2, we first study the problem of finding a realization of an imprecise terrain that minimizes the number of local minima (or local maxima). We show that all potential local minima (resp. maxima) of the terrain are independent, that is, whether we remove one does not influence whether or not we can remove another. Using this property, we then present a relatively simple algorithm that finds a realization of the imprecise terrain with minimum number of local minima (maxima) in $O(n \log n)$ time.

In Section 3, we turn our attention to finding a realization that minimizes the number of minima and maxima, that is, removing both minima and maxima simultaneously. In this case we no longer have the independence property, and as a consequence the problem becomes much harder. In fact, we show that the problem of minimizing the total number of local extrema is NP-hard, even hard to approximate within a factor $O(\log \log n)$ unless $\mathrm{P} = \mathrm{NP}$.

All results mentioned above assume general position of the input (that is, all the top and bottom ends of the intervals have different heights), and consider points chosen to be at the same height as a connected group to be at most one single local extremum. In Section 4, we discuss how these assumptions influence the results presented in Section 3.

Finally, in Section 5, we consider a simplified version of the problem of removing local extrema, where can have only three different types of intervals for the vertices. We show that this problem is already NP-hard, and cannot be approximated within a factor $3/2$. For this, we prove that the planar version of 2-Disjoint Connected Subgraphs is NP-hard. The latter problem has received quite some attention recently, and we consider the connection with this hardness proof to be of independent interest.

2 Removing local minima

We begin with the problem of finding a realization that has the smallest number of local minima, that is, the minimizing-minima problem. We propose an efficient algorithm based on the idea of selectively *flooding* parts of the terrain. The algorithm begins with all vertices as low as possible, and simulates flooding parts of the terrain.

Algorithm Conceptually, we raise all local minima as much as possible, that is, we raise each minimum and its neighbors as we meet them, merging minima as we sweep the terrain bottom-up. The process stops when one of the vertices in a local minimum cannot be raised any further. Also, when there is only one local minimum left and no more higher terrain it could merge into, the process stops.

We sweep a horizontal plane vertically, starting at the lowest interval end and moving upwards in the z direction. As the plane moves up, it *pulls* some of the vertices with it, whose height is changing together with the plane. At any moment during the sweep, each vertex is in one of three states:

(a) Moving, if it is currently part of a local minimum, and is moving up together with the sweep plane.

(b) Fixed, at a height lower than the current height of the sweep plane.

(c) Unprocessed, if it has not been reached by the sweep plane yet.

As the sweep plane moves vertically up, we mark the moments at which something interesting happens as *events*. We precompute these events and put them in an event queue. We distinguish two types of events:

(i) The plane reaches the beginning (lowest end) of the interval of a vertex,

(ii) The plane reaches the end (highest end) of the interval of a vertex.

Let v denote the vertex whose interval just began or ended, and let h be the current height of the plane. Note that all fixed vertices are fixed at a height lower than h.[1]

An event of type (i) can create a number of situations.

If v has a neighbor that is already fixed, then v will never be a local minimum, thus v is fixed at its lowest possible height. Moreover, if some other neighbor of v is currently part of a local minimum (i.e. is moving), then all the vertices part of that local minimum become fixed at h, and automatically stop being a minimum. This occurs for each neighbor of v that is currently part of a local minimum.

If all neighbors of v are currently unprocessed, then v becomes a new local minimum, and starts to move up together with the plane.

Finally, if no neighbor is fixed but some neighbor is moving—thus this neighbor is part of a local minimum—then v will join that existing local minimum and also start to move up together with the plane (note that if there is more than one local minimum that is connected to v, at this step they all merge into one).

Events of type (ii), when an interval ends, are easier to handle. If v is fixed, nothing occurs. If v was moving, then it becomes fixed at h, and the same occurs to all the vertices of the local minimum that contains v. Thus the whole local minimum becomes fixed, and will be present in the final solution.

When all events in the queue have been processed, we finally set the remaining moving vertices to fixed as well.

[1] For simplicity we are assuming in this description that all interval heights are different. The removal of this assumption does not pose any problem for the algorithm.

Correctness Let T be the original imprecise terrain. For a given height h, let T_h be the modified imprecise terrain constructed by the plane-sweep algorithm, where each vertex that has been fixed is precise (has an imprecision interval of length 0), and the remaining vertices have their original imprecision intervals.

Claim 1: For any h, the minimum number of local minima over all realizations of T and T_h are the same.

We prove this claim by induction on the number of events of the sweep (associated with exactly $2n$ height values).

Proof: Before the first event, the claim clearly holds. If we now process an event at some vertex v of type (i), we have to distinguish two cases: First, v has a fixed neighbor. This neighbor guarantees that we can fix v—i.e., changing its interval to length 0—without getting a new local minimum. For the same reason, all moving neighbors of v as well as the neighbors of the neighbors and so on can be fixed, too. In the remaining case of event type (i), we do not modify our terrain. Let us now consider an event of type (ii). If v is already fixed, again, we do nothing. Otherwise, v was moving. Let C be the connected component in the graph induced by the moving vertices that contains v. Our algorithm fixes all vertices of C at the current height of the sweep. Note that all vertices adjacent to a vertex in C have an interval with a lowest end higher than v, i.e., we must have one local minimum among the vertices in C. Thus, taking the minimum over all realizations before the fixing cannot be better than taking the minimum over all realizations after the fixing. By induction we can therefore conclude that the claim still holds. ⊠

Running time To initialize the event queue, we need to sort the end points of all intervals by height. There are n intervals, so this requires $O(n \log n)$ time. We remove all events of type (ii) that come after the last event of type (i).

The rest of the steps can be implemented in linear time as follows. For every vertex, we simply maintain a label that has a value of either *moving*, *fixed* or *unprocessed*.

At an event of type (i) where the sweep plane reaches the bottom of an unprocessed vertex v, we first inspect all neighbor of v to determine which subcase we are in. This takes time proportional to the number of neighbors, and we charge this cost to the edges connecting v to its neighbors. Since we charge each edge at most twice over the whole algorithm, this takes linear time in total. Now, if no neighbor of v is fixed, we simply set the label of v to *moving* in constant time. If some neighbor of v is fixed, we set the label of v to *fixed*, and we start a floodfill (for example using a depth first search) in the graph induced by the moving vertices to find all vertices connected to v that are currently set to *moving*; we set them to *fixed* as well, and we set their height to the current height of the sweep plane. This takes time proportional to the number of vertices that are being fixed plus the number of edges connecting these vertices to other vertices. Since each vertex gets fixed only once, this also amounts to linear work in total.

Events of type (ii) are handled similarly. If v was moving, we set its label to *fixed* and also start the floodfill in the same way.

Note that by multiplying all interval ends with -1, we can solve also the minimizing-maxima problem.

Theorem 1: The minimizing-minima (or minimizing-maxima) problem in an imprecise terrain with n vertices can be solved in $O(n \log n)$ time—no matter if the given intervals for the vertices are in general position or not.

It is interesting to note that when a group of k connected vertices at the same height without any lower neighbors is regarded as k different local minima, the problem can be proved NP-hard. More details on this are given in Section 4.

3 Removing all local extrema

We now move on to the problem of removing all local extrema at the same time. Although the algorithm in the previous section works for both removing minima and removing maxima, it is not possible to use both height assignments simultaneously. We will show in the next section that we can still use the algorithm twice to narrow down the problem, without changing the value of the solution. Unfortunately, such an approach does not help much to find an optimal realization minimizing the number of extrema. In Section 3.2 we give a proof that shows that minimizing-extrema is NP-hard to approximate within a factor of $O(\log \log n)$, using a reduction from Set Cover.

3.1 Canonical form of an imprecise terrain

Recall that the floor F is the realization formed by all lower endpoints of the imprecise vertices, and the ceiling C is the realization formed by all upper endpoints, as shown in Figure 1(b).

Given two realizations X and Y of the same imprecise terrain, we use the notation (X, Y) to refer to the imprecise terrain truncated by X and Y: the bottom interval ends are taken from the heights in X, and the top interval ends from the heights in Y (we assume here that X is never above Y).

We are searching for a surface between the floor and the ceiling that optimizes the number of local extrema. We will run the algorithm in Section 2 on (F, C) to remove the local minima, and call the result F', and run it again on (F, C) to remove local maxima and call the result C'. We call the imprecise terrain (F', C') the *canonical form* of (F, C).

Lemma 1: The imprecise terrain induced by (F', C') is still a valid imprecise terrain which has the same optimal solution for removing local extrema as the original terrain (F, C).

Proof: We need to show two things. To show that (F', C') is still a valid imprecise terrain, we need that the height of any vertex in F' is at least the height of that vertex in C'. If there exists a height h such that the entire floor lies below h and the entire ceiling lies above h, then neither of them will ever raise/lower beyond h, because of the stop condition when there are no new interval events anymore. Otherwise, if a vertex v does not rise higher this is because its plateau (formed by v and its connected component in the graph induced by the moving vertices) hits a point of the ceiling C; clearly a plateau lowering v will never move past this point. In both cases, a plateau rising the floor and one lowering the ceiling of the same vertex never cross each other.

To show that (F', C') has the same optimal solution as (F, C), we need to show that there exists an optimal terrain T^* between F and C that in fact also lies between F' and C'. This is true because if a terrain would have a local minimum below F', we could freely lift it together with its neighbors until it coincides with F'. By the construction of F', we never hit the ceiling during this process, so we never increase the number of minima or maxima. The converse is true for local maxima above C'. $\boxtimes$

Lemma 1 implies that the canonical form (F', C') of an instance (F, C) has a solution as good as in (F, C). Thus, we can use the algorithm of Secion 2 as a kind of a preprocessing step to obtain a terrain that has more structure than the original one: Every remaining local minimum of the floor touches the ceiling, and every remaining local maximum of the ceiling touches the floor. We can show the following:

Lemma 2: The total number of extrema in the optimal solution T^* is never greater than the number of local maxima of the floor F' + the number of local minima of the ceiling C'.

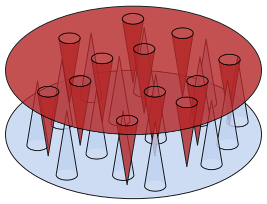


Fig. 2: An imprecise terrain that has many maxima on the floor and many minima on the ceiling.

Proof: Consider the solution $T = C'$. Any local maximum of T must touch one local maximum of F' because when lowering the maxima of C this was exactly the condition on which we stopped. Therefore, the number of local maxima of T is smaller than the number of local maxima of F'. Thus, the number of local extrema in T is at most the number of local maxima of F' + the number of local minima of C' (which is T itself). Clearly, the number of local extrema in the optimal solution T^* can only be even smaller. ☒

If we denote by $\mathrm{lmin}(T)$ the number of local minima in a terrain T and by $\mathrm{lmax}(T)$ the number of local maxima in T, and we denote by T^* the optimal solution of our problem (F, C), then we can summarize these observations as follows:

$$\mathrm{lmin}(F') + \mathrm{lmax}(C') \leq \mathrm{lmin}(T^*) + \mathrm{lmax}(T^*) \leq \mathrm{lmin}(C') + \mathrm{lmax}(F') \qquad (1)$$

This formula gives a lower and an upper bound on the values of a particular instance. In theory, the gap may still be arbitrarily large. For example, consider an instance where the floor is more or less flat except for a number of “stalagmites” that reach all the way to the ceiling, and the ceiling is more or less flat except for a number of “stalactites” that reach all the way to the floor. Figure 2 show such a situation. In this case, the number of maxima of the floor and minima of the ceiling is large, while the floor has only a single minimum and the ceiling has only a single maximum, and the preprocessing step will not make a difference. We see in the next section that this makes the problem very hard to solve.

On the other hand, such terrains seem unlikely to appear in real applications. It may be likely that in practice, the gap in Equation 1 is quite small. Under which properties of terrains this is the case remains an interesting open question.

3.2 Hardness of approximation

In this section we show by a direct reduction from the Set Cover problem that we cannot approximate the number of local extrema on n-vertex graphs within any factor better than $O(\log \log n)$ unless $\mathrm{P} = \mathrm{NP}$.

Given a tuple (U, C), where U is a finite set called *universe* and C is a collection of subsets of U with $\bigcup_{S \in C} S \subseteq U$, a *set cover* for (U, C) is a collection $C' \subseteq C$ such that the union of all sets in C' is equal to U. The *size* of C' is its cardinality. The Set Cover problem is to find a set cover of minimal size.

Note that we can easily recognize a Set Cover-cover instance that has no solution in polynomial time, and we can create a trivial no instance for our minimizing problem. Therefore, let (U, C) be an instance of the Set Cover problem that has a solution. We start by defining a graph G with colored vertices. We then construct a terrain by embedding G in the plane, and triangulating the faces that have more than 3 incident vertices. Finally, we assign heights to the terrain vertices, where the height of every vertex depends on its color.

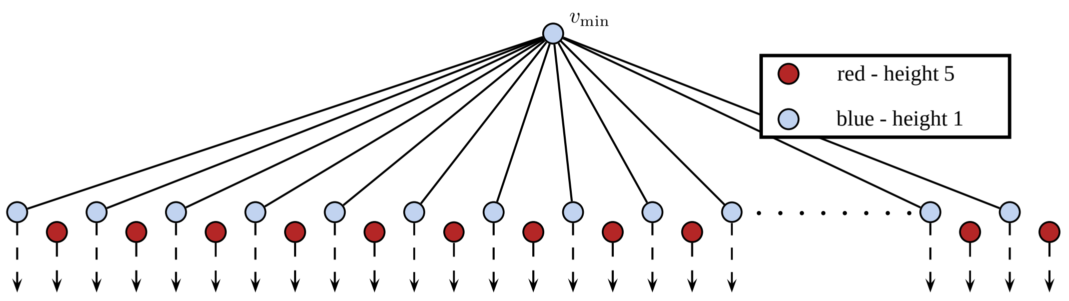


Fig. 3: North gadget.

In the remainder, we will use the terms *west* and *east* to refer to the negative and positive x-direction, and *south* and *north* to refer to the negative and positive y-direction.

We begin by describing the northern edge of the constructed graph, which consists in a *north gadget* depicted in Figure 3. For each item x of the universe U, we introduce $|U| + 3$ red vertices with a blue vertex between each pair of red vertices. All blue vertices are connected to another blue vertex $v^{\min}$ at the north of the construction. Each vertex $v \neq v^{\min}$ in the north gadget is the beginning of a path—that we call *southward path* (indicated in the figures by dashed arrows). Moreover, southward paths are marked as either *covered* or *uncovered*. At this stage, all southward paths starting with a red vertex are considered uncovered, and the ones starting with a blue vertex are covered.

The construction continues by adding one *row-gadget* for each set $S \in C$ (see Figure 4 for a schematic representation). Each row-gadget extends the southward paths southwards and consists of a row of vertices that we call the *decision row*. The westernmost and easternmost vertex of each decision row are, in fact, the same—the edges connected to these vertices meet in the space above the northern edge of the currently-constructed graph. Every decision row consists of white vertices that must be assigned a color. To achieve the behavior needed for the reduction, we need to ensure that the colors assigned to the white vertices alternate between blue and non-blue (red or yellow). In order to do that, we place inverter gadgets as shown in Figure 5(a) between every pair of white vertices in a decision row. Additionally, each row-gadget contains one yellow vertex y_S and several subgadgets that we describe next.

As shown in Figure 4, the subgadgets always have a white vertex above them. Depending on the situation of that white vertex, we distinguish two different kinds of subgadgets.

If the white vertex above a subgadget is part of an uncovered southward path that was introduced for an item $x \in S$, we use the subgadget of Figure 5(b). The southward path going through the white vertex above such subgadget continues its way by either the west or east white vertex in the subgadget. The other white vertex is the first vertex of a new southward path for item x so that we say that both southward paths leaving the subgadget of Figure 5(b) from a white vertex are introduced for item x. We mark the west southward path as covered, and the east southward path as uncovered.

Otherwise, we use the subgadget of Figure 5(c). A southward path exits this subgadget marked as covered if and only if it entered this subgadget marked covered.

The construction ends with a *south gadget*, which is more or less symmetric to the north gadget, see Figure 6. The lowest vertex $v^{\max}$ is red. The color of the vertices with the two colors in Figure 6 is decided with the following rule. If such a vertex is the end of a path that is marked

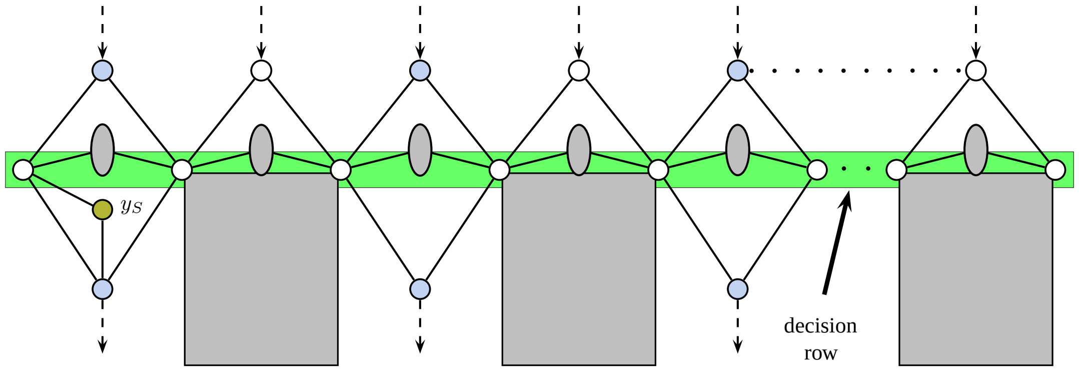


Fig. 4: Row-gadget for a set S of C. Each gray box contains a subgadget.

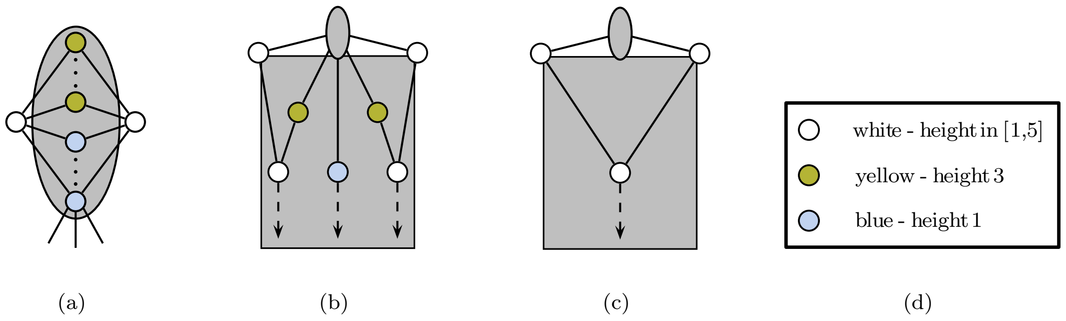


Fig. 5: (a) Inverter gadget consisting of $|U|+3$ blue and $|U|+3$ yellow vertices. (b) and (c): Subgadget part of a gadget for a set S in C. (d) The heights of the vertices in Figures (a)-(c).

uncovered, then it is colored yellow. Otherwise, it is colored red.

Let G be the graph obtained, with a straight-line embedding φ. The heights are assigned to vertices as follows. Vertices colored red have height 5, yellow vertices have height 3, blue vertices have height 1, and white vertices have a height in the range $[1, 5]$. To triangulate G, we add a vertex v_F of height 2 into each face F of φ and connect v_F to all vertices adjacent to F in φ. For

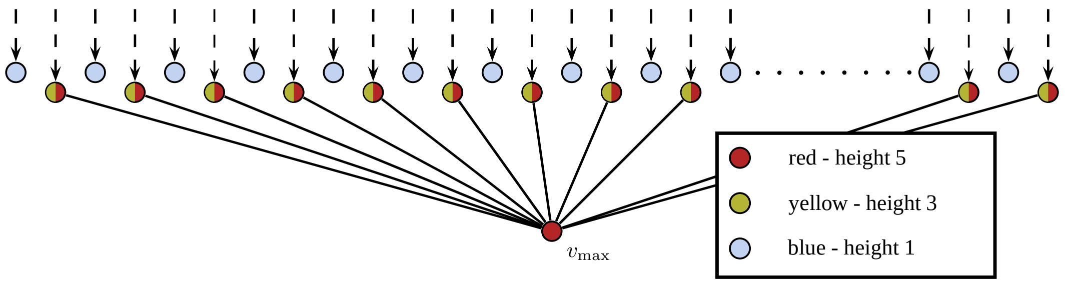


Fig. 6: South gadget. A two-colored vertex is colored either red or yellow depending on whether it is the endpoint of a path marked covered or uncovered.

the remainder of this section, let V' be the set of vertices added during the triangulation.

To see that the size of our reduction is polynomial, we must show that the number n of vertices of G is polynomial in $|C|$ and $|U|$ since this graph is planar and thus $|V'|$ is linear in n.

Lemma 3: The number of vertices in G is $n = O(|C|^2|U|^3)$.

Proof: The northernmost row contains $O(|U|^2)$ vertices, which are the starting points of $O(|U|^2)$ uncovered southward paths. Then, at every row, each uncovered path may split into a covered path and an uncovered path. Covered paths do not split further. This means that in each row we have $O(|U|^2)$ uncovered southward paths and that the total number of paths increases by at most $O(|U|^2)$ in each row, so in the last row the number of paths is at most $O(|C||U|^2)$.

Now, note that an inverter gadget consists of $O(|U|)$ vertices. Moreover, a row-gadget H for a set being crossed by $z = O(|C||U|^2)$ southward paths consists, for each southward path, of an inverter gadget and a constant number of further vertices. Thus, H has $O(z|U|) = O(|C||U|^3)$ vertices. Since the north gadget and the south gadget has fewer vertices and since we have $|C|$ row-gadgets, the total number of vertices is $n = O(|C|^2|U|^3)$, as claimed. ⊠

To minimize the number of local extrema we need to assign a height to each white vertex such that all blue vertices form one connected component and all red vertices form one connected component. Moreover, every connected components of yellow vertices needs to be connected to both a blue and a red vertex.

Theorem 2: If we allow to have the same imprecise interval for several of the n vertices of an imprecise terrain, the minimizing-extrema problem cannot be approximated within a factor of $O(\log\log n)$ in polynomial time, unless $\mathrm{P} = \mathrm{NP}$.

Proof: We first show that each solvable instance I_1 for the SET COVER problem of optimal cost $z-2$ is reduced to an instance I_2 for the minimizing-extrema problem of optimal cost z such that

- each solution for I_2 of cost y can be easily transformed into a solution for I_1 of cost $y+2$.
- each solution for I_1 of cost z can be easily transformed into a solution for I_2 of cost $z-2$.

Let $C' \subseteq C$ be a set cover. A coloring of the graph can be found as follows. We color the westernmost vertex in each decision row blue if and only if the decision row is part of a gadget for a set $S \in C'$. Moreover, we color the vertices alternating in blue and non-blue (red or yellow). We choose for a vertex v between yellow and red depending on whether v is incident (from above) to a red vertex. If so, color v red, and otherwise yellow. Color the white vertices in the gadget of Figure 5(b) by the same rule, in red or yellow. See Figures 7 and 8 for an sketch of this coloring.

In this coloring, it is easy to verify that all vertices in V' are connected to both a blue vertex and either a red or yellow vertex. Thus the vertices in V', at height 2, cannot be local extrema. Ignoring the initially yellow vertices y_S, all yellow components are connected to a blue vertex and a red vertex; this implies that yellow vertices cannot be minima or maxima. Moreover, all blue vertices form one connected component. By our choice of the height of two-colored vertices in the south gadget, the red vertices also induce a connected component since a path starting from a red vertex in the north gadget contains only red vertices and ends in the southernmost row in a covered vertex, which is by construction red. Apart from $v^{\min}$ and $v^{\max}$, the only local extrema that we have are created by the initially yellow vertices in the row-gadgets introduced for each set in C'. In other words, we have exactly $|C'|+2$ local extrema.

For the converse, let us first consider the case in which we have a solution with at least $z = |U|+2$ local extrema. Then, a set cover of size $z-2 = |U|$ exists since we only consider SET COVER-cover

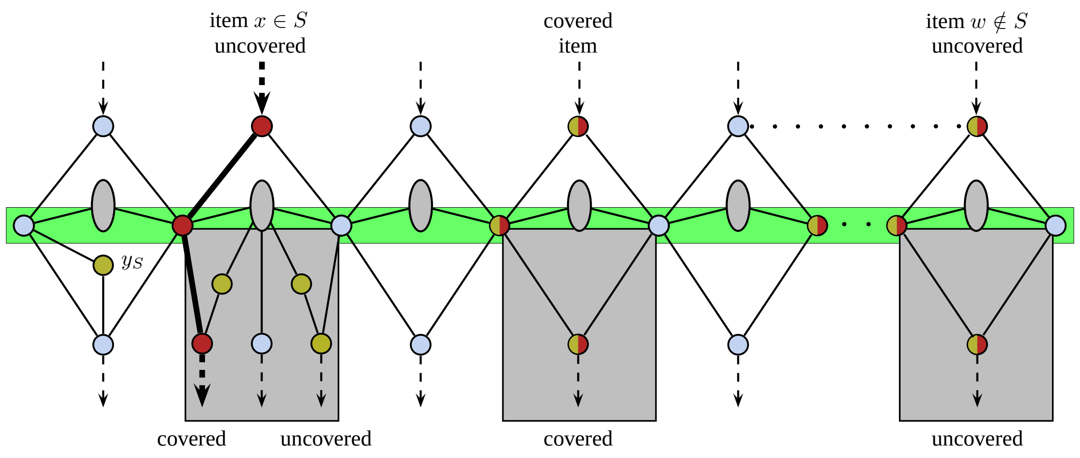


Fig. 7: A set $S \in C'$ with a local maximum at y_S allows to switch a red southward path from uncovered to covered.

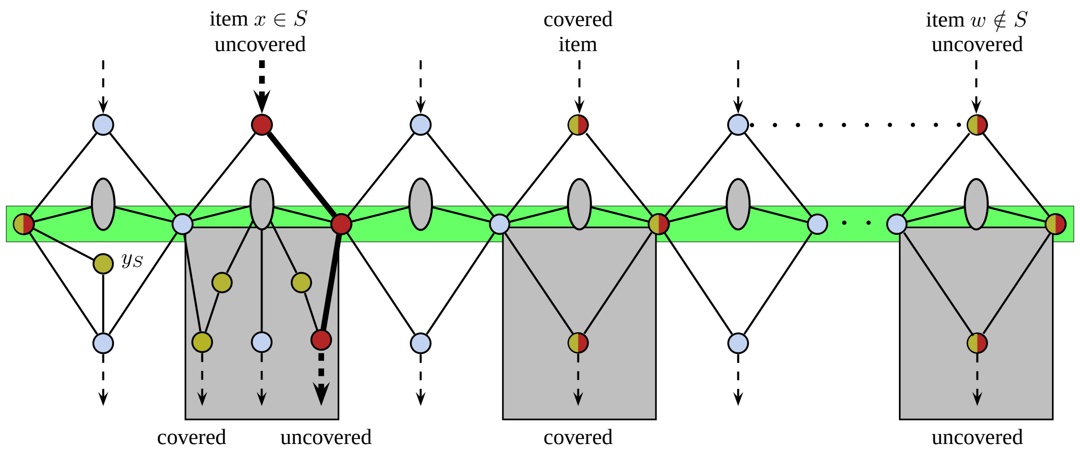


Fig. 8: A set $S \notin C'$ with no local maximum at y_S forbids to switch a red southward path from uncovered to covered.

instances with a solution. Now, let us assume that we have a solution of cost $z < |U| + 2$. Then the vertices in each decision row are alternately colored in blue and non-blue since, otherwise, the inverter gadget between two equal colored vertices has already $|U| + 3$ extrema. In addition, there is a red vertex in the north gadget introduced for each item x of the universe U that is connected by a red colored southward path to $v^{\max}$ since there are $|U| + 3$ such red vertices. Our construction then implies that there is a set S in C with $x \in S$ such that the row-gadget for S has a local maximum at its yellow vertex y_S. If we choose C' as the collection containing all sets S of C whose row-gadget contains a local maximum at its yellow vertex y_S, then C' is a cover of size at most $z - 2$.

Alon, Moshkovitz, and Safra [2] showed that, for an appropriately chosen constant $c' > 0$, there is no polynomial-time approximation algorithm of ratio $c' \ln |U|$ for the Set Cover problem with universe U unless $\mathrm{P} = \mathrm{NP}$. Since each set-cover instance with an optimal solution of cost 1 can be solved to optimality in linear time, there can not exist a polynomial-time approximation of approximation ratio $c' \ln |U|$ for the Set Cover problem restricted to instances with optimal solutions of cost at least 2 unless $\mathrm{P} = \mathrm{NP}$.

Assume for contradiction that an approximation algorithm exists for the minimizing-extrema problem in an imprecise terrain with n vertices with an approximation ratio $(c'/8) \ln \ln n$. This means, each instance of the minimizing-extrema problem of optimal cost x' can be solved with cost at most $((c'/8) \ln \ln n)x'$. By our reduction from above, each set-cover instance $I_1 = (U, C)$ with optimal cost $x \geq 2$ can be first transformed into an instance I_2 of the minimizing-extrema problem with $n \leq |C|^3|U|^2 \leq (2^{|U|})^3|U|^2 \leq 18^{|U|}$ vertices and with optimal cost $x' = x + 2 \leq 2x$. By our assumption, we can solve I_2 such that the obtained solution has cost at most $((c'/8) \ln \ln n)x' \leq ((c'/4) \ln \ln n)x$, that is, we can solve I_1 with cost at most $((c'/4) \ln \ln n)x - 2 \leq ((c'/4) \ln \ln 18^{|U|})x \leq (c' \ln |U|)x$—for the latter inequality we use the fact that the cost of an optimal solution is at least 2, i.e., $|U| \geq 2$. This is a contradiction to the last paragraph. Thus, there cannot exist an approximation algorithm for the minimizing-extrema problem in an imprecise terrain with n vertices with an approximation ratio $(c'/8) \ln \ln n$. ⊠

Note that Kumar, Arya, and Ramesh [14] showed that the Set Cover problem with universe U cannot be approximated within a factor of $o(\log |U|)$ in random polynomial time unless $\mathrm{NP} \subseteq \mathrm{ZTIME}(n^{O(\log \log n)})$ even if we restrict the collections C such that $|S_1 \cap S_2| \leq 1$ for all $S_1 \neq S_2$ in C. This restriction to C means that $|C| \leq |U|^2$. To see this, let $U = \{u_1, \ldots, u_{|U|}\}$, and consider a subcollection C_1 of C, whose subsets all contain u_1. Then it is easy to see that the number of sets in C_1 with a fixed $u' \in U \setminus \{u_1\}$ in them is at most 1. Thus, $|C_1| \leq |U|$. Applying the same argument to all subcollections $C_2, \ldots C_{|U|}$ that have $u_2, \ldots, u_n$, respectively, in them we conclude that $|C| \leq |U|^2$.

Using the reduction from above, the graph obtained for such a restricted set cover instance (U, C) has $n' = |U|^{O(1)}$ vertices; thus, we can conclude the following:

Corollary 1: If we allow to have the same imprecise interval for several of the n vertices of an imprecise terrain, the minimizing-extrema problem cannot be approximated within a factor of $o(\log n)$ in random polynomial time, unless $\mathrm{NP} \subseteq \mathrm{ZTIME}(n^{O(\log \log n)})$.

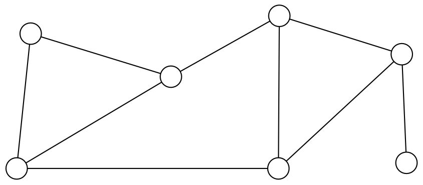
4 Height Degeneracy

We have shown that removing local minima is easy and removing local extrema is hard. However, some of our results are dependent on degeneracy issues. In this section we will describe how these issues influence the results.

There are two separate aspects to be considered. One is how we treat vertices of the same height in a realization of an imprecise terrain (that is, in a precise terrain). The other is whether we allow the tops and bottoms of the intervals in the imprecise terrain to have duplicate heights, that is, whether the input is assumed to be in general position.

4.1 Minimizing-minima is sometimes hard

In Section 2, we have shown that all local minima can be removed from an imprecise terrain in $O(n \log n)$ time. However, this result is based on the viewpoint that when a group of vertices all have the same height, we count them as a single minimum. This is a common viewpoint in the literature, and it is very reasonable from the application point of view. Nonetheless, we show here that if we count them as individual local minima, and the input is not in general position, the problem becomes hard.


(a)

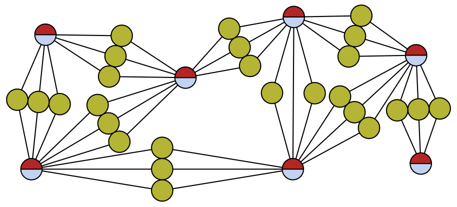

(b)

Fig. 9: (a) A planar graph (b) The white vertices are imprecise vertices at height $[1, 5]$, the edges have been replaced by groups of k yellow vertices, which are fixed at height 3.

The reduction is from the NP-hard problem to find a minimum vertex cover on planar graphs [**?**]. In this problem, a planar graph $G = (V, E)$ is given an the goal is to find a *vertex cover* of minimal size; that is a set such that each edge of G is incident to at least one vertex in the set. We build an imprecise terrain as follows. Each vertex of G is replaced by an imprecise vertex with minimum height 1 and maximum height 5. Then, each edge is replaced by $k \in \mathbb{N}$ new vertices at fixed height 3 in the middle of the edge, which are connected to both neighboring vertices. Figure 9 shows an example. Finally, we complete the triangulation by adding dummy vertices at height 5 and triangulating the resulting point set.

Having a vertex cover S in G, one can easily find a solution for the minimizing-minima problem of the same size by giving the vertices in S height 1 and all other vertices in $V - S$ height 5. For the reverse direction, let us consider a fixed solution for the minimizing-minima problem. Note that a new vertex becomes a local minimum if both neighbors are assigned a height of at least 3. If k is chosen large enough, this implies that we can only make one of the two vertices incident to an edge higher than 3. Thus, the set S of all vertices of height strictly smaller than 3 defines a vertex cover. Moreover, since each vertex with a height strictly smaller than 3 is a minimum, $|S|$ is bounded by the size of the solution for the minimizing-minima problem.

Theorem 3: The minimizing-minima (or minimizing-maxima) problem in an imprecise terrain with n vertices is NP-hard if a local minimum (resp. maximum) is defined as a vertex without any lower (resp. higher) neighbors.

Note, however, that this proof also relies on degeneracy in the input. If we assume that all input heights are different, then the edges incident to a given vertex have different heights, meaning we can put the vertex higher than some of them but lower than others. In this situation, the algorithm from Section 2 (with some small adaptations) can still be used to remove all local minima.

4.2 Minimizing-extrema is still hard when all heights are different

We can adapt the construction in Section 3.2 to use different heights at all vertices. The reason is that in the final solution of that problem, all paths of red vertices are routed south towards a single high vertex at the southern edge of the construction, while all paths of blue vertices are routed north towards a single low vertex at the northern edge of the construction. This means we can alter the construction, replacing all points (x, y, z) by a point $(x, y, z - \varepsilon y)$. If ε is small enough, this will make all points with different y-coordinates have different z-coordinates too, while not changing any property of the construction (if before two neighboring vertices had the same height, in the modified construction the southern one will be higher than the northern one). Finally, we can make the points with different x-coordinates have different heights as well by simply adding some random noise (even smaller than ε).

Theorem 4: The minimizing-extrema problem in an imprecise terrain with n vertices, for terrains in general position, cannot be approximated in polynomial time within a factor of $O(\log \log n)$, unless $\mathrm{P} = \mathrm{NP}$.

5 Relation to Splitting Graphs

In this section, we explore a relation between the problem of removing local extrema from imprecise terrains and a graph problem which we call PLANAR 2-DISJOINT MAXIMALLY CONNECTED SUBGRAPHS (or P2-MAXCON for short), which we will define shortly. In particular, this shows that removing local extrema is already NP-hard for a very restricted type of terrain: one where vertices can only be one of three types: *low*, *high*, or *unknown*, the last meaning they could be either high or low.

5.1 P2-MaxCon

This problem is a special case of 2-DISJOINT CONNECTED SUBGRAPHS (or 2-CON for short). In that problem, one is given a graph $G = (V, E)$ and two disjoint subsets $R \subset V$ and $B \subset V$ of vertices that are colored red and blue. The objective is to find two disjoint subsets $R' \supset R$ and $B' \supset B$ such that both (R', E) and (B', E) are connected graphs, that is, to color some of the remaining vertices red or blue to make both the red and the blue subgraph connected.

The 2-CON problem has received quite some attention lately. Van 't Hof *et al.* [23] showed that 2-CON is already NP-hard when there are only two red vertices. Paulusma and Van Rooij [19] try to tackle the problem by designing more efficient exact algorithms. Kammer and Tholey [11] study a related problem called the *restricted convex coloring problem* where one can color initially uncolored vertices and where one can additionally uncolor initially colored vertices. This allows, e.g., to handle corrupt data. The results in [11] are restricted to graphs of bounded treewidth. To our knowledge there are no results on the 2-CON problem for planar graphs.

We define PLANAR 2-DISJOINT CONNECTED SUBGRAPHS (or P2-CON for short) as the same problem as 2-CON, except that G is known to be planar. We also define PLANAR 2-DISJOINT MAXIMALLY CONNECTED SUBGRAPHS as the optimization variant, where the goal is to optimize the number of connected components (red and blue together) in the output graph, rather than to require that both graphs are completely connected.

The idea of the reduction to minimizing-extrema is to take an instance of P2-MAXCON, and construct from it an imprecise terrain similar to the one shown in Figure 2, by replacing the red vertices by stalactites and the blue vertices by stalagmites, and the white vertices by open space. We describe the reduction in detail in the next subsection. We then proceed to show that P2-MAXCON is NP-hard, and finally extend the proof to show that P2-CON is also NP-hard.

5.2 P2-MaxCon reduces to minimizing-extrema

We now show that the relation between the problem of removing local extrema from imprecise terrains and the graph problem P2-MAXCON implies that minimizing the number of local extrema in an imprecise terrain is NP-hard. The idea is to take an instance of P2-MAXCON, and construct from it an imprecise terrain similar to the one shown in Figure 2, by replacing the red vertices by stalactites and the blue vertices by stalagmites, and the white vertices by open space.

In our reduction, we take the input to P2-MAXCON—a planar graph with red, blue and white vertices—and build an imprecise terrain from it. We will first embed the graph in the plane with straight edges. We then turn all red vertices into precise vertices at height 5, and all blue vertices

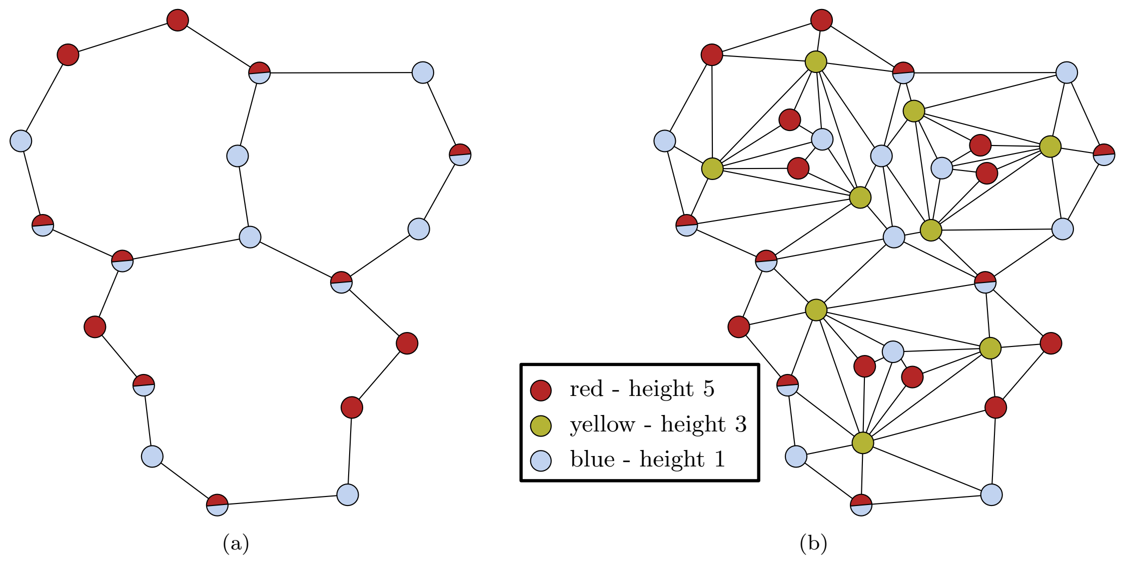


Fig. 10: (a) An instance of P2-MaxCon. (b) In the output, we fixed the red vertices at height 5, and the blue at height 1. The vertices with two colors are imprecise vertices with interval $[1, 5]$. The rest of the vertices are added to make sure that the graph is triangulated, and that the new vertices do not interfere with the number of local extrema.

into precise vertices at height 1. Finally, we turn the white vertices into imprecise vertices with interval $[1, 5]$.

The problem of minimizing extrema on this graph is equivalent to that of minimizing connected components after recoloring. This is due to the fact that the only way to remove local minima in this terrain is by connecting the minima to each other by putting white vertices at height 1. Similarly, the only way to remove maxima is to put white vertices at height 5 in order to connect the maxima to each other.

However, to have a proper imprecise terrain, we must triangulate the graph. To do this, we add extra vertices and edges as shown in Figure 10(b). This adds a component with one extra minimum and two extra maxima per inner face of the graph. In this way, all previous vertices are connected to new yellow vertices at height 3. These cannot help the red or blue vertices to stop being extrema, and at the same time cannot be extrema because they are connected to a lower and higher vertex inside the component (at heights 1 and 5). Given a solution that minimizes the number of extrema in the imprecise terrain that we have constructed, we can color the white vertices either red or blue. For any white vertex whose height is set to 1, we set the color to blue, otherwise we set the color to red.

5.3 P2-MaxCon is NP-hard

We prove that P2-MaxCon is NP-hard by a reduction from planar 3-SAT [15]. In this problem, the normal 3-SAT problem is restricted so that the bipartite graph connecting variables and clauses is planar. We call this graph $G_S = ((V \cup C), E)$, where an edge $e = (v, c) \in E$ if and only if variable v is in clause c. As usual in such reductions, we first embed G_S in the plane so that none of the edges in E cross. We then replace the vertices and edges in the embedding with “gadgets”.

The variable gadget is simply a white vertex. We show below that coloring the vertex red is equivalent to setting the corresponding variable to `true` and coloring the vertex blue is equivalent to setting the corresponding variable to `false`.

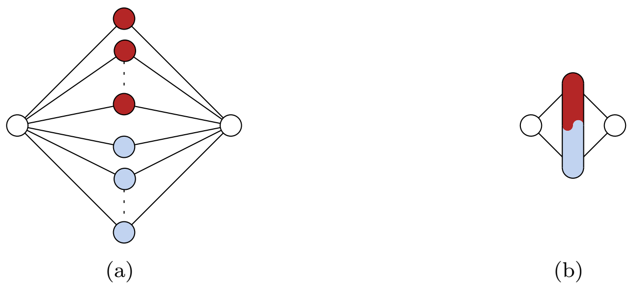


Fig. 11: (a) An inverter, consisting of k red and blue vertices. (b) Symbolic representation of an inverter.

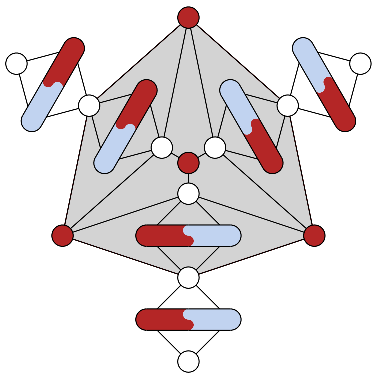


Fig. 12: A clause gadget (shown with gray background) connects three inverter gadgets using four extra red vertices.

Another gadget that we use is the inverter gadget, shown in Figure 11(a). This gadget consists of two white vertices, k red vertices, and k blue vertices. Each colored vertex is connected to both white vertices. This gadget ensures that one of the white vertices must be colored red and the other one blue, because otherwise there will be k components in the output. To ensure that this is unacceptable for any optimal solution, we make k at least as large as the number of gadgets that we use in our construction.

A clause gadget is a collection of three inverter gadgets, as well as four extra red vertices. These are all connected as shown in Figure 12. The red vertices form one large component as long as at least one of the white vertices adjacent to the central red vertex is colored red.

Finally, we create edge gadgets to connect variable gadgets to clause gadgets. An edge gadget is simply a chain of inverter gadgets. See Figure 13. If a variable v is negated in clause c, then we replace the edge (v, c) with a chain of an odd number of inverter gadgets, otherwise, we use an even-length chain. Since the number of inverter gadgets between a variable gadget and one of the clause gadgets that it is connected to determines the color of the final white vertex in the chain, we can see that coloring a variable gadget red corresponds to the final white vertex in a chain to a clause in which that vertex is not negated being colored red. This implies that coloring a variable gadget red is equivalent to setting its value to `true`, and that coloring a variable gadget blue is equivalent to setting its value to `false`.

The total number of connected components is equal to the number of white vertices in the construction, minus 2 per clause since the red components are connected, plus the number of unsatisfied clauses. Hence, minimizing the number of connected components involves determining whether the 3-SAT clause can be satisfied, which proves the following.

Theorem 5: P2-MaxCon is NP-hard.

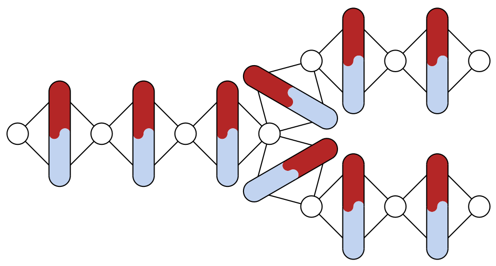


Fig. 13: We can chain inverter gadgets. The white vertices must always be colored alternately red and blue.

5.4 P2-Con is NP-hard

We now show how the construction above can be extended to show that also the more specialized problem P2-Con is NP-hard. The main difference is that we must ensure that at the end of the construction, all blue components become connected into one large blue component, and all red components become connected into one large red component.

First of all, we need some property of triangulated graphs. Let $G = (V, E)$ be a *planar triangulated graph*, i.e., a graph embedded in the plane such that all faces of G, except the outer face, are triangles. Moreover, let us color the vertices of G red and blue such that the red vertices R on the outer face of G form more than one single connected component and the same is true for the blue vertices B. We say that the subgraph of G induced by R contains a *proper loop* if it contains a cycle that has at least one vertex of $V \setminus R$ inside. This leads to the following well-known observation. See, for example, the book by West [24] for a proof.

Observation 1: If the subgraph of G induced by R has no proper loops, then the subgraph of G induced by $V \setminus R$ is connected.

Now, let G again be a planar triangulated graph, and suppose that all vertices of G are colored either red or blue, so $V = R \cup B$, and suppose further that every red or blue component has at least one vertex on the outer face. Figure 14(a) shows such a graph. Then obviously neither R nor B has a proper loop.

This means that whenever we have such a graph with a sufficient number of layers of white vertices around it, then we can color it such that we get only one large red component and only one large blue component.

Lemma 4: Let G be as before, and let G' be a larger graph that contains G and has 2 extra layers of white vertices around G, each at least as large as the outer face of G. Then we can color the white vertices of G' red or blue such that G' has only one red and one blue component.

Proof: We know that all components have at least one vertex on the outer face. For each red component, we select exactly one such vertex and color its counterpieces on the two extra layers also red. Then we color the vertices of the outer layer red to connect all the red components into one large component, as shown in Figure 14(b).

By doing this, we cannot create any proper loops because we only took one vertex from each red component, and because of the regular structure of the two outer layers. Therefore, the red component does not have proper loops, so by Observation 1 the complement is connected. Hence, we can color the complement blue to obtain a valid coloring. $\boxtimes$

We now show how the construction in the previous section can be extended to show that not only P2-MaxCon, but also P2-Con is NP-hard. To do this, we must make a construction such that, when the SAT formula is satisfiable, all red components can be connected into one large red

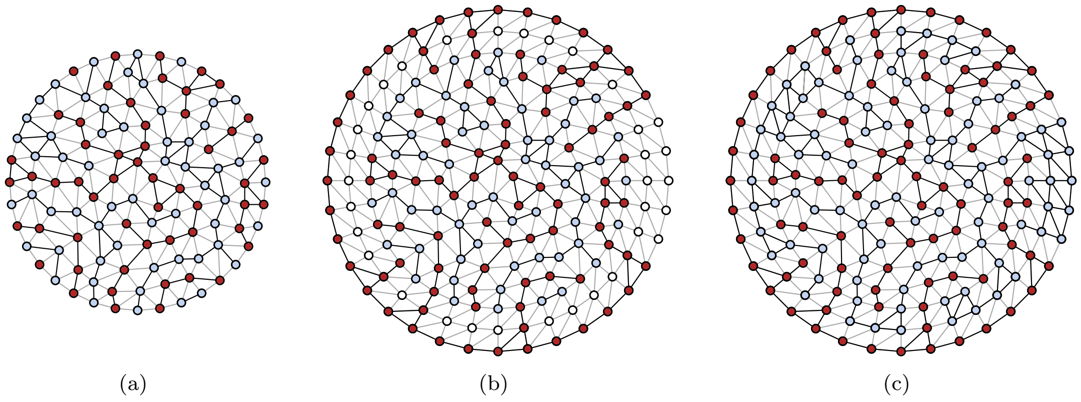


Fig. 14: (a) A triangulated graph, colored such that every component has a vertex on the outer face. (b) The same graph, augmented with two extra layers of white vertices. The red components are connected into a large component without proper loops. (d) All the remaining white vertices can be colored blue, making the blue component also connected.

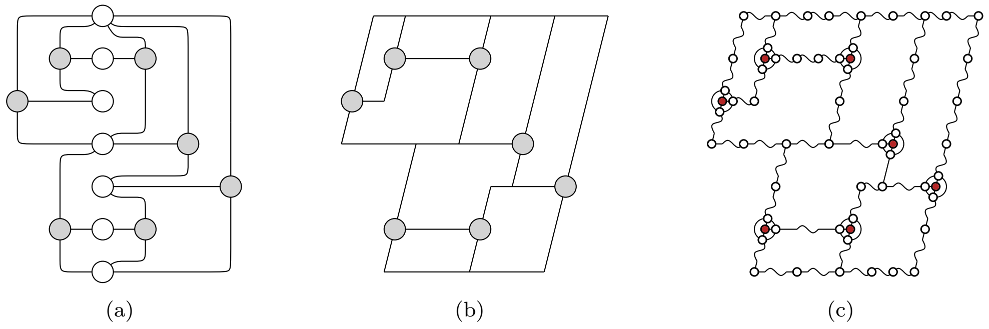


Fig. 15: (a) A layout of a planar 3-SAT instance, where the variable nodes (white) are aligned on a single vertical line, and the clause nodes (gray) are on both sides of the line. (b) Polygonal subdivision into x-monotone polygons as a result of replacing the variable nodes and adjacent edges by polygonal trees. (c) The construction embedded onto the subdivision. The inverter gadgets are indicated by wiggling edges. Note that there is some freedom in the construction as to how many vertices and inverter gadgets are placed on the edges; only the parity matters.

component, and all the blue components can be connected into one large blue component. If the formula is not satisfiable, this should not be possible.

It has been showed by Knuth and Raghunathan [13] that a planar 3-SAT graph can be embedded in the plane having all variable nodes on one vertical line, and the clause nodes on both sides, as in Figure 15(a). From this, we replace the variable nodes and incident edges by polygonal trees, and we obtain an embedding, as shown in Figure 15(b) for the example graph. If any faces of the embedding are not x-monotone polygons, we subdivide them by vertical line segments (except for the outer face). This implies a partial ordering on the faces of the graph, based on their above-below relation.[2]

We now replace the polygonal trees by chains of inverter gadgets, as in the previous section. We also replace the clause nodes by clause gadgets as before, except that we do not include the three extra red vertices. Instead, we connect the neighboring vertices of each clause. Figure 16 shows

[2] In fact we don't really need them to be x-monotone polygons, any embedding with a directed dual graph that induces a partial ordering such that all faces at the north of the ordering are on the outside of the construction would be good enough for the argument.

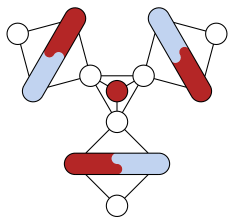


Fig. 16: A clause gadget, simplified.

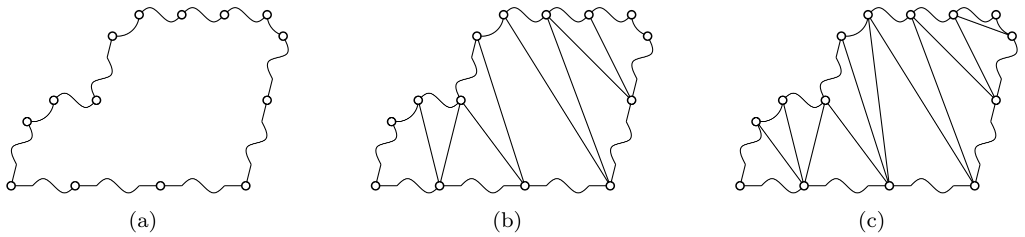


Fig. 17: (a) An x-monotone polygon with some white vertices and inverter gadgets on its boundary. (b) Each white vertex on the south has been connected to two white vertices at the north that are separated by an inverter gadget. (c) Some additional edges are inserted to make the graph formed by the white vertices triangulated.

the simplified clause gadget. Figure 15(c) shows an example of the resulting embedding.

The resulting construction has many white vertices on the boundaries of the x-monotone polygons, which end up being colored either red or blue. Furthermore, all components in a final coloring contain at least one of these white vertices, except for those in the unsatisfied clauses. So, what remains to be done is make sure that these white vertices can be connected into one large red and one large blue component, no matter how they are colored.

To do this, consider one x-monotone polygon and its white vertices, as in Figure 17(a). An x-monotone polygon has two x-monotone polygonal chains that both connect the westernmost point to the easternmost point. We assume that on both of these chains there are at least 2 white vertices. Furthermore, we assume that the northern chain of any polygon has at least as many white vertices as the southern chain. If any of these assumptions is not satisfied, we simply add more gadgets to the chains: adding two inverter gadgets into a chain does not change the properties of the construction, and the above/below relations of the polygons define a partial order on them so this process will end. We then add edges to the interior of the polygon, connecting every white vertex on the southern chain to two adjacent white vertices on the northern chain that are on both sides of an inverter gadget, as in Figure 17(b). This means that whatever the color of a vertex of the southern chain is, it is always connected to at least one vertex of the same color on the northern chain. By induction, this means that every white vertex in the whole construction will be connected to some vertex on the north of the construction that has the same color. Finally, we triangulate the polygon (or rather, the graph of the white vertices involved in the polygon) by adding arbitrary edges if necessary, see Figure 17(c). We call the resulting graph G.

To prove that the construction is indeed colorable with two components if the 3-SAT formula is satisfiable, first consider the coloring that makes all clause gadgets satisfied. Then remove the clause vertices (leaving only the three white vertices and the triangle of edges connecting them), and replace all inverter gadgets by edges. The resulting graph is triangulated, and by the above argument, each component contains at least one vertex on the outside. Conversely, if the 3-SAT formula is not satisfiable, it is not possible to color the construction with two colors such that the vertices with equal colors form two connected components because one of the clauses cannot be satisfied. This implies that at least one of the clause gadgets cannot be properly colored. Finally, by Lemma 4, there also exists a corresponding graph G' that can be colored in two *connected* colors if and only if the 3-SAT formula is satisfyable. We conclude:

Theorem 6: P2-CON is NP-hard.

6 Discussion

We have studied the complexity of removing local extrema from imprecise terrains. We conclude that this complexity changes dramatically between the problem of removing only one kind of extrema (either local minima or local maxima) and the problem of removing both at the same time. When one is interested in removing only either local minima or local maxima, this problem can be solved efficiently in $O(n \log n)$ time. This problem has real applications in, for example, hydrology, and we believe our solution is both simple and practical. On the other hand, removing local extrema is hard to approximate, even within a factor of $O(\log \log n)$.

In addition, we show that even a simplified version of the problem, where precise vertices have only three possible heights, is already NP-hard. This hardness proof exploits a relation to PLANAR 2-DISJOINT CONNECTED SUBGRAPHS, which we also prove is NP-hard. To the best of our knowledge, this constitutes the first result for planar graphs for the popular 2-DISJOINT CONNECTED SUBGRAPHS problem.

We believe the main remaining open question is whether any constructive results for the problem of minimizing local extrema are possible. Though our hardness result shows there is no hope for even an approximation algorithm with a practical approximation factor in the general case, it is still possible that something better can be done for special classes of imprecise terrains. Moreover, Equation 1 suggests that in real terrains, it may be much easier to remove local extrema than what our theoretical results suggest. It would be interesting to run experiments on real terrains in order to analyze the size of the gap left by Equation 1 in practice, and to investigate if there is some set of *realistic* conditions on the input terrain for which the gap is small.

Acknowledgments

We thank Jeff Phillips for proposing the problem in Section 3. We thank the anonymous reviewers for many useful suggestions that improved the paper in many aspects. C.G. is funded by the German Ministry for Education and Research (BMBF) under grant number 03NAPI4 “ADVEST”. M.L. is partially funded by the U.S. Office of Naval Research under grant N00014-08-1-1015, and partially supported by the Netherlands Organisation for Scientific Research (NWO) under grant 639.021.123. R.I.S. is supported by the Netherlands Organisation for Scientific Research (NWO).

References

[1] P. K. Agarwal, L. Arge, and K. Yi. I/o-efficient batched union-find and its applications to terrain analysis. *ACM Transactions on Algorithms*, 7(1):11, 2010.

[2] N. Alon, D. Moshkovitz, and S. Safra. Algorithmic construction of sets for k-restrictions. *ACM Transactions on Algorithms*, pages 153–177, 2006.

[3] R. Carlson and A. Danner. Bridge detection in grid terrains and improved drainage enforcement. In *Proc. ACM Symposium on Advances in Geographic Information Systems*, pages 250–260, 2010.

[4] J. Charleux-Demargne and C. Puech. Quality assessment for drainage networks and watershed boundaries extraction from a digital elevation model (DEM). In *Proc. 8th Internat. Symp. Advances Geographic Information Systems (ACM-GIS)*, pages 89–94, New York, 2000. ACM Press.

[5] T. de Kok, M. van Kreveld, and M. Löffler. Generating realistic terrains with higher-order Delaunay triangulations. *Comput. Geom. Theory Appl.*, 36:52–65, 2007.

[6] P. F. Fisher and N. J. Tate. Causes and consequences of error in digital elevation models. *Progress in Physical Geography*, 30(4):467–489, 2006.

[7] J. C. Gallant and J. P. Wilson. *Terrain Analysis: Principles and Applications.* John Wiley & Sons, Inc., 1998.

[8] C. Gray and W. Evans. Optimistic shortest paths on uncertain terrains. In *Proc. 16th Canadian Conference on Comput. Geom.*, pages 68–71, 2004.

[9] C. Gray, M. Löffler, and R. I. Silveira. Smoothing imprecise 1.5D terrains. *Int. J. Comput. Geometry Appl.*, 20(4):381–414, 2010.

[10] J. Gudmundsson, M. Hammar, and M. van Kreveld. Higher order Delaunay triangulations. *Comput. Geom. Theory Appl.*, 23:85–98, 2002.

[11] F. Kammer and T. Tholey. The complexity of minimum convex coloring. In *Proc. 19th International Symposium on Algorithms and Computation*, pages 16–27, 2008.

[12] Y. Kholondyrev and W. Evans. Optimistic and pessimistic shortest paths on uncertain terrains. In *Proc. 19th Canadian Conference on Comput. Geom.*, pages 197–200, 2007.

[13] D. E. Knuth and A. Raghunathan. The problem of compatible representatives. *SIAM J. Discrete Math.*, 5(3):422–427, 1992.

[14] V. S. A. Kumar, S. Arya, and H. Ramesh. Hardness of set cover with intersection 1. In *ICALP*, pages 624–635, 2000.

[15] D. Lichtenstein. Planar formulae and their uses. *SIAM J. Comput.*, 11(2):329–343, 1982.

[16] Y. Liu and J. Snoeyink. Flooding triangulated terrain. In P. Fisher, editor, *Developments in Spatial Data Handling, proceedings of the 11th Internat. Sympos.*, pages 137–148, Berlin, 2004.

[17] D. Mark. Network models in geomorphology. In M. G. Anderson, editor, *Modelling Geomorphological Systems*, chapter 4, pages 73–97. John Wiley & Sons, 1988.

[18] L. W. Martz and J. Garbrecht. An outlet breaching algorithm for the treatment of closed depressions in a raster DEM. *Computers & Geosciences*, 25:835–844, 1999.

[19] D. Paulusma and J. van Rooij. On partitioning a graph into two connected subgraphs. In *Proc. 20th International Symposium on Algorithms and Computation*, pages 1215–1224, 2009.

[20] W. Rieger. A phenomenon-based approach to upslope contributing area and depressions in DEMs. *Hydrological Processes*, 12:857–872, 1998.

[21] R. I. Silveira and R. van Oostrum. Flooding countries and destroying dams. *Int. J. Comput. Geom. Appl.*, 20(3):361–380, 2010.

[22] A. Temme, J. Schoorl, and A. Veldkamp. Algorithm for dealing with depressions in dynamic landscape evolution models. *Computers & Geosciences*, 32:452–461, 2006.

[23] P. van 't Hof, D. Paulusma, and G. Woeginger. Partitioning graphs in connected parts. *Theoretical Computer Science*, 410:4834–4843, 2009.

[24] D. B. West. *Introduction to Graph Theory.* Prentice Hall, 2nd edition, September 2000.

[25] Q. Zhu, Y. Tian, and J. Zhao. An efficient depression processing algorithm for hydrologic analysis. *Computers & Geosciences*, 32:615–623, 2006.