# Flow Computations on Imprecise Terrains

Anne Driemel* Herman Haverkort† Maarten Löffler‡ Rodrigo I. Silveira§

**Abstract**

We study the computation of the flow of water on imprecise terrains. We consider two approaches to modeling flow on a terrain: one where water can only flow along the edges of a predefined graph (for example a grid, a triangulation, or its dual), possibly non-planar, and one where water flows across the surface of a polyhedral terrain in the direction of steepest descent. In both cases each vertex has an imprecise height, given by an interval of possible values, while its $(x, y)$-coordinates are fixed. For the first model, we give a simple $O(n \log n)$ time algorithm to compute the maximal watershed of a vertex, where $n$ is the number of edges of the graph. We show that, in contrast, in the second model the problem of deciding whether one vertex may be contained in the watershed of another is NP-hard.

## 1 Introduction

Simulating the flow of water on a terrain is a problem that has long been studied in geographic information science (GIS). It is common practice to derive drainage networks, channel lines, and catchment areas directly from a digital elevation model.

Naturally, these computations are affected by measurement errors of the elevations. A frequent way to deal with this imprecision is to model the elevation at a point of the terrain using stochastic methods [3, 7], leading to results that are not fully reliable. An approach taken in computational geometry [1, 2, 4] is to replace the exact elevation of each surface point by an imprecision interval, and computing the outcome of the most optimistic and pessimistic scenarios exactly. This is the approach we take in this paper.

When simulating water flow on terrain surfaces, it is assumed that water flows downward, in the direction of steepest descent. Most hydrological research in GIS uses a grid elevation model, in which each grid cell can drain to one or more of its eight neighbors, such as in the D-8 model [5]. For a discussion about the most common flow direction models see Tarboton [6]. When the surface is represented by a polyhedral terrain, the flow of water can be traced across the surface of a triangle, as discussed by Yu et al. [8].

**Definitions.** We define an ***imprecise terrain*** $T$ as a possibly non-planar geometric graph in $\mathbb{R}^2$ in which each vertex $v \in \mathbb{R}^2$ has an imprecise third coordinate, which represents its ***elevation***. We denote the bounds of the elevation of $v$ with $low(v)$ and $high(v)$. A ***realization*** $R$ of an imprecise terrain $T$ consists of the given graph together with an assignment of elevations to vertices such that for each vertex $v$ its elevation $elev_R(v)$ is at least $low(v)$ and at most $high(v)$. We denote the set of all realizations of an imprecise terrain $T$ with $\mathcal{R}_T$. We study the flow of water on imprecise terrains in two different models.

In the ***network model*** we assume that water only flows along the edges of the realization. The steepness of descent along an edge $(p, q)$ in a realization $R$ is defined as $\sigma_R(p, q) = (elev_R(p) - elev_R(q))/|pq|$. The water that arrives at a particular vertex $p$, flows to the neighbor $q$, such that $\sigma_R(p, q)$ is positive and maximal over all edges incident to $p$. For simplicity of exposition, we assume that this steepest descent neighbor is always unique and that edges are never horizontal in the realizations considered. If water flows from $p$ to $q$ in a realization $R$ we write $p \underset{R}{\rightarrow} q$. If a vertex does not have a lower neighbor we call it a ***local minimum***.

The ***watershed*** of a vertex $q$ in a realization $R$ is defined as the set $\mathcal{W}(R, q) = \{p : p \underset{R}{\rightarrow} q\}$. The ***potential watershed*** of a vertex $q$ in a terrain $T$ is $\mathcal{W}_\cup(q) = \bigcup_{R \in \mathcal{R}_T} \mathcal{W}(R, q)$, that is, it is the set of points $p$ for which there exists a realization $R$, such that water flows from $p$ to $q$.

If the graph in the $(x, y)$-domain is a planar triangulation, then we can also consider the case that water flows across the polyhedral terrain represented by a realization. We call this the ***surface model***. The water that arrives at a particular point on this surface now flows in the true direction of the steepest descent, possibly across the interior of a triangle.

**Results.** In Section 2 we give a simple $O(n \log n)$ time algorithm to compute the potential watershed of a vertex in the network model, where $n$ is the number of edges of the graph. The analysis also shows that for every vertex $p$, there is a realization of the terrain

*Utrecht University, The Netherlands, anne@cs.uu.nl. This work has been supported by the Netherlands Organisation for Scientific Research (NWO) under RIMGA.

†Dept. of Computer Sc., TU Eindhoven, the Netherlands

‡Computer Science Department, University of California, Irvine, USA, mloffler@uci.edu. Funded by the U.S. Office of Naval Research under grant N00014-08-1-1015.

§Departament de Matemàtica Aplicada II, Universitat Politècnica de Catalunya, Spain, rodrigo.silveira@upc.edu. Supported by NWO.

in which the watershed of $p$ is its complete potential watershed, i.e., $\mathcal{W}_{\cup}(p)$ is realizable. In contrast, we show in Section 3 that in the surface model the problem of deciding whether water can possibly flow from a given point $p$ to a given point $q$ is NP-hard.

## 2 Computing watersheds in the network model

**Canonical realization.** We first show that the potential watershed of a vertex $q$ is realizable.

**Definition 1** *The* **watershed-overlay** *of a set of watersheds $\mathcal{W}(R_1, q_1), ..., \mathcal{W}(R_k, q_k)$ is the realization $R^*$ such that for every vertex $v$, we have that $elev_{R^*}(v) = high(v)$ if $v \notin \bigcup \mathcal{W}(R_i, q_i)$ and $elev_{R^*}(v) = \min_{i: v \in \mathcal{W}(R_i, q_i)} elev_{R_i}(v)$ otherwise.*

**Lemma 1** *Let $R^*$ be the watershed-overlay of $\mathcal{W}(R_1, q), \ldots, \mathcal{W}(R_k, q)$, then it holds that $\mathcal{W}(R^*, q)$ contains $\mathcal{W}(R_i, q)$, for any $i \in \{1, \ldots, k\}$.*

**Proof.** Let $u$ be a vertex of the terrain, which is contained in one of the given watersheds. Let $R_i$ be a realization from $R_1, ..., R_k$ such that $elev_{R^*}(u) = elev_{R_i}(u)$. To prove the lemma, we show that $u$ is contained in $\mathcal{W}(R^*, q)$ by induction on increasing elevation of $u$ in $R^*$. The base case is that $u$ is equal to $q$, and in this case the claim holds trivially.

Now, consider the vertex $v$ which is reached from $u$ by taking the steepest descent edge in $R_i$. Since $elev_{R^*}(v) \leq elev_{R_i}(v) \leq elev_{R_i}(u) = elev_{R^*}(u)$, it holds that $v$ lies lower than $u$ in $R^*$.

If $v$ is still the steepest descent neighbor of $u$ in $R^*$, then, by induction, $v \in \mathcal{W}(R^*, q)$ and therefore $u \in \mathcal{W}(R^*, q)$. Otherwise, there is a vertex $\widehat{v}$ such that $\sigma_{R^*}(u, \widehat{v}) > \sigma_{R^*}(u, v)$. There must be an $R_j$ such that $\widehat{v} \in \mathcal{W}(R_j, q)$, since otherwise, by construction of the watershed-overlay, we have $elev_{R^*}(\widehat{v}) = high(\widehat{v}) \geq elev_{R_i}(\widehat{v})$ and thus, $\sigma_{R_i}(u, \widehat{v}) \geq \sigma_{R^*}(u, \widehat{v}) > \sigma_{R^*}(u, v) \geq \sigma_{R_i}(u, v)$ and $v$ would not be the steepest descent neighbor of $u$ in $R_i$. Therefore, by induction, also $\widehat{v} \in \mathcal{W}(R^*, q)$ and, again, $u \in \mathcal{W}(R^*, q)$. □

The above lemma implies that for any vertex $q$, the watershed-overlay $R^*$ of all possible realizations in $\mathcal{R}_T$, realizes the potential watershed of $q$, i.e., $\mathcal{W}_{\cup}(q) = \mathcal{W}(R^*, q)$. Therefore, we call $R^*$ the ***canonical realization*** of the potential watershed $\mathcal{W}_{\cup}(q)$.

**Algorithm.** Next, we describe how to compute the canonical realization of $\mathcal{W}_{\cup}(q)$ for a given vertex $q$. The idea of the algorithm is to compute the vertices of $\mathcal{W}_{\cup}(q)$ and their canonical elevations in increasing order of elevation, similar to the way in which Dijkstra's shortest path algorithm computes distances from the source. Refer to Algorithm 1 for the general outline.

The algorithm uses a subroutine EXPAND$(q', z')$, which returns for a vertex $q'$ and an elevation $z' \in [low(q'), high(q')]$ the set of neighbors $P$ of $q'$, such that for each $p \in P$, there exists a realization $R$ with $elev_R(q') \in [z', high(q')]$, such that $p \xrightarrow[R]{} q'$. In particular, it returns tuples of the form $(p, z)$, where $z$ is the minimum elevation of $p$ over all such realizations $R$. We will now explain the preprocessing step that allows an efficient computation of EXPAND$(q', z')$.

**Algorithm 1** COMPUTEPWS$(q)$

```
1: Enqueue $(q, z)$ with key $z = low(q)$
2: while the Queue is not empty do
3:   $(q', z')$ = DequeueMin()
4:   if $q'$ is not already in the output set then
5:     Output $q'$ and set $elev_{R^*}(q') = z'$
6:     Enqueue each $(p, z) \in$ EXPAND$(q', z')$
7:   end if
8: end while
```

We define the ***slope diagram*** of a vertex $p$ as the set of points $\widehat{q_i} = (d_i, high(q_i))$, such that $q_i$ is a neighbor of $p$ and $d_i$ is its distance to $p$ in the $(x, y)$-projection. Let $q_1, q_2, ...,$ be the neighbors of $p$ indexed such that $\widehat{q}_1, \widehat{q}_2, ...$ appear in counter-clockwise order along the boundary of the convex hull in the slope diagram, starting from the leftmost point and continuing to the lowest point (for simplicity of exposition we ignore the remaining neighbors). Let $z_i$ be the value where the supporting line of the edge $\overline{\widehat{q}_i, \widehat{q}_{i+1}}$ intersects the vertical axis of the slope diagram, see Figure 1. We denote with $Z(p)$ the resulting decomposition of $[low(p), high(p)]$ into intervals that is given by the $z_i$ and annotated by the corresponding points $q_i$. We can precompute $Z(p)$ in time $O(d \log d)$ and space $O(d)$, where $d$ is the vertex degree of $p$. Since the sum of vertex degrees is $O(n)$, this takes $O(n \log d_{\max})$ time and $O(n)$ space overall, where $d_{\max}$ is the maximum vertex degree in the terrain.

Now, for a neighbor $p$ of $q'$, we can compute its elevation as it should be returned by EXPAND$(q', z')$ by computing the lower tangent to the convex hull in the slope diagram, which passes through the point $\widehat{q'} = (d', z')$, where $d'$ is the distance to $p$ in the $(x, y)$-

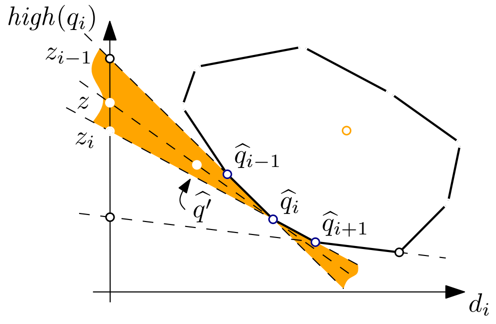


Figure 1: Slope diagram of the neighbors of $p$.

projection, see Figure 1. This can be done via a binary search on $Z(p)$ in time in $O(\log d_{\max})$. Intuitively, we annotated each interval of $Z(p)$, with the neighbor of $p$ that the vertex $q'$ has to compete with for being the steepest-descent neighbor if the elevation of $p$ is in this interval. Doing this for all neighbors of $q'$, we find that $\textsc{Expand}(q', z')$ runs in $O(d \log d_{\max})$ time, where $d$ is the vertex degree of $q'$. We get the following lemma. We omit the full proof due to space limitations.

**Lemma 2** *After precomputations in $O(n \log d_{\max})$ time and $O(n)$ space, $\textsc{Expand}(q, z)$ can be implemented to run in $O(d \log d_{\max})$ time.*

To prove the correctness of Algorithm 1 we use induction on the vertices extracted from the priority queue in the order of their extraction. The induction hypothesis says that for each extracted tuple $(q', z')$, it holds that there exists a realization $R$ with $elev_R(q') = z'$ and $q' \xrightarrow[R]{} q$, such that this elevation is minimal. By Lemma 1, this implies that the algorithm outputs the canonical realization of $\mathcal{W}_\cup(q)$. As for the running time, it is easy to see that each vertex is expanded at most once. Using Lemma 2, we get the following theorem. We omit the full proof due to space limitations.

**Theorem 3** *The algorithm $\textsc{ComputePWS}(q)$ computes the canonical realization of the potential watershed $\mathcal{W}_\cup(q)$ in time $O(n \log n)$, where $n$ is the number of edges of the graph.*

## 3 NP-hardness in the surface model

In this section we sketch how to prove that deciding whether water potentially flows from a point $s$ to another point $t$ on an imprecise triangulated terrain, under the surface model, is NP-hard. The reduction is from 3-SAT; the input is an instance with $n$ variables and $m$ clauses. Globally, the construction consists of a grid with $O(m) \times O(n)$ squares, where each clause corresponds to a column and each variable to a row of the grid; the construction also contains some columns and rows that do not directly correspond to clauses and variables. The grid is placed across the slope of a “mountain” with shape similar to that of a pyramidal frustum. Figure 2 illustrates the construction. (The vertical faces in the illustration can easily be replaced by non-vertical triangulated slopes without affecting the construction.) A key element in the construction is the *divider* gadget (Figure 3, left), which is placed at every intersection of a clause column and a variable row. It consists of two imprecise vertices with a long edge between them and ensures that only if the two imprecise vertices are both at opposite extreme heights, can any water pass the divider gadget, otherwise it will flow to a local minimum. This way it carries over the (inverted) state of each imprecise vertex from left to right.

Across each divider gadget, water may flow in several courses, that may each veer off to the left or to the right, depending on the elevations of the imprecise vertices. We assume that the water takes the left courses if the variable is true, and the right courses if it is false. Now, to encode each clause, we let the water flow to a local minimum if and only if the clause is not satisfied. Figure 2 (right) shows an example.

In order to link the values of the imprecise vertices of the heights in the divider gadgets that belong to the same variable, we need to make sure that neighboring vertices have opposite extremal heights, just like in divider gadgets. For this, we use a *connector gadget*, which is basically the same construction as the divider gadget, see Figure 3 (right). As in the divider gadget, we only let the water escape if the heights of the imprecise vertices are at opposite extremes.

With this construction water can flow from $s$ to $t$ if and only if the 3-SAT formula can be satisfied.

## 4 Further Work

There are many related problems in the network model that cannot be discussed due to space limitations. We want to mention at least some of them.

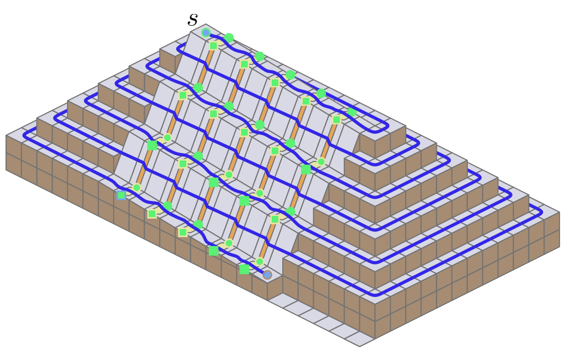
Similar to the potential watershed of $q$, we can define the set of points that potentially *receive* water from $q$. Naturally, there is not always a canonical realization for this set, however, it can be computed in the same way as described in Section 2 using a priority queue that processes vertices in decreasing order of their maximal elevation, such that they would still receive water from $q$. Our definitions and algorithms naturally extend to sets of nodes. Therefore, we can also compute potential watersheds with respect to lakes or river beds.

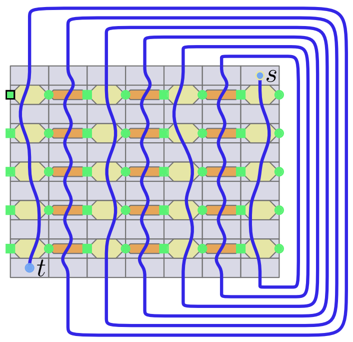
Secondly, besides the potential flow paths, one may be interested in the question of whether water *always* flows between two vertices. We can define the set

$$\mathcal{W}_\cap(q) = \bigcap_{R \in \mathcal{R}_T} \mathcal{W}(R, q),$$

which is the set of points from which water flows to $q$ in any realization. Observe that a vertex is contained in $\mathcal{W}_\cap(q)$ if and only if it does not have a potential flow path to a potential local minimum or a point outside $\mathcal{W}_\cup(q)$, which does not go through $q$. We can compute this set using the techniques described here. However, this definition may be very sensitive to flow paths being interrupted by overlapping imprecision intervals of neighboring vertices in relative flat terrain. Therefore, it is not clear if this is the right definition of persistent water flow.

Thirdly, note that for two given vertices $p$ and $q$, such that $p \in \mathcal{W}_\cup(q)$, it is not necessarily true





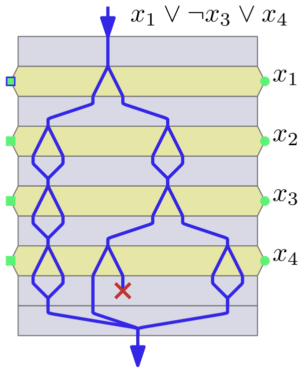
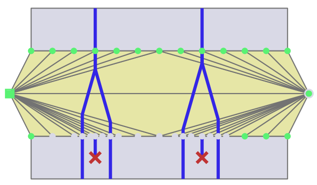


Figure 2: Left: Global view of the construction, showing the grid on the mountain slope. The fixed parts are shown in gray, the variable parts are shown yellow (for divider gadgets) and orange (for connector gadgets). Center: Top down view showing the locations of the gadgets, and the $n \times 2m$ green vertices, the only ones with imprecise heights. Right: Detail of a clause, which forms one of the columns of the grid in the center.



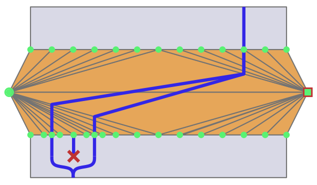


Figure 3: Left: A divider gadget consists of two imprecise vertices with an edge between them. Right: A connector gadget. The triangle needs to be much narrower, and the water streams need to be much closer to the center of the construction than in the picture.

that $\mathcal{W}_{\cup}(p) \subseteq \mathcal{W}_{\cup}(q)$. Therefore the potential watersheds of a terrain do not form a proper hierarchy. This makes it challenging to design a data structure that stores imprecise watersheds and answers queries about the flow of water between vertices efficiently.

Finally, the contrast between the results in Section 2 and Section 3 leaves room for further research questions, i.e., would it be possible to apply realistic input assumptions to make the potential flow computation in the surface model tractable? Another question is whether there exists a model of approximation for water flow and how it relates to the network model.

**Acknowledgments.** We are grateful to Chris Gray for many interesting and useful discussions.

## References

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

[2] C. Gray, M. Löffler, and R. I. Silveira. Smoothing imprecise 1.5D terrains. In *Proc. 6th International Workshop on Approximation and Online Algorithms*, pages 214–226, 2009.

[3] F. Hebeler and R. Purves. The influence of elevation uncertainty on derivation of topographic indices. *Geomorphology*, 111(1-2):4 – 16, 2009.

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

[5] J. O'Callaghan and D. Mark. The extraction of drainage networks from digital elevation data. *Computer vision, graphics, and image processing*, 28(3):323–344, 1984.

[6] D. Tarboton. A new method for the determination of flow directions and upslope areas in grid digital elevation models. *Water Resources Research*, 33(2):309–319, 1997.

[7] S. P. Wechsler. Uncertainties associated with digital elevation models for hydrologic applications: a review. *Hydrology and Earth System Sciences*, 11(4):1481–1500, 2007.

[8] S. Yu, M. van Kreveld, and J. Snoeyink. Drainage queries in TINs: from local to global and back again. In *Proc. 7th Int. Symp. on Spatial Data Handling*, pages 13A.1–13A.14, 1996.