Flood Risk Analysis on Terrains

Mathias Rav
MADALGO, Aarhus University

Aaron Lowe
Duke University

Pankaj K. Agarwal
Duke University

ABSTRACT

An important problem in terrain analysis is modeling how water flows across a terrain and creates floods by filling up depressions. In this paper we study the *flooding query* problem: Given a rain region R and a query point q on the terrain, quickly determine how much rain has to fall in R so that q is flooded. Available terrain data is often subject to uncertainty which must be incorporated into the terrain analysis. For instance, the digital elevation models of terrains have to be refined to incorporate underground pipes, tunnels, and waterways under bridges, but there is often uncertainty in their existence. By representing the uncertainty in the terrain data explicitly, we can develop methods for flood risk analysis that properly incorporate terrain uncertainty when reporting what areas are at risk of flooding.

We present two results. First, we present a linear size data structure that given a terrain (with no data uncertainty) can answer the flooding query in $O(m \log^2 n)$ time, where m is the number of minima of the terrain at which rain is falling and n is the number of vertices of the terrain. Next, we extend this data structure to handle "uncertain" terrains, using a standard Monte Carlo method. Given a probability distribution on terrains, our data structure solves the problem of determining the probability that if a specified amount of rain falls on a given region a query point is flooded. We implement our data structures and show that they work very well in practice.

CCS CONCEPTS

• **Information systems → Geographic information systems**; *Uncertainty*;

KEYWORDS

Terrains, geographical information systems, contour trees, stochastic process, data uncertainty, Monte Carlo method

ACM Reference format:
Mathias Rav, Aaron Lowe, and Pankaj K. Agarwal. 2017. Flood Risk Analysis on Terrains. In *Proceedings of SIGSPATIAL'17, Los Angeles Area, CA, USA, November 7–10, 2017,* 10 pages.
https://doi.org/10.1145/3139958.3139985

SIGSPATIAL'17, November 7–10, 2017, Los Angeles Area, CA, USA

ACM ISBN 978-1-4503-5490-5/17/11…$15.00
https://doi.org/10.1145/3139958.3139985

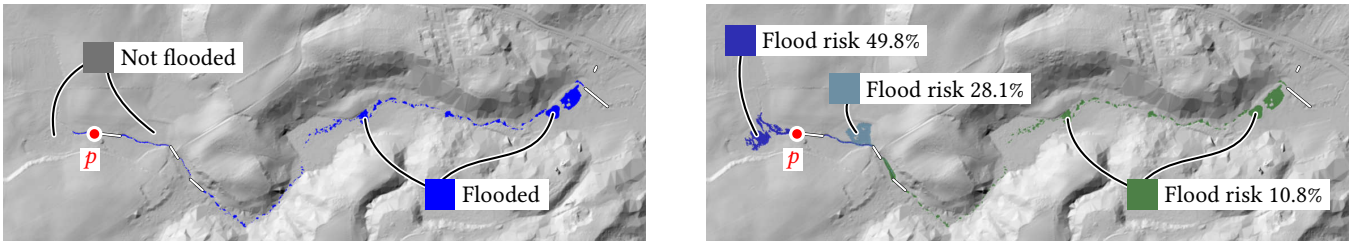
1 INTRODUCTION

An important problem in terrain analysis is modeling how water flows across a terrain and creates floods by filling up depressions. When rains falls, the rate at which a depression fills up depends on its volume and the size of its watershed (the area of the terrain contributing water to the depression). However, it also depends on other depressions filling up. Water that falls on the watershed of a full depression will flow to a neighboring depression, effectively making its watershed larger and thus making it fill up faster. Modeling how depressions fill and spill into other depressions during a flash flood event is therefore an important problem. In this paper we study the *flooding query* problem: Given a rain region R and a query point q on the terrain, quickly determine how much rain has to fall in R so that q is flooded.

Current geographic information systems (GIS) algorithms are often only designed to handle terrains where the height of each vertex is specified precisely, but in reality, terrain models represent only an approximation of the ground truth and are subject to a degree of uncertainty. For instance, in many large, high-resolution terrain datasets such as the ones acquired by LiDAR and other similar technology, raw data is noisy as it includes heights of various natural and anthropological features (e.g. bridges, trees, electric wires, cars, mailboxes, etc.), and one has to clean the data to get the height of the ground. The data cleaning is performed either manually, automatically, or often using a hybrid approach. The output after data cleaning (whether manual or automatic) is not perfect, and this causes uncertainty in the resulting terrain model. By representing the uncertainty in terrain data explicitly we develop methods for flood risk analysis that compute the flooding probability of a point q when a specified volume of rain falls on a given region R of the uncertain terrain.

Previous work. There has been extensive work in the GIS literature on modeling water flow on a terrain. Most flow modeling approaches first remove all depressions by flooding the terrain, that is, they conceptually pour water onto the terrain until all depressions are filled [3, 17, 20]. However, this often leads to unrealistic flow networks, since many important geographical features are removed. Therefore, several methods based on partial flooding algorithms that flood only "small" depressions (based on height, volume or area) have been proposed [1, 5, 9, 11]. Partial flooding methods provide a basis only for approximate solutions to flow modeling, as the underlying assumption is that all "small" depressions are flooded at a certain time, while all "big" depressions are not. To model the flow network at time t accurately, it is necessary to compute the times at which the depressions of the terrain fill and remove the depressions that are full at time t.

Liu and Snoeyink [19] (see also [6]) proposed an $O(n \log n)$ time algorithm, where n is the number of vertices of the terrain, that computes the fill times of all depressions when rain is falling at a constant rate on the entire terrain. In reality, localized extreme



(a) (b)

Figure 1: When it rains on p, the existence of each underground pipe (white segment) determines which areas are flooded. (a) Using the flooding query (without terrain uncertainty), the flooded areas for a given rain amount ψ on p can quickly be determined. (b) When each pipe exists with probability 0.5, our algorithm quickly estimates the flooding probability of each point of the terrain for a given rain amount ψ.

rainfall can affect downstream areas that do not receive heavy rainfall directly, so it is important to model non-uniform events as well. Arge et al. [4] described an algorithm in the I/O model to compute the set of flooded vertices a given volume of rain $\psi \geq 0$ falls on a given region R. We are not aware of any algorithm that can answer a flooding query in sub-linear time in the worst case.

As the resolution in available terrain models increases, there is an increasing demand for methods to handle terrain models that do not fit into main memory. However, the size of the terrain data is only one aspect, and it is equally important to be able to handle uncertainty in the terrain data. To this end, Agarwal et al. [2] proposed data structures to solve several different problems on uncertain terrains such as the *sea-level rise flood risk problem*, that is, they compute the flooding probability of a query point q given that the sea-level at a point p increases by ℓ unit of height. They show that it is enough to evaluate the sea-level rise scenario on a small number of random terrains to guarantee that the estimate of the flooding probability is within ε of correct with high probability.

Our results. We present two main results. In Section 3 we present a linear-size data structure with $O(n \log n)$ preprocessing time that computes for a query rain region R and a query point q the volume it must rain on R before q is flooded. The query time is $O(|R| + m \log^2 n)$, where $|R|$ is the number of vertices at which rain is falling, and m is the number of minima of the terrain at which rain is falling. We first define a simple data structure using the so-called merge tree of the terrain with $O(|R| + (m + H) \log n)$ query time where H is the height of the merge tree which is $\Theta(n)$ in the worst case. We then present our first main result, a fast data structure that uses a heavy-path decomposition [21] of the merge tree to compress the set of full depressions down to $O(\log n)$ sets, each of which is handled in $O(m \log n)$ time for a total query time of $O(|R| + m \log^2 n)$. Figure 1a shows the set of flooded vertices for a given rain amount ψ when R is a single point.

Next, we consider computing the flooding probability of q given that it has rained a volume of ψ on the rain region R on an uncertain terrain. Computing the flooding probability exactly can be shown to be #P-hard by reduction from the Counting-Knapsack problem, but due to space constraints we do not include the proof in the paper. We extend our data structure (in Section 4) to estimate the flooding probability by following a standard Monte-Carlo approach: fix a parameter s, choose s random terrains from the uncertain terrain, and preprocess each of them using the data structure in Section 3 to determine whether q is flooded. Figure 1b shows the flooding probability as estimated by our data structure when R is a single point.

The main technical challenge is to bound the value of s to ensure a desired level of accuracy. Since the number of different terrains can be exponential in n, a standard analysis based on Chernoff bounds suggests that one has to set $s = \Omega(n)$ [2]. Using sophisticated techniques from combinatorial geometry and probabilistic methods, we show that only $O(\frac{1}{\varepsilon^2} \log \frac{n}{\delta})$ samples are needed for all single-point queries, and only $O(\frac{\log n}{\varepsilon^2} \log \frac{n}{\delta})$ samples are needed for a general rain region, to ensure that the answer is correct within error ε with probability at least $1 - \delta$.

We have conducted experiments with our algorithms on two realistic datasets, and in Section 5 we show that our algorithms work well in practice.

2 PRELIMINARIES

Terrains. Let $\mathbb{M}$ be a triangulation of $\mathbb{R}^2$, and let $\mathbb{V}$ be the set of vertices of $\mathbb{M}$; set $n = |\mathbb{V}|$. We assume that $\mathbb{V}$ contains a vertex v_∞ at infinity, and that each edge $\{u, v_\infty\}$ is a ray emanating from u; the triangles in $\mathbb{M}$ incident to v_∞ are unbounded. Let $h : \mathbb{M} \to \mathbb{R}$ be a height function. We assume that the restriction of h to each triangle of $\mathbb{M}$ is a linear map, that h approaches $+\infty$ at v_∞, and that the heights of all vertices are distinct. Given $\mathbb{M}$ and h, the graph of h, called a *terrain* and denoted by Σ_h, is an xy-monotone triangulated surface whose triangulation is induced by $\mathbb{M}$. If h is clear from the context, we denote Σ_h by Σ.

Critical vertices. There is a natural cyclic order on the neighbor vertices of a vertex v, and each such vertex is either an *upslope* or *downslope* neighbor. If v has no downslope (resp. upslope) neighbor, then v is a *minimum* (resp. *maximum*). We will also refer to a minimum as a *sink*. If v has four neighbors w_1, w_2, w_3, w_4 in clockwise order such that $\max(h(w_1), h(w_3)) < h(v) < \min(h(w_2), h(w_4))$ then v is a *saddle* vertex.

Level sets, contours, depressions. Given $\ell \in \mathbb{R}$, the *ℓ-sublevel set of h* is the set $h_{<\ell} = \{x \in \mathbb{R}^2 \mid h(x) < \ell\}$, and the *$\ell$-level set of h* is the set $h_{=\ell} = \{x \in \mathbb{R}^2 \mid h(x) = \ell\}$. Each connected component of $h_{<\ell}$ is called a *depression*, and each connected component of $h_{=\ell}$ is called a *contour*. Note that a depression is not necessarily simply connected, as a maximum can cause a hole to appear; refer to Figure 2 for an example. We define the *height-level map* of Σ_h,

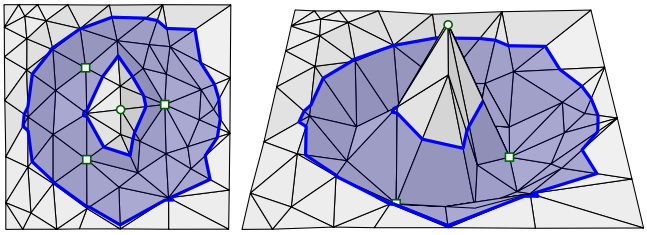


Figure 2: Example terrain showing a depression that is not simply connected. The depression contains three sinks (minima), marked with squares, and a maximum is marked with a circle.

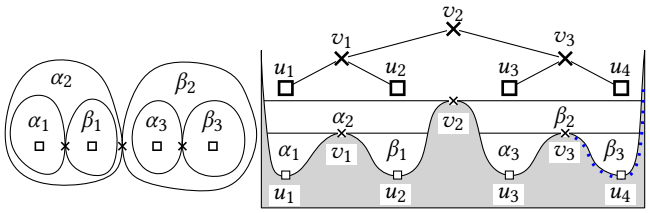


Figure 3: An example terrain with saddle vertices v_1-v_3. Each saddle v_i delimits two maximal depressions α_i and β_i. (a) Terrain seen from above. Sinks are marked with a square and saddles are marked with a cross. The maximal depressions α_1, β_1, α_3 and β_3 are elementary. (b) Terrain seen from the side, showing the watershed of u_4 marked with a dotted line. The merge tree of the terrain shows how the maximal depressions nest within each other.

denoted by M_h, to be the planar subdivision induced by the level sets through all the vertices of the triangulation. Note that M_h can have $\Theta(n^2)$ vertices in the worst case.

A depression β of $h_{<\ell}$ is said to be *delimited by the point* x if x is in the boundary of β, which implies that $h(x) = \ell$. A depression β_1 is *maximal* if every depression $\beta_2 \supset \beta_1$ contains strictly more sinks than β_1. A maximal depression that contains exactly one sink is called an *elementary depression*. Note that each maximal depression is delimited by a saddle, and a saddle that delimits more than one maximal depression is called a *negative saddle*. Refer to Figure 3. A point $u \in \mathbb{R}^2$ is *regular* if it delimits exactly one depression, that is, if it is not a sink or a negative saddle.[1] The *volume* of a depression β of $h_{<\ell}$ is

$$V_h(\beta) = \int_{\beta} (\ell - h(v))dv, \tag{1}$$

and for a regular point u delimiting the depression β_u we define the *depression volume of* u to be $\mu_h(h) = V_h(\beta_u)$.

Merge tree. The maximal depressions of a terrain form a hierarchy that is easily represented using a rooted tree, called the *merge tree* [8, 18] and denoted by T_h. Suppose we sweep a horizontal plane from $-\infty$ to ∞. As we vary ℓ, the depressions in $h_{<\ell}$ vary continuously, but their structure changes only at sinks and negative saddles. If we increase ℓ, then a new depression appears at a sink, and two depressions merge at a negative saddle. The merge tree is a tree that tracks these changes. Its leaves are the sinks of the terrain, and its internal nodes are the negative saddles. Each edge (u, v) in the merge tree corresponds to a depression that is created at v and merged at u. The edges of the merge tree are in one-to-one correspondence with the maximal depressions, that is, we associate each edge (u, v) with the maximal depression delimited by u and containing v. Refer to Figure 3b. For simplicity we assume that the merge tree is binary, that is, each negative saddle delimits exactly two depressions. Non-simple saddles can be unfolded into a number of simple saddles [12].

Formally, T_h is the quotient space in which each depression is represented by a point and connectivity is defined in terms of the quotient topology. Let $\rho_h : \mathbb{M} \to \mathsf{T}_h$ be the associated quotient map, which maps all points that delimit a depression to a single point on an edge of T_h. Fix a point x in $\mathbb{M}$. If x is not a sink or negative saddle, $\rho_h(x)$ lies in the relative interior of an edge in T_h; if x is a sink, $\rho_h(x)$ is a leaf node of T_h; and if x is a negative saddle then $\rho_h(x)$ is a non-leaf node of T_h. We will use h to denote the height function on the points of T_h as well.

Van Kreveld et al. [18] gave an $O(n \log n)$-time algorithm for constructing the merge tree. The algorithm was later extended to 3D by Tarasov and Vyalyi [22], and to arbitrary dimensions by Carr et al. [8]. Carr et al. [9] observed that the merge tree construction can be extended to store the depression volume $\mu_h(x)$ of any point $x \in \mathbb{R}^2$ in the merge tree edge containing $\rho_h(x)$ without increasing the asymptotic construction time or space usage.

Uncertainty model. In our uncertainty model, $\mathbb{M}$ is fixed but the height function h is drawn from a distribution H. We assume that the height of vertex v_i is drawn from a discrete set $\mathsf{H}_i = \{h_i^1, \ldots, h_i^k\}$ and we say that H has *description complexity* k. We do not require the vertex heights to be independent. We use h to denote a random height function drawn from H. H induces distributions Σ_{H}, T_{H}, and M_{H} over terrains, merge trees, and height level maps. Σ_h, T_h, and M_h are random terrain, merge tree, and height level map drawn from Σ_{H}, T_{H}, and M_{H}, respectively.

3 FLOODING QUERIES

In this section we give a precise definition of the flooding query, and then we present two data structures to compute the flooding query. The first data structure (Section 3.2) traverses the merge tree directly and has a query time that depends on the height of the merge tree. The second data structure (Section 3.3) speeds up the worst-cast query time by using a heavy-path decomposition of the merge tree.

3.1 Flooding model

To model water flow on a terrain, each non-sink vertex v is assigned a *flow direction* that indicates the neighbor to which water will flow from v. The flow direction of v is the lowest neighbor v' of v. Each sink u defines a unique *watershed*, consisting of all vertices v that are reachable to u by following the flow directions. Note that each vertex belongs to exactly one watershed, and we assume that each vertex has a pointer to the sink defining the watershed.

For a negative saddle v, the lowest vertices in the two sets of adjacent downslope neighbors of v define two directions, of which

[1] Note that we consider maxima and non-negative saddles to be regular since they do not play a role in our paper.

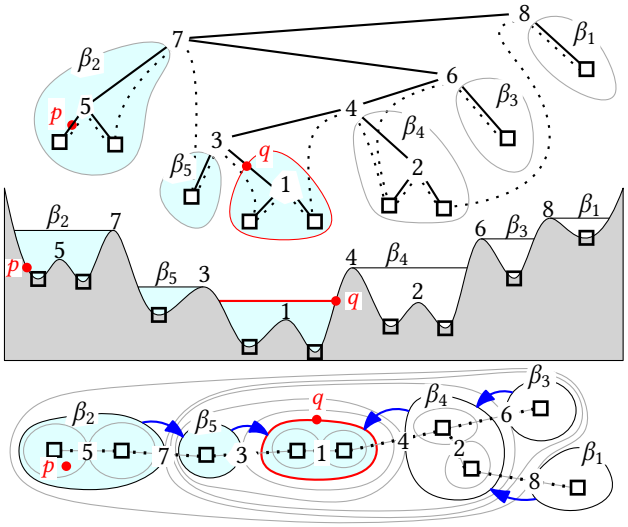


Figure 4: Example terrain and query (p, q). Sinks are marked with squares, and saddles are marked with labels 1-8 indicating the saddle elevation. Dotted lines indicate the two spill sinks of each saddle. **Top:** Merge tree. **Middle:** Terrain seen from the side. **Bottom:** Terrain seen from above. Blue arrows are the edges in S_q.

one is the flow direction of v; the other is the so-called *secondary flow direction*. By following flow directions from v, we reach a sink u_1 containing v in its watershed, furthermore, by following flow directions from the secondary flow direction of v, we reach another sink u_2. We say that u_1 and u_2 are the *spill sinks* of v. Let β_1 be the maximal depression delimited by v and containing u_1. Then the *spill sink of* β_1 is u_2. Similarly we define the spill sink of β_2 to be u_1.

We let R denote a *rain region*, which is specified as a vector on the vertices of the terrain such that for each vertex $v \in \mathbb{V}$, $R(v) \geq 0$ indicates the rate at which rain falls on v; we require $\sum_v R(v) = 1$. We denote by $|R|$ the number of vertices with positive rainfall in R, and we assume that R is represented as a list of $|R|$ pairs $(v, R(v))$.

Our flooding model follows the depression filling model of Liu and Snoeyink [19]. When rain falls on a region R of the terrain, water follows flow directions and accumulates in depressions of the terrain. When rain falls on the watershed of a sink u, the rate at which the elementary depression β containing u fills up is equal to the sum of $R(v)$ over all vertices v in its watershed. When a maximal depression β containing u fills up, water on the watershed of u spills over the saddle delimiting β into a neighboring watershed of a sink u'. We refer to this event as a *spill event*. At this event the watershed of u is merged into the watershed of u', which increases the rate at which u' is filling up.

The above process defines a sequence of spill events, each event marking a sink u as full and merging the watershed of u into a neighboring watershed. In our model, the maximal depressions of the terrain fill up at a constant rate between any two consecutive spill events. That is, after a spill event occurs at time t_1 and until the next occurs at time t_2, the volume of water in each elementary depression is a non-decreasing linear function of time. The *R-fill time* (or *fill time* for brevity) of a point q is the time when the rain

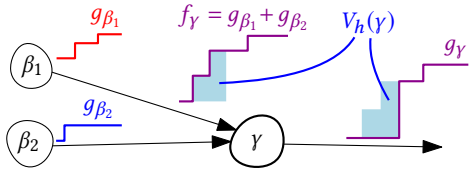


Figure 5: The fill rate f_γ is computed from the spill rates g_{β_1} and g_{β_2}, and then the spill rate g_γ is computed from f_γ.

water (on R) reaches q. Given a height function h and a rain region R, let $\sigma_h(R, q)$ denote the total volume it must rain on R before the water surface reaches q. For simplicity we assume throughout the paper that the point q is not a saddle. Our definitions and algorithms can easily be extended to handle saddles. Since we assume that rain falls at the rate of unit volume, the fill time of q is the same as $\sigma_h(R, q)$.

When rain falls on a vertex v, water follows flow directions towards a sink u. Given a rain region R, we can define an equivalent rain region R' in which $R'(u)$ for a sink u is the sum of $R(v)$ over vertices v in its watershed, and $R'(v)$ is zero when v is not a sink. Then $\sigma_h(R, q) = \sigma_h(R', q)$ for all q, so we may always replace R by R' in $O(|R|)$ time and then assume that it is only raining on sinks. When all rain in R is contained in the watershed of a sink p, we denote $\sigma_h(R, q)$ by $\sigma_h(p, q)$.

Any given point q is contained in a sequence of maximal depressions $\alpha_1 \supset \cdots \supset \alpha_k \ni q$, each α_i delimited by a saddle v_i which delimits another maximal depression β_i. Note that these saddles form a path in T_h from q to the root. We refer to the maximal depressions $\beta_1, \ldots, \beta_k$ as the *q-tributaries*. The q-tributaries form the *q-tributary tree*, denoted by S_q and defined as follows. S_q is a directed tree, with each node pointing to its parent, rooted at the maximal depression α_k containing q, and the non-root nodes are the q-tributaries $\beta_1, \ldots, \beta_k$.[2] We define the parent of β_i in S_q as the node γ containing the spill sink of β_i, which is either the root α_k or β_j for some $j > i$. Refer to Figure 4, in which β_1 and β_3 are the children of β_4.

Given a rain region R and a query point q, for each node $\beta \in S_q$ (either a q-tributary or α_k), we denote by $f_\beta(t)$ the *fill rate of* β *at time* t, which is the rate at which rain is falling directly on β plus the rate at which other q-tributaries are spilling into β. The fill rate $f_\beta : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is a monotone piecewise constant function, and we denote the set of all such functions by G. We denote by t_β the *fill time of* β and by $g_\beta(t)$ the *spill rate of* β *at time* t, and we define these formally as follows. For any function $f \in G$ and $\psi \geq 0$ we let $\tau(f, \psi)$ be the time at which the integral of f is ψ, that is, $\tau(f, \psi)$ is t_0 such that $\int_0^{t_0} f(t)dt = \psi$. Refer to Figure 5. Then $t_\beta = \tau(f_\beta, V_h(\beta))$, and we define the spill rate as $g_\beta(t) = 0$ for $t < t_\beta$ and $g_\beta(t) = f_\beta(t)$ for $t \geq t_\beta$.

Using S_q we can now recursively define the fill rate of β as follows: the fill rate of β is the rate at which rain falls directly into β, plus the spill rates of the children of β in S_q. Let $R(\beta)$ be the sum of R over all the vertices in watersheds of sinks in β, and let B be the set of children of β in S_q. We define $f_\beta(t) = R(\beta) + \sum_{\gamma \in B} g_\gamma(t)$.

[2] There is a natural bijection between the saddles $v_1, \ldots, v_k$ and the q-tributaries $\beta_1, \ldots, \beta_k$, and thus we sometimes refer to a saddle v_i as a node of S_q.

In particular, when β is a leaf of S_q, f_β is the constant function $R(\beta)$.

Now the rain volume function $\sigma_h(R, q)$ is the time at which the root of S_q, α, has accumulated a volume of $\mu_h(q)$, the depression volume of q, so we define $\sigma_h(R, q) = \tau(f_\alpha, \mu_h(q))$.

We define the *tributaries from R to q*, denoted by $A(R, q)$, as the subtree of S_q induced by the vertices which have non-zero fill-rate functions.

Remark. Note that $A(R, q)$ changes only when q is in a different edge of the merge tree T_h, so $A(R, q) = A(R, q')$ for all q, q' on the same edge of T_h.

Single-source spilling. If R is contained in the watershed of a single sink p, $A(p, q)$ is a path of tributaries $(\gamma_1, \ldots, \gamma_\ell)$, and we observe that $t_{\gamma_1} = V_h(\gamma_1)$, and $t_{\gamma_{i+1}} = t_{\gamma_i} + V_h(\gamma_{i+1})$. Thus we have

$$\sigma_h(p, q) = \sum_{i=1}^{\ell} V_h(\gamma_i) + \mu_h(q). \tag{2}$$

3.2 A simple data structure

Now we describe a data structure for computing $\sigma_h(R, q)$ for a rain region R and query point q. Our data structure consists of T_h, for each saddle v in T_h it stores (i) pointers to the two spill sinks of v, and (ii) the volumes of the two maximal depressions delimited by v. We compute a depth-first numbering of the leaves of T_h and store at each node of T_h the DFS interval associated with its subtree. Using the DFS intervals it is possible to check in constant time whether a sink u is contained in a subtree of T_h. We store T_h as a link-cut tree [21] to support lca (lowest common ancestor) queries in $O(\log n)$ time. The construction time is $O(n \log n)$, and the data structure has size $O(n)$. Now we describe the query algorithm to compute $\sigma_h(R, q)$, first in the *single-point case* when R is restricted to the watershed of a single sink p, and then in the case when R is any region.

Single-point source. Suppose all rain in R falls on the watershed of a single sink p. Our algorithm to compute $\sigma_h(p, q)$ traverses the path $(v_1, \ldots, v_k)$ in T_h from the root to the edge containing q and along the way identifies the path $A(p, q) = (\gamma_1, \ldots, \gamma_\ell)$ in the q-tributary tree S_q. For each $i \le \ell$, let p_i be the sink in γ_i at which water initially collects when γ_i starts to fill up. Then p_1 is p, and p_{i+1} is the spill sink of γ_i. For $j = 1, \ldots, k$ we visit the saddle v_j while maintaining the invariant that we have identified the tributaries that are delimited by the first $j - 1$ saddles $v_1, \ldots, v_{j-1}$, and we have furthermore identified the next spill sink p_{j+1}. After we have visited v_k, we have identified all the tributaries between p and q. We return the rain volume $\sigma_h(p, q)$ computed using (2), where each term $V_h(\gamma_i)$ is computed by looking up the volume stored at the merge tree node for the saddle delimiting γ_i.

Computing $\sigma_h(p, q)$ in this way requires $O(k)$ time, where k is the number of maximal depressions that contain q.

Region source. We now extend this algorithm to handle rain on the watersheds of multiple sinks of the terrain as specified by the rain region R. As in the single-point case, the algorithm visits the saddles $(v_1, \ldots, v_k)$ on the path from the root of T_h to the edge containing q while incrementally identifying $A(R, q)$, the tributaries between R and q.

First we add the rain rate $R(v)$ of each non-sink vertex v to the rain rate of the sink u containing v in its watershed. Then we add the rain rate of each sink u to the node β_i containing u in the q-tributary tree S_q, where β_i is found by querying the lca of u and q in T_h. For each $i \le k$, let ρ_i denote the resulting rain rate of β_i.

Then we iteratively construct S_q while computing the fill-rate of each tributary. For $i = 1, \ldots, k$, we visit the saddle v_i while maintaining the following invariant: For each edge $(\beta_r, \beta_s) \in \mathsf{S}_q$ with $r < i \le s$, g_{β_r} is stored at the node $\beta_s \in \mathsf{S}_q$. When visiting the saddle v_i which delimits β_i, we compute the fill-rate function $f_{\beta_i}(t) = \sum_j g_{\beta_j}(t) + \rho_i$ where the sum is taken over all children β_j for β_i in S_q. Given $f_{\beta_i}(t)$, g_{β_i} can be be computed by computing the fill time t_{β_i}. The parent of β_i in S_q can be computed in $O(\log n)$ by computing the lca of the spill sink of β_i and q in T_h.

At the end, α, the root of S_q, stores the spill rate functions of the children of α as well as the rain rate of α, so we compute the fill-rate function f_α and return $\sigma_h(p, q) = \tau(f_\alpha, \mu_h(q))$.

Let g, g' be two spill rate functions. Using the representation described below, we support the following two operations:

> **Add.** An Add operation replaces g and g' by $g + g'$.
> **Truncate.** The Truncate operation takes a volume ψ, computes $t_0 = \tau(g, \psi)$ and sets $g(t) := 0$ for $t < t_0$.

Since each spill rate function g is piecewise constant, we can represent g as a linked list of pairs (t_i, Δ_i), where t_i is a time at which g changes value and Δ_i is the increase of g at time t_i. The collection of pairs is stored in a strict Fibonacci heap [7] keyed on time. Add is then computed as a merge of two heaps, and Truncate is computed by iteratively removing the top pair (t_i, Δ_i) and maintaining the total volume of rain at time t_i until it exceeds ψ. Add takes constant time, and Truncate takes time $O((r + 1) \log n)$, where r is the number of pairs (t_i, Δ_i) removed by Truncate. Since Add does not change the total number of spill rate function point-pairs (t_i, Δ_i) and the initial number of point-pairs is $O(k)$, the time required in total for the Truncate operations is $O(k \log n)$.

Theorem 3.1. *Given a triangulation $\mathbb{M}$ of $\mathbb{R}^2$ with vertex set $\mathbb{V}$ of size n and a height function $h : \mathbb{M} \to \mathbb{R}$ that is linear on each face of $\mathbb{M}$, a data structure of size $O(n)$ can be constructed in time $O(n \log n)$ that for any rain region R and query point $q \in \mathbb{R}^2$ returns $\sigma_h(R, q)$ in $O(|R| + k \log n)$ time, where k is the number of maximal depressions containing q.*

3.3 The fast data structure

In this section we present a fast data structure for computing $\sigma_h(R, q)$. First we define a number of tree structures that our data structure will use. Then we describe the data structure itself and analyze the space and construction time. Finally, we describe the query algorithm to compute $\sigma_h(R, q)$.

Heavy-path tree. We define the *heavy-path decomposition* [21] of T_h, as follows. For each internal node v in T_h, let w and w' be the children of v such that w has the larger subtree, with ties broken arbitrarily. Then we call the edge (v, w) *heavy* and the edge (v, w') *light*. This definition ensures that every parent has exactly one heavy edge to a child, so the heavy edges partition the tree T_h into a set of heavy paths $\{\pi_i\}$ ending at sinks (where some paths consist of a single sink).

Next, we define the *heavy-path tree* P_h, as follows: the nodes of P_h are the heavy paths of T_h, and P_h contains the edge (π_i, π_j) whenever the parent of the head of π_j in T_h is a node in π_i. We observe that for any light edge (v, w) in T_h, corresponding to an edge in P_h, the subtree rooted at w contains at most half as many vertices as the subtree rooted at v. Therefore the height of P_h is at most $\log_2(n)$.

For each heavy path $\pi = (v_1, \ldots, v_k, u)$ where v_i is the parent of v_{i+1} in the merge tree, and v_k is the parent of the sink u, we define the *tributary tree of* π, denoted by S_π, as the subtree of the u-tributary tree containing only the tributaries delimited by the saddles $v_1, \ldots, v_k$. We define the height of a tributary $\beta \in \mathsf{S}_\pi$ to be the height of the saddle delimiting β. Each tributary tree node $\beta \in \mathsf{S}_\pi$ stores the volume of β as well as the prefix sum of the volumes of tributaries from β to the root of S_π.

For a given node $\beta \in \mathsf{S}_\pi$, let $\rho(\beta, \alpha_u)$ be the path in S_π from β to the root α_u. By storing S_π as a link-cut tree [21], the tributary tree supports the following *tributary-sum* operation in $O(\log n)$ time: Given a node $\beta \in \mathsf{S}_\pi$ and a height z such that $h(\beta) > z \geq h(\alpha_u)$, return the edge (w, w') of $\rho(\beta, \alpha_u)$ with $h(w) > z \geq h(w')$ along with the sum of tributary volumes on the path from β to w in S_π. Furthermore, the tributary tree supports an lca query in $O(\log n)$ time.

Data structure. Given a triangulation $\mathbb{M}$ and a height function h, our data structure for computing $\sigma_h(p, q)$ preprocesses the terrain to compute the merge tree T_h, the heavy-path tree P_h, and the tributary trees S_π for each heavy path π. We assume that T_h is augmented to compute the depression volume $\mu_h(x)$ of any point $x \in \mathbb{R}^2$ in $O(\log n)$ time. Each saddle v stores pointers to the two spill sinks of v, to the heavy path $\pi \in \mathsf{P}_h$ containing v, and to its node in S_π. After constructing T_h in time $O(n \log n)$ [18], the additional structures can be built in time $O(n \log n)$ using standard techniques. Each node stores $O(1)$ information, so the total space is $O(n)$.

Query procedure. We now describe the procedure for computing $\sigma_h(p, q)$, the volume of the rain that must fall on point p before q gets flooded. We omit the extension to general rain regions R due to space constraints. Let $A(p, q) = (\gamma_1, \ldots, \gamma_\ell)$ be the sequence of tributaries between p and q. For each $i \leq \ell$, let v_i be the saddle that delimits γ_i. Then v_1 is the lca of p and q in the merge tree, which can be computed in $O(\log n)$ time by following parent pointers in P_h. Let $(\pi_1, \ldots, \pi_k)$ be the path in P_h from the heavy path containing v_1 to the heavy path containing q.[3] Our query procedure performs a tributary-sum operation on each of the paths $\pi_1, \ldots, \pi_k$ to determine the contribution of each heavy path to $\sigma_h(p, q)$, as follows.

The sequence of saddles $(v_1, \ldots, v_\ell)$ delimiting the maximal depressions $A(p, q)$ form a subsequence of the path from the root of T_h to q, so we may partition $A(p, q)$ into consecutive subsequences $(A_1, \ldots, A_k)$, each A_i consisting of the tributaries delimited by saddles in π_i (Figure 6a). Fix $i \leq k$, and let $A_i = (\gamma_r, \gamma_{r+1}, \ldots, \gamma_s)$. We denote by $\sigma(A_i)$ the contribution of A_i to the sum (2), that is, $\sigma(A_i) = \sum_{j=r}^{s} V_h(\gamma_j)$. Let $\mathcal{P}(A_i)$ be the sink in γ_r on which water initially collects when γ_r starts to fill up. For the first heavy path π_1, $\mathcal{P}(A_1)$ is the source p, and for each subsequent path π_i, $\mathcal{P}(A_i)$ is the spill sink of the predecessor of γ_r in $A(p, q)$. Our query algorithm computes the contribution $\sigma(A_i)$ using a tributary-sum operation on π_i for each $i \leq k$, and it also computes the starting sink $\mathcal{P}(A_{i+1})$ for $i < k$. Finally, our query algorithm returns $\sigma_h(p, q)$ as $\sum_{i=1}^{k} \sigma(A_i) + \mu_h(q)$.

[3] If q lies on a light edge (u, v), then we consider q to be contained by the heavy path below the edge, that is, the heavy path for which v is the head.

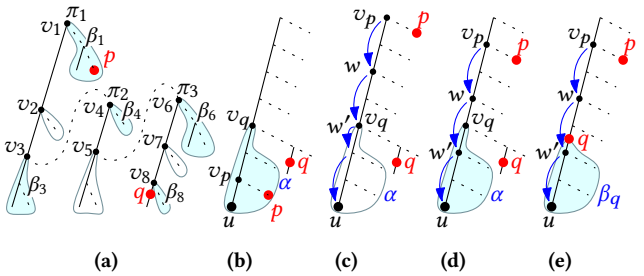


Figure 6: (a) Given the heavy-path decomposition, $A(p, q) = (\beta_1, \beta_3, \beta_4, \beta_6, \beta_8)$ is partitioned into $A_1 = (\beta_1, \beta_3)$, $A_2 = (\beta_4)$, $A_3 = (\beta_6, \beta_8)$. **(b-e)** The cases identified by our algorithm when processing a heavy path.

We now describe the procedure to compute $\sigma(A_i)$ and $\mathcal{P}(A_{i+1})$ from $\mathcal{P}(A_i)$. For now we assume $i < k$, which implies that q is contained in a tributary of π_i; later we will sketch how to handle the last iteration $i = k$ when q lies on the heavy path π_i. Let u be the sink at the tail of π_i, let v_p (resp. v_q) be the lowest ancestor of $\mathcal{P}(A_i)$ (resp. q) in the merge tree which lies in π_i, and let α be the maximal depression delimited by v_q and containing u. We compute v_p and v_q in $O(\log n)$ time by following parent pointers in P_h from $\mathcal{P}(A_i)$ and q. There are three possible cases depending on the relative height of v_p and v_q. If $h(v_p) = h(v_q)$ then A_i is empty, and $\mathcal{P}(A_{i+1}) = \mathcal{P}(A_i)$. If $h(v_p) < h(v_q)$ then A_i consists of the single maximal depression α (Figure 6b) so $\sigma(A_i) = V_h(\alpha)$, and $\mathcal{P}(A_{i+1})$ is the spill sink of α. Finally if $h(v_p) > h(v_q)$, consider the tributaries $\gamma \in \mathsf{S}_{\pi_i}$ along the path from v_p towards the sink u with $h(\gamma) \geq h(v_q)$. If v_q lies in this path, then the sequence of tributaries, excluding the last depression delimited by v_q, is equal to the sequence A_i (Figure 6c). Then $\sigma(A_i)$ is the sum of tributary volumes from v_p to the predecessor of v_q along this path. If v_q does not lie on this path, this sequence of tributaries followed by the depression α is equal to A_i (Figure 6d). Then $\sigma(A_i)$ is the sum of tributary volumes along the path, plus $V_h(\alpha)$. In both of these cases, $\mathcal{P}(A_{i+1})$ is computed to be the spill sink of the last tributary in A_i. In the last iteration when $i = k$ (Figure 6e), q lies on an edge of π_k, and A_k consists of the tributaries γ on the path from v_p towards u for which $h(\gamma) \geq h(q)$ followed by the depression delimited by q.

We visit $k = O(\log n)$ vertices of P_h and spend $O(\log n)$ time on each of them, so the total query time is $O(\log^2 n)$.

The fast query algorithm to compute $\sigma_h(p, q)$ can be extended to the general case of computing $\sigma_h(R, q)$ in time $O(|R| + m \log^2 n)$, but we do not provide the details due to lack of space.

Theorem 3.2. *Given a triangulation $\mathbb{M}$ of $\mathbb{R}^2$ with vertex set $\mathbb{V}$ of size n and a height function $h : \mathbb{M} \to \mathbb{R}$ that is linear on each face of $\mathbb{M}$, a data structure of size $O(n)$ can be constructed in time $O(n \log n)$ that for any rain region R and point $q \in \mathbb{R}^2$, in $O(|R| + m \log^2 n)$ time, returns $\sigma_h(R, q)$.*

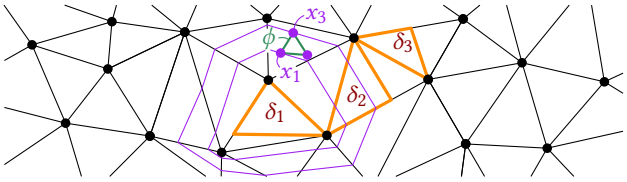


Figure 7: Example triangulation with triangle $\phi \in \widetilde{\mathsf{M}}$, triangles δ_1-$\delta_3 \in \Delta$, and the lowest point x_1 and highest point x_3 in ϕ. δ_1 is contained in β_{x_1}, δ_3 is disjoint from β_{x_3}, and δ_2 intersects the boundary of β_x for all $x \in \phi$.

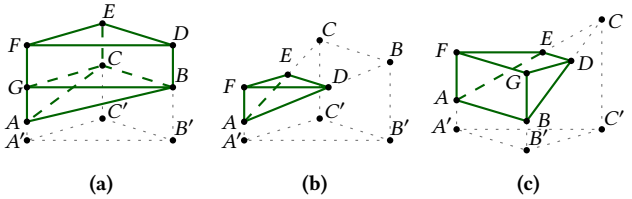


Figure 8: Measuring depression volume above a triangle ABC of Σ. (a) The triangle is completely contained in the depression. (b) The triangle with upper flat edge is intersected by the contour DE. (c) The triangle with lower flat edge is intersected by the contour DE.

4 MONTE CARLO ALGORITHM

In this section we present a data structure to estimate the probability of flooding on an uncertain terrain. Let H be a distribution on height functions of description complexity k. Given a rain region R, volume $\psi \geq 0$, and point $q \in \mathbb{M}$, we define the probability that q is flooded, denoted by $\pi(R, q, \psi)$, as

$$\pi(R, q, \psi) = \Pr_{h \sim \mathsf{H}}[h \in \mathsf{H}_{R,q,\psi}],$$

where $\mathsf{H}_{R,q,\psi}$ is the set $\{h \in \mathsf{H} \mid \sigma_h(R, q) \leq \psi\}$.

Our data structure to estimate $\pi(R, q, \psi)$ works as follows. During preprocessing, we sample s height functions $H = \{h_1, \ldots, h_s\}$ i.i.d. from H, with s to be specified later. For each $i \leq s$, we construct in $O(n \log n)$ time the data structure f_i, of size $O(n)$, described in Section 3.3, on h_i. The size of our data structure is $O(sn)$, and the preprocessing time is $O(sn \log n)$.

Given a query (R, q, ψ), we query each f_i to determine the number c of height functions $h_i \in H$ for which $\sigma_{h_i}(R, q) \leq \psi$. We return c/s as the probability estimate $\hat{\pi}(R, q, \psi)$. The query time is $O(s|R| \log^2 n)$.

Now we will bound s, the number of samples in our Monte Carlo scheme, to ensure that the approximation error does not exceed ε on any query with probability at least $1 - \delta$ for fixed parameters $\varepsilon, \delta \in (0, 1)$. First we analyze the rain-volume function σ_h to show that it has a nice structure. Exploiting this structure, we apply ideas from statistical learning to bound the value of s.

Rain-volume function. Recall that the height-level map of a height function h is the planar subdivision induced by the $h(v)$-level sets for all $v \in \mathbb{V}$. We fix the distribution H on the height functions, as defined in Section 2. M_H is the distribution of height-level maps induced by H. Let $\widetilde{\mathsf{M}}$ denote the triangulation of the overlay of $\mathbb{M}$ with all the height-level maps in M_H. Agarwal et al. [2] showed that $\widetilde{\mathsf{M}}$ has complexity $O(n^3k^8)$ despite being the overlay of k^n different height-level maps in the worst case.

Lemma 4.1. *The following properties hold for any $h \in \mathsf{H}$ and for any face ϕ in $\widetilde{\mathsf{M}}$:*

(i) No level-set through a vertex of Σ intersects ϕ.
(ii) For any two points $x, x' \in \phi$, the depression delimited by x contains the same set of sinks as the depression delimited by x'.
(iii) For any fixed $p \in \mathbb{V}$ and for any $x, x' \in \phi$, $A(p, x) = A(p, x')$, where $A(p, x)$ is the set of tributaries between p and x.

Proof. (i) follows from the construction. (ii) follows from the fact that x and x' belong to the same edge of T_h. (iii) follows from the remark in Section 3.1 since x and x' belong to the same edge of T_h. □

Lemma 4.2. *For any height function $h \in \mathsf{H}$ and any face ϕ (of any dimension) of $\widetilde{\mathsf{M}}$, the depression volume function is a bivariate cubic function denoted by μ_h^ϕ, that is, for all $x \in \phi$, $\mu_h(x) = \mu_h^\phi(x)$.*

Proof. We first split each triangle of $\mathbb{M}$ into two triangles as follows. For each triangle of $\mathbb{M}$ with vertices $A, B, C \in \mathbb{V}$ such that $h(A) < h(B) < h(C)$, let D be the point on AC such that $h(D) = h(B)$. We split the face by the line BD into a bottom half ABD and a top half BCD. The ℓ-level set, for $h(A) \leq \ell \leq h(C)$, intersects the face ABC in a line segment parallel to BD, which implies that no level set intersects both the bottom half and the top half of a face ABC.

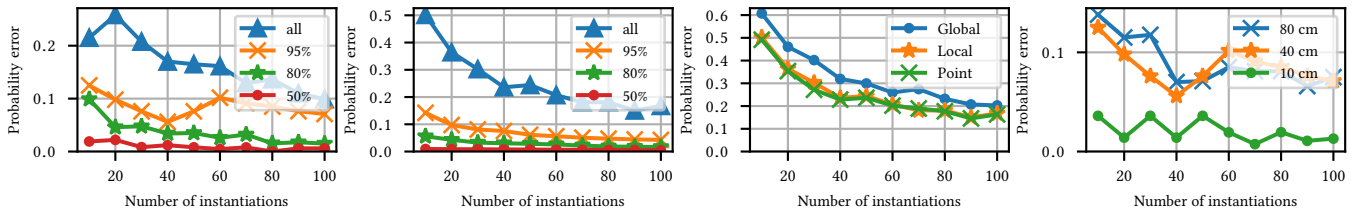
Let Δ denote this resulting set of triangles in $\mathbb{M}$.

$$\mu_h(x) = \sum_{\delta \in \Delta} \int_{\delta \cap \beta_x} (h(x) - h(v))dv. \tag{3}$$

Fix $\phi \in \widetilde{\mathsf{M}}$, and let x_1, x_2 and x_3 be the vertices of ϕ such that $h(x_1) \leq h(x_2) \leq h(x_3)$ (Figure 7). For all $x \in \phi$, $\beta_{x_1} \subseteq \beta_x \subseteq \beta_{x_3}$, where $\beta_{x_1}, \beta_x, \beta_{x_3}$ are the depressions delimited by respectively x_1, x, and x_3. By construction, the boundary of β_x crosses the same set of edges of $\mathbb{M}$ as β_{x_1}. Using this property, we will argue that each term in the sum (3) is a cubic function of $x \in \phi$, from which the claim follows.

Fix a triangle $\delta \in \Delta$. By Lemma 4.1(i), no level-set through a vertex of δ intersects ϕ, so δ is contained in β_{x_1}, disjoint from β_{x_3}, or δ intersects the boundary of β_x for all $x \in \phi$. Thus, the term of μ_h^ϕ corresponding to the half-face δ falls into one of three cases.

If δ is disjoint from β_{x_3}, then the term is an empty integral which is zero. If $\delta \subseteq \beta_{x_1}$ (refer to Figure 8a), then $\delta \cap \beta_x = \delta$ for all $x \in \phi$. The portion of β_x over δ is a triangular prism whose top face is a copy of δ at height $h(x)$ and the bottom face is the triangle of Σ corresponding to δ. Let h_δ be the height of the highest vertex of δ. Then the volume of this prism is a constant plus $\text{Area}(\delta) \cdot (h(x) - h_\delta)$, and thus a linear function. If for all $x \in \phi$, $\delta \cap \beta_x \subset \delta$, meaning the contour through x intersects δ, then we argue that the integral is a cubic function. Let A, B, C be the vertices of δ such that either $h(A) < h(B) = h(C)$ (Figure 8b) or $h(A) = h(B) < h(C)$ (Figure 8c). The integral over $\delta \cap \beta_x$ equals the volume of the frustum Ω that lies between the plane $z = h(A)$ and the plane $z = h(x)$. Each vertex of Ω can be written as a linear function of $h(x)$ with coefficients depending on A, B, and C. By tetrahedralizing Ω we can write its volume as the sum of volumes of tetrahedra. Since the volume of



(a) Denmark. **(b) Indiana, $\Delta = 40$ cm, $k = 8$.** **(c) Varying size of region.** **(d) Varying amount of rain.**

Figure 9: (a-b) Error when it rains 40 cm on the rain region. (c) Max error when varying the size of the Indiana region. (d) 95-th percentile error on the Denmark medium-sized region when the amount of rain varies.

each tetrahedron is a cubic function of $h(x)$, it follows that the integral over $\delta \cap \beta_x$ is a cubic function of $h(x)$.

Finally, $h(x)$ is a linear function of x for $x \in \phi$. Hence $\mu_h(x)$ is a bivariate cubic function of x. □

We omit proofs of the following three statements due to space constraints.

Lemma 4.3. *Let h be a height function and p a vertex of $\mathbb{M}$. Restricting the query point q to a face ϕ of $\widetilde{\mathsf{M}}$, the rain-volume function is a univariate cubic function of height denoted by $s_{h,p,\phi}$, that is, for all $q \in \phi$, $\sigma_h(p,q) = s_{h,p,\phi}(h(q))$.*

Corollary 4.4. *Let h be a height function and p a vertex of $\mathbb{M}$. Restricting the query point q to a face ϕ of $\widetilde{\mathsf{M}}$, the rain-volume function is a bivariate cubic function of q denoted by $\sigma_{h,p,\phi}$, that is, for all $q \in \phi$, $\sigma_h(p,q) = \sigma_{h,p,\phi}(q)$.*

Corollary 4.5. *For a fixed rain region R, for any height function $h \in \mathsf{H}$, and for any face $\phi \in \widetilde{\mathsf{M}}$, the rain-volume function is a piecewise cubic function of height.*

Point-source query. We now bound the number of samples needed in the case when the source is a single vertex $p \in \mathbb{V}$. For a point $q \in \mathbb{R}^2$ and for a volume $\psi \geq 0$, we use $\pi(p,q,\psi)$ to denote the flooding probability of q if ψ units of rain falls at p, and $\hat{\pi}(p,q,\psi)$ to denote the estimate of $\pi(p,q,\psi)$ returned by our data structure. Let $\mathrm{err}(p,q,\psi) = |\pi(p,q,\psi) - \hat{\pi}(p,q,\psi)|$. For a face $\phi \in \widetilde{\mathsf{M}}$ and a vertex $p \in \mathbb{V}$, we define

$$\mathrm{err}(p,\phi) = \sup_{\substack{q \in \phi \\ \psi \geq 0}} \mathrm{err}(p,q,\psi).$$

Finally, let err be the maximum value of $\mathrm{err}(p,\phi)$ over all vertices $p \in \mathbb{V}$ and $\phi \in \widetilde{\mathsf{M}}$.

First we bound the probability of the event that $\mathrm{err}(p,\phi) > \varepsilon$. We use the so-called *Veronese map* [15] to map a bivariate cubic function to a linear function with 9 variables. In particular, each monomial $x^i y^j$ is mapped to a new variable z_{ij} for $1 \leq i+j \leq 3$, $i,j \geq 0$. That is, we define the map $\nu : \mathbb{R}^2 \to \mathbb{R}^9$ as

$$\nu(x,y) = \langle z_{ij} \mid i,j \geq 0, 1 \leq i+j \leq 3 \rangle.$$

Now a cubic function $f(x,y) = \sum_{i=0}^{3} \sum_{j=0}^{3-i} a_{ij} x^i y^j$ is mapped to the linear function $\tilde{f}(z_{10},\ldots,z_{03}) = a_{00} + \sum_{1 \leq i+j \leq 3} a_{ij} z_{ij}$.

Lemma 4.6. *For any given $p \in \mathbb{V}$ and face ϕ of $\widetilde{\mathsf{M}}$,*

$$\Pr[\mathrm{err}(p,\phi) > \varepsilon] = O(\exp(-s\varepsilon^2)). \tag{4}$$

Proof. Fix a vertex $p \in \mathbb{V}$ and a face ϕ of $\widetilde{\mathsf{M}}$. We consider the set system $(\mathsf{H}, \Omega_{p,\phi})$, where $\Omega_{p,\phi} = \{\mathsf{H}_{p,q,\psi} \mid q \in \phi, \psi \geq 0\}$, and $\mathsf{H}_{p,q,\psi}$ is the set $\{h \in \mathsf{H} \mid \sigma_h(p,q) \leq \psi\}$. By Corollary 4.4, for any bivariate height function $h \in \mathsf{H}$ and for a fixed source $p \in \mathbb{V}$, $\sigma_h(p,q)$ restricted to ϕ is a bivariate cubic function $\sigma_h^\phi(q)$. Let $\tilde{\sigma}_h^\phi(\nu(q))$ denote the linearization of $\sigma_h^\phi(q)$ as defined above. We now define a new set system (S,R) where $S = \{\tilde{\sigma}_h^\phi \mid h \in \mathsf{H}\}$ and $R = \{\{\tilde{\sigma}_h^\phi \mid \tilde{\sigma}_h^\phi(\nu(q)) \leq \psi\} \mid q \in \phi, \psi \geq 0\}$. Note that S is a set of linear functions, so a classical result in random sampling by Vapnik and Chervonenkis [23] (see also [14]) implies for a random sample of size $O(\frac{1}{\varepsilon^2} \log \frac{1}{\delta})$, $\mathrm{err}(p,\phi) \leq \varepsilon$ with probability at least $1-\delta$. This immediately implies the lemma. □

Since $|\widetilde{\mathsf{M}}| = (nk)^{O(1)}$, using a union bound we obtain the following:

Theorem 4.7. *If the number of samples is $s = \Omega(\frac{1}{\varepsilon^2} \lg \frac{nk}{\delta})$, then the error is at most ε with probability at least $1-\delta$.*

Fixed region query. We now extend the above analysis to the case when the source is a fixed region R. For a point $q \in \mathbb{R}^2$ and for a volume $\psi \geq 0$, we use $\pi(R,q,\psi)$ to denote the flooding probability of q if ψ units of rain falls at R, and $\hat{\pi}(R,q,\psi)$ to denote the estimate of $\pi(R,q,\psi)$ returned by our data structure. Let $\mathrm{err}(R,q,\psi) = |\pi(R,q,\psi) - \hat{\pi}(R,q,\psi)|$. For a face $\phi \in \widetilde{\mathsf{M}}$, we define

$$\mathrm{err}(R,\phi) = \sup_{\substack{q \in \phi \\ \psi \geq 0}} \mathrm{err}(R,q,\psi).$$

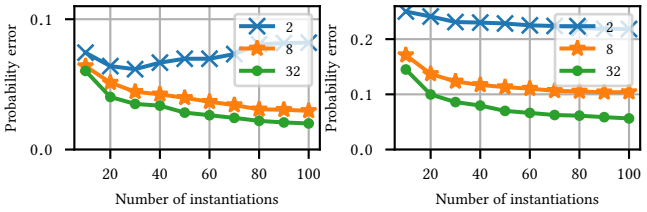
Finally, let $\mathrm{err}(R)$ be the maximum value of $\mathrm{err}(R,\phi)$ over all $\phi \in \widetilde{\mathsf{M}}$.

Let H_ϕ^* denote the set of all possible height functions for ϕ. Since ϕ is contained inside a triangle of $\mathbb{M}$, H_ϕ is a set of k^3 height functions $\{h_1^*,\ldots,h_{k^3}^*\}$. For each $i \leq k^3$ we let H_i be the set of height functions equal to h_i^* on ϕ.

Lemma 4.8. *For any fixed rain region R and any face ϕ of $\widetilde{\mathsf{M}}$,*

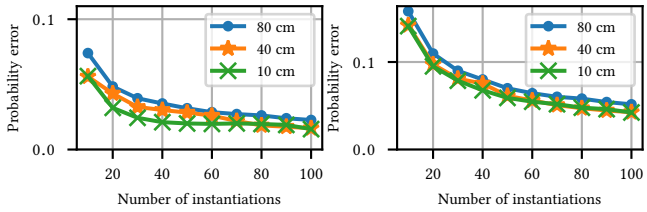
$$\Pr[\mathrm{err}(R,\phi) > \varepsilon] = O(\exp(-s\varepsilon^2 k^3 \log n)). \tag{5}$$

Proof. Fix a face ϕ of $\widetilde{\mathsf{M}}$. We consider the set system $(\mathsf{H}, \Omega_\phi)$, where $\Omega_\phi = \{\mathsf{H}_{R,q,\psi} \mid q \in \phi, \psi \geq 0\}$. By Corollary 4.5, for any bivariate height function $h \in \mathsf{H}$, $\sigma_h(R,q)$ restricted to ϕ is a piecewise cubic function $\sigma_h^\phi(q) = s_h^\phi(h_i^*(q))$ with at most n pieces, where $h_i^* \in \mathsf{H}_\phi$ is the restriction of h to ϕ. For each $i \leq k^3$ we define a set system (S_i, R_i) where $S_i = \{s_h^\phi \mid h \in \mathsf{H}_i\}$ and $R_i = \{\{s_h^\phi \in S_i \mid$



(a) 80-**th percentile.** **(b)** 95-**th percentile.**

Figure 10: Varying k on Indiana with $\Delta = 40$ cm, 40 cm rain.



(a) 80-**th percentile.** **(b)** 95-**th percentile.**

Figure 11: Varying Δ on Indiana with $k = 8$, 40 cm rain.

$s_h^{\phi}(h_i^*(q)) \le \psi\} \mid q \in \phi, \psi \ge 0\}$. We now define a new set system (S, R) where $S = \bigcup_{i=1}^{k^3} S_i$ and $R = \bigcup_{i=1}^{k^3} R_i$. Note that each S_i is a set of functions that intersect pair-wise at most $6n$ times, so the analysis by Grünbaum [13] and Har-Peled [14] implies for a random sample of size $O(\frac{k^3 \log n}{\varepsilon^2} \log \frac{1}{\delta})$, $\mathrm{err}(R, \phi) \le \varepsilon$ with probability at least $1 - \delta$. We omit the details due to space constraints. □

Since $|\widetilde{\mathsf{M}}| = (nk)^{O(1)}$, using a union bound we obtain the following:

Theorem 4.9. *If the number of samples is* $s = \Omega\Big(\frac{k^3 \log n}{\varepsilon^2} \lg\big(\frac{nk}{\delta}\big)\Big)$, *then the error on any fixed region R is at most ε with probability at least $1 - \delta$.*

5 EXPERIMENTS

In this section we present the experiments we have conducted with two real datasets to demonstrate the efficiency and usefulness of our method. Our experiments show that only a few samples are needed in practice, and by inspecting the output we learn something interesting about the flood risk in an uncertain terrain.

We have implemented the simple data structure in Section 3.2 in C++, and we use a computer with a 3.5 GHz Intel Xeon E5-1650 CPU, 128 GB RAM, running Linux 4.4, for the experiments.

We did not conduct experiments with the fast data structure, as our experiments are about uncertainty, not speed.

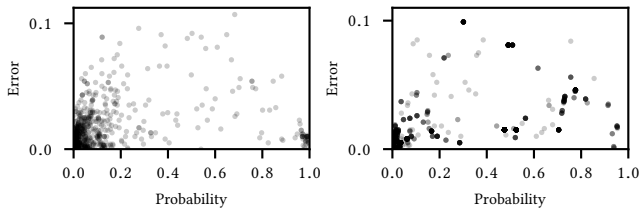
Terrains. We study the performance of our algorithms on two terrain datasets. The *Indiana dataset* is a 0.4 km^2 model of an area 1 km northeast of Holland, Indiana, USA, extracted from the publicly available 5 ft resolution DEM of Indiana [16]. The *Denmark dataset* is a 33 km^2 model of the city of Vejle, Denmark, extracted from the publicly available 1.6 m resolution DEM of Denmark [10]. Both are high-resolution grid datasets containing respectively 190 000 and 13 000 000 vertices, and for efficiency we use an implementation of our algorithm tuned to grid datasets rather than general TINs. The datasets do not have uncertainty on the vertex heights, and we introduce uncertainty below.

Uncertainty. For the Indiana dataset we introduce uncertainty on the vertex heights by choosing parameters k and Δ and generating data as follows. For each vertex v we draw k numbers H_v^k from the uniform distribution $U(h(v) - \Delta/2, h(v) + \Delta/2)$. To instantiate a random terrain we pick the height of a vertex v uniformly at random from H_v^k. In this uncertain terrain, vertex heights are independent.

The source of uncertainty in the Denmark dataset is the set of publicly available so-called *hydrological corrections* for the terrain model [10]. Each hydrological correction is a polygon with an associated height function and represents an underground pipe, a waterway under a bridge, or a similar terrain feature that permits water flow under the surface represented by the terrain model. Of the 114 000 corrections in the dataset for all of Denmark, 220 corrections are inside the Vejle region we use in our experiments. We assign existential probabilities to each of the 220 corrections by taking independent samples from a normal distribution with mean $\mu = 0.5$ and standard deviation $\sigma = 0.05$. To instantiate a random terrain, we update the base terrain model with the height functions of a random subset of the corrections, where each correction is included independently according to its probability. The vertex heights in this uncertain terrain are *not* independent, since the vertices contained in the same hydrological correction are all updated to the height function of the correction if the correction is included in the instantiated terrain.

Queries. We considered the following queries (R, q, ψ) to our data structure. For the rain region R we considered convex regions with different sizes from the smallest region consisting of a single vertex, to the largest region consisting of all terrain vertices, and we set the same rain rate on all vertices on which it is raining. For each dataset we selected regions with large downstreams in the terrain to ensure that flooding queries would exercise flooding of areas far away from the rain region. Except where noted, our experiments use a specific small rain region of 4000 m^2 for the Indiana model and 0.24 km^2 for the Denmark model. For the point q we query all vertices of the terrain. We selected rain volumes ψ corresponding to between 1 cm and 80 cm rain on the rain region. As an example of the output of our algorithm, Figure 1b shows a 1.7 km × 0.7 km portion of the output when it rains 800 m^3 water on the marked point p of the Denmark dataset, and Figure 13 shows a 3.1 km × 1.2 km portion of the output when it rains 40 cm on the region R of the Denmark dataset.

Measuring the convergence. For each query (R, q, ψ) and round j we compute $|\hat{\pi}_j(R, q, \psi) - \pi(R, q, \psi)|$, where $\hat{\pi}_j(R, q, \psi)$ denotes the estimate of $\pi(R, q, \psi)$ computed on the first j instances of the uncertain terrain. Over all query points $q \in \mathbb{V}$ we report the 50-th, 80-th, 95-th, or 100-th percentile of these probability differences. Note that if $\pi(R, q, \psi) = 0$, then $\hat{\pi}(R, q, \psi) = 0$ and the error is 0; the error is also 0 when $\pi(R, q, \psi) = 1$. We discard such queries when we measure the errors; including them would only reduce the error. Figures 9a-b illustrate that the error decreases quickly, with 50 samples yielding a reasonably small error.



(a) Indiana model. **(b) Denmark model.**

Figure 12: Errors vs. exact values.

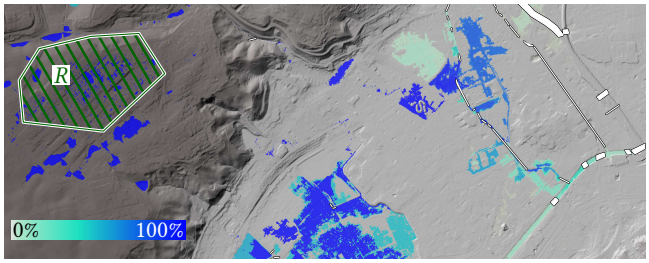


Figure 13: Example output with 40 cm rain on region R.

Since computing $\pi(R, q, \psi)$ is hard, we instead apply our Monte Carlo method with 3000 samples and use the resulting estimates as the exact value of $\pi(R, q, \psi)$.

Size of region. First we measured the convergence as we vary the size of the rain region. Figure 9c illustrates that the error decreases quickly for both the single-point case, the small region case, and the case when it is raining on the entire Indiana terrain. The error is greatest when it is raining on the entire terrain which agrees with our theoretical analysis.

Amount of rain. We measured the convergence as we vary ψ, the amount of rain. Figure 9d illustrates that the error increases as the amount of rain increases. This is to be expected since the number of vertices with a non-zero flood risk increases as one increases the amount of rain.

Description complexity k. We considered varying the description complexity k, that is, the number of possible heights in each vertex on the Indiana model. We fix $\Delta = 40$ cm and choose k from $\{2, 8, 32, 64\}$. For each vertex v we draw random sets $H_v^2 \subset H_v^8 \subset H_v^{32} \subset H_v^{64}$ from the uniform distribution $U_v = U(h(v)-\Delta/2, h(v)+\Delta/2)$, and we compare the probability estimates for $k = 2, 8, 32$ with the error probability $\pi(R, q, \psi)$ computed on 3000 samples of the uncertain terrain with $k = 64$. We expect that smaller values of k will converge faster, but will converge at a higher error level. Figure 10 illustrates that the error does decrease as k increases, however, the effect of k on the rate of convergence is not as clear.

Height variation Δ. We measured the convergence as the height variation Δ varies on the Indiana dataset. The vertex heights on the base dataset are between 150 m and 170 m above sea-level, and the median height distance between a vertex and its lowest neighbor is 12 cm. Accordingly we varied Δ between 10 cm and 80 cm. Figure 11 shows that the error increases as Δ increases, which is expected.

Distribution of errors vs. exact values. We examined the distribution of errors as a function of the exact probability. Figures 12a and 12b illustrate that the estimation errors of probabilities are independent of the underlying exact probability. For the simulation in Figure 12a we used the Indiana dataset with $\Delta = 40$ cm, $k = 8$, 40 cm rain. For the simulation in Figure 12b we used the Denmark dataset with 40 cm rain.

ACKNOWLEDGMENTS

Work by Rav is supported by the Danish National Research Foundation and Innovation Fund Denmark; work was done while visiting Duke University. Work by Lowe and Agarwal is supported by NSF under grants CCF-15-13816, CCF-15-46392, and IIS-14-08846, by ARO grant W911NF-15-1-0408, and by Grant 2012/229 from the U.S.-Israel Binational Science Foundation.

REFERENCES

[1] PK Agarwal, L Arge, and K Yi. 2006. I/O-efficient Batched Union-Find and its Applications to Terrain Analysis. In *Proc. 22nd annu. Sympos. on Comp. Geom.* 167–176.
[2] PK Agarwal, S Mukherjee, and W Zhang. 2015. Contour trees of uncertain terrains. In *Proc. 23rd SIGSPATIAL Intl. Conf. on Advances in GIS*. 43.
[3] L Arge, JS Chase, P Halpin, L Toma, JS Vitter, D Urban, and R Wickremesinghe. 2003. Efficient Flow Computation on Massive Grid Terrain Datasets. *GeoInformatica* 7, 4 (2003), 283–313.
[4] L Arge, M Rav, S Raza, and M Revsbæk. 2017. I/O-Efficient Event Based Depression Flood Risk. In *Proc. 19th Workshop on Algorithm Engineering and Experiments*. 259–269.
[5] L Arge and M Revsbæk. 2009. I/O-efficient Contour Tree Simplification. In *Intl. Sympos. on Algos. and Computation*. 1155–1165.
[6] L Arge, M Revsbæk, and N Zeh. 2010. I/O-efficient computation of water flow across a terrain. In *Proc. 26th annu. sympos. on Comp. geom.* 403–412.
[7] GS Brodal, G Lagogiannis, and RE Tarjan. 2012. Strict fibonacci heaps. In *Proceedings of the forty-fourth annu. ACM sympos. on Theory of computing*. 1177–1184.
[8] H Carr, J Snoeyink, and U Axen. 2003. Computing contour trees in all dimensions. *Comp. Geom.* 24, 2 (2003), 75–94.
[9] H Carr, J Snoeyink, and M Panne. 2010. Flexible isosurfaces: Simplifying and displaying scalar topology using the contour tree. *Comp. Geom.* 43, 1 (2010), 42–58.
[10] Danish Geodata Agency. 2007. The Danish Elevation Model DHM-2007/Terræn_bro. http://eng.gst.dk. (2007).
[11] A Danner, T Mølhave, K Yi, PK Agarwal, L Arge, and H Mitásová. 2007. TerraStream: from elevation data to watershed hierarchies. In *Proc. 15th Annu. ACM Intl. Sympos. on Advances in GIS*. 28.
[12] H Edelsbrunner, J Harer, and A Zomorodian. 2001. Hierarchical Morse complexes for piecewise linear 2-manifolds. In *Proc. 17th annu. sympos. Comp. geom.* 70–79.
[13] B Grünbaum. 1975. Venn diagrams and independent families of sets. *Mathematics Magazine* 48, 1 (1975), 12–23.
[14] S Har-Peled. 2011. *Geometric approximation algorithms*. Vol. 173.
[15] J Harris. 1992. *Algebraic geometry: a first course*.
[16] Indiana Spatial Data Portal. 2013. Indiana Orthophotography (RGBI), LiDAR and Elevation. http://gis.iu.edu/datasetInfo/statewide/in_2011.php. (2013).
[17] SK Jenson and JO Domingue. 1988. Extracting topographic structure from digital elevation data for geographic information system analysis. *Photogrammetric Engineering and Remote Sensing* 54, 11 (1988), 1593–1600.
[18] M Kreveld, R Oostrum, C Bajaj, V Pascucci, and D Schikore. 1997. Contour trees and small seed sets for isosurface traversal. In *Proc. 13th annu. sympos. on Comp. geom.* 212–220.
[19] Y Liu and J Snoeyink. 2005. Flooding triangulated terrain. In *Proc. 11th Intl. Sympos. on Spatial Data Handling*. 137–148.
[20] JF O'Callaghan and DM Mark. 1984. The extraction of drainage networks from digital elevation data. *Computer Vision, Graphics, and Image Processing* 28, 3 (1984), 323–344.
[21] DD Sleator and RE Tarjan. 1983. A data structure for dynamic trees. *Journal of computer and system sciences* 26, 3 (1983), 362–391.
[22] SP Tarasov and MN Vyalyi. 1998. Construction of contour trees in 3D in O(n log n) steps. In *Proc. 14th annu. sympos. on Comp. geom.* 68–75.
[23] VN Vapnik and AY Chervonenkis. 1971. On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities. *Theory Probab. Appl.* 16, 2 (1971), 264–280.