DOI:10.1145/3410413

Flood-Risk Analysis on Terrains

By Aaron Lowe, Pankaj K. Agarwal, and Mathias Rav

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 a number of *flood-risk* related problems: given a terrain Σ, represented as a triangulated *xy*-monotone surface with *n* vertices, a rain distribution $\mathcal{R}$, and a volume of rain ψ, determine which portions of Σ are flooded. We give an overview of efficient algorithms for these problems as well as explore the efficacy and efficiency of these algorithms on real terrains.

1. INTRODUCTION

Flooding can be extremely dangerous and damaging. The United States experienced the wettest 12-month period from June 2018 to May 2019, with major flooding in the Midwest affecting millions of people and causing several billion dollars in damages. Being able to accurately and quickly model flooding can help predict and prepare for the risks. Flood-risk analysis has been studied widely across multiple research communities including environmental science, engineering, machine learning, and GIS communities: see Section 7.

Flood risk analysis also has been a focus of a number of companies as well. SCALGO[22] is a software development and services company that uses massive terrain dataprocessing technology to provide a flood risk platform for Scandinavian countries. 3Di Water Management[1] provides interactive hydrodynamic simulation software combining overland, channel, sewer and ground water flow. Fathom[13] uses high-resolution global datasets and hydrological modeling to provide flood hazard data for many applications, including insurance and disaster response.

In this paper, we will focus on two key problems related to flood-risk analysis, which are defined more formally in later sections:

Terrain-flood query: given a terrain Σ and a rain pattern, determine which portions of Σ will be flooded. (See Figure 1 for an example.)

Point-flood query: In some applications, the terrain Σ is fixed and we wish to know whether a query point on Σ will be flooded for a given rain pattern. Preprocess Σ into a data structure so that for a given rain pattern and query point $q \in \Sigma$, one can quickly determine whether q is flooded. Alternatively, one can ask how long rain must fall before q is flooded.

When rain falls, the rate at which a depression fills up depends not only on its shape and the size of its watershed (the area of the terrain that contributes water to the depression), but also on other depressions filling up. Water falling on the watershed of a depression that is already filled flows to a neighboring depression, effectively making its watershed larger and thus making it fill up faster. Maintaining how depressions fill and spill into other depressions during a flash flood makes the above problem challenging.

Figure 1. A terrain-flood query over a region of Philadelphia, PA, USA. The areas marked in blue are flooded, with regions that water flows over marked in orange.

We present an efficient algorithm for the terrain-flood query in Section 4. The algorithm works by sweeping descending contours on the terrain, tracking where water flows and determining which depressions become full. A key feature of the algorithm is that once we find a contour delimiting a flooded region, we can mark the enclosed region as flooded and prune it from consideration.

For the point-flood query, we present in Section 5 an algorithm that preprocesses the terrain into a data structure so that queries can be answered quickly. If we assume the single-flow direction (SFD) model in which water from a vertex flows along the steepest descending edge, we describe an algorithm that, after preprocessing the terrain, can answer point-flood queries in time logarithmic in the number of vertices on the terrain. We also briefly discuss algorithms for the *flood-time query* that asks *when*, rather than *if*, a point will become flooded. These algorithms all work by recognizing that not all depressions are equally important from the perspective of the query point. The terrain can be simplified into a set of depressions called the tributaries. The local behavior within each tributary can then be ignored. Instead, the algorithms depend only on the global behavior: does the tributary become full, and if so, which downstream tributaries does it spill into?

We have implemented the algorithms for terrain-flood and point-flood queries and tested them on real terrains. We show that the algorithms are efficient in practice, across a variety of queries, and give some analysis as to how varying

The results described here originally appeared in "Flood-risk analysis on terrains under the multiflow-direction model"[18] and "Flood risk analysis on terrains,"[21] respectively.

The original version of this paper was published in the *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems* (Nov. 2017) Article No. 36.

the query affects the running time. We also compare terrain-flood queries qualitatively under two variants, the multi-flow direction (MFD) and single-flow direction (SFD) models, showing cases where the flooded regions are significantly different under the two (Section 6).

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 $\Sigma = (\mathbb{M}, h)$, is an xy-monotone triangulated surface whose triangulation is induced by $\mathbb{M}$.

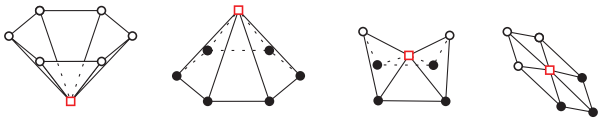
Critical vertices. There is a natural cyclic order on the neighbor vertices of a vertex v of $\mathbb{M}$, and each such vertex u is either an *upslope* or *downslope* neighbor, that is, $h(u) > h(v)$ or $h(u) < h(v)$, respectively. If v has no downslope (resp. upslope) neighbor, then v is a *minimum* (resp. *maximum*). We 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. If a vertex is not a critical point, call it *regular*. See Figure 2.

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*. Each connected component of the boundary of a depression is a *contour*. A contour not passing through a critical vertex is a simple polygonal cycle. Note that a depression is not necessarily simply connected, as a maximum can cause a hole to appear.

For a point $x \in \mathbb{M}$, a depression β_x of $h_{<\ell}$ is said to be *delimited by the point x* if x lies on 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*. For a maximal depression β, let $\mathrm{Sd}(\beta)$ denote the saddle delimiting β, and let $\mathrm{Sk}(\beta)$ denote the set of sinks in β.

Merge tree. 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, denoted by T_h, is a tree that tracks these changes.

Figure 2. From left to right: minimum (sink), maximum, saddle, and regular vertices. Upslope and downslope neighbors are marked in white and black, respectively.



Its leaves are the sinks of the terrain, and its internal nodes are the negative saddles. The edges of T_h are in one-to-one correspondence with the maximal depressions of Σ_h, that is, we associate each edge (u, v), for $h(u) > h(v)$, with the maximal depression delimited by u and containing v. The point of height $\ell \in [h(v), h(u)]$ on edge (u, v) represents the depression of $h_{<\ell}$ contained in β_v. We assume that T_h has an edge from the root of T_h extending to $+\infty$, corresponding to the depression that extends to ∞. See Figure 4. For simplicity, we assume that T_h is binary, that is, each negative saddle delimits exactly two depressions. Nonsimple saddles can be unfolded into a number of simple saddles.[12]

Let u be a negative saddle, let (u, v_1) and (u, v_2) be two down edges in T_h from u, and let (w, u) be the up edge from u. We call the depression associated with (u, v_2) (resp. with (w, u)) as the *sibling* (resp. *parent*) (depression) of that associated with (u, v_1). Van Kreveld et al.[16] gave an $O(n \log n)$-time algorithm for constructing the merge tree in 2D. The algorithm was later extended to 3D by Tarasov and Vyalyi,[23] and to arbitrary dimensions by Carr et al.[8]

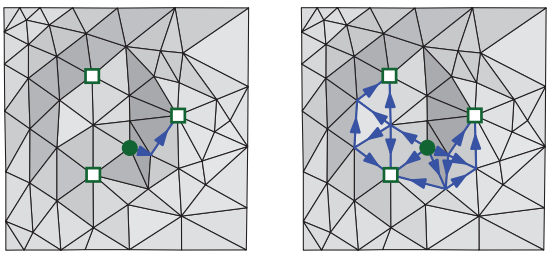
T_h can be preprocessed in $O(n)$ additional time so that for a point $x \in \mathbb{R}^2$, $\mathrm{Vol}(\beta_x)$, the volume of the depression delimited by x can be computed in $O(\log n)$ time. In the following sections, we will be working with a fixed height function, so will drop the subscript h from T_h, Vol_h, etc.

3. FLOODING MODEL

Flow graph and flow functions. We transform $\mathbb{M}$ into a directed acyclic graph $\mathcal{M}$, referred to as the *flow graph*, by directing each edge $\{u, v\}$ of $\mathbb{M}$ in downward direction, that is, from u to v if $h(u) > h(v)$, and from v to u otherwise. For each (directed) edge (u, v), we define the *local flow* $\lambda(u, v)$ to be the portion of water arriving at u that flows along the edge (u, v) to v.

The value of $\lambda(u, v)$ is, in general, based on relative heights of the downslope neighbors of u. We will refer to the general version in which water can flow along multiple downward edges from u as the *multi-flow direction* (MFD) model. If $\lambda(u, v) > 0$ for exactly one downslope edge from u, we will refer to this as the *single-flow direction* (SFD) model. See Figure 3 for an example of how these models differ. In some cases, notably for point-flood and flood-time queries, the SFD model will admit more efficient algorithms. We will not focus on the specifics of the local flow function, and only

Figure 3. Rain falls at the local maximum at the green circle toward local minima marked with squares. Left: an SFD model, water flows along a single path to a single minimum. Right: an MFD model, water flows along multiple paths to three local minima.



assume that it is specified and for a pair u, v can be retrieved in $O(1)$ time.

Following the edges of $\mathbb{M}$, water reaches a set of sinks of $\mathbb{M}$. We define a *flow function* $\phi : \mathbb{V}^2 \to [0, 1]$, which specifies the proportion of water that flows from a vertex u to another vertex v. Note that under the MFD model, water can flow from u to v along many paths. The flow function is defined recursively as follows:

$$\phi(u,v) = \begin{cases} 1 & \text{if } u = v, \\ \sum_{(u,w)\in\mathbb{E}(\mathbb{M})} \lambda(u,w)\cdot\phi(w,v) & \text{otherwise.} \end{cases} \quad (1)$$

For a maximal depression β, we define

$$\phi(u,\beta) = \sum_{s\in \mathrm{Sk}(\beta)} \phi(u,s)$$

to be the portion of water that reaches from a vertex u to β. Recall that $\mathrm{Sk}(\beta)$ is the set of sinks in β.

If a maximal depression β delimited by saddle u is full, we define $\phi_\beta(u, v)$ to be the modified flow function, computed as if the flooded vertices have a height of $h(u)$.

Rain distribution. We let $\mathcal{R}$ denote a *rain distribution*, which is specified as a probability distribution on the vertices of the terrain, that is, for each vertex $v \in \mathbb{V}$, $\mathcal{R}(v) \geq 0$ indicates the rate at which it rains on v, and we require that $\sum_v \mathcal{R}(v) = 1$. For a given depression β, let $\mathcal{R}(\beta) = \sum_{v\in\beta} \mathcal{R}(v)$ be the portion of rain falling directly into β. We denote by $|\mathcal{R}|$ the number of vertices with positive rainfall in $\mathcal{R}$, and we assume that $\mathcal{R}$ is represented as a list of $|\mathcal{R}|$ pairs $(v, \mathcal{R}(v))$. In practice, $|\mathcal{R}| \ll n$.

Flood propagation. Our flooding model follows a similar depression-filling model as Liu and Snoeyink.[17] As rain falls according to a distribution $\mathcal{R}$ on Σ, water flows along the downward edges according to the flow function and accumulates in depressions of Σ. When a maximal depression β_i fills up, water spills from the saddle v_i delimiting β_i toward sinks in the sibling depression. We refer to this event as a *spill event*.

The above process defines a sequence of spill events, each event marking a sink u as full, and redistributing the rain falling on u to other sinks. See Figure 4 for an example. In our model, the maximal depressions of Σ 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 maximal depression is a nondecreasing linear function of time.

Tributaries. Any given point $q \in \mathbb{M}$ 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 with sibling depression β_i. These saddles form a path in T from q to the root. We refer to the maximal depressions $\beta_1, \ldots, \beta_k$ as the *tributaries of q* and denote them by $\mathcal{T}_q$. See Figure 4.

Fill and spill rates. For a maximal depression β, we define the *fill rate* $F_\beta : \mathbb{R}_{\geq 0} \to [0, 1]$ as the rate at which water is arriving in the depression β as a function of time. That is, the rate at which rain is falling directly in β plus the rate at which other depressions are spilling water into β. The fill rate F_β is a monotonically nondecreasing piecewise-constant function. Similarly, we define the *spill rate* $S_\beta : \mathbb{R}_{\geq 0} \to [0, 1]$ as the rate (as a function of time) at which water spills from β through the saddle that delimits β. Let τ_β, called the *fill time*, be the time at which β becomes full. Let β' be the sibling depression of β. Then,

$$S_\beta(t) = \begin{cases} 0 & \text{if } t < \tau_\beta \vee \tau_{\beta'} \leq \tau_\beta. \\ F_\beta(t) & \text{if } t > \tau_\beta \wedge \tau_{\beta'} > \tau_\beta. \end{cases}$$

By the definition of the fill rate, for any maximal depression β, its initial fill rate is

$$F_\beta(0) = \sum_{s\in\mathrm{Sk}(\beta)} \sum_{v\in\mathbb{V}} R(v)\phi(v,s), \quad (2)$$

which is how much rain water flows initially to the sinks of β. We can define $F_\beta(0)$ recursively using (1) as follows: abusing the notation a little, let $F_v(0)$ denote the water reaching a vertex v at time 0. Then,

$$F_v(0) = R(v) + \sum_{(w,v)\in\mathbb{E}(\mathbb{M})} F_w(0)\lambda(w,v) \quad (3)$$

$$F_\beta(0) = \sum_{s\in\mathrm{Sk}(\beta)} F_s(0). \quad (4)$$

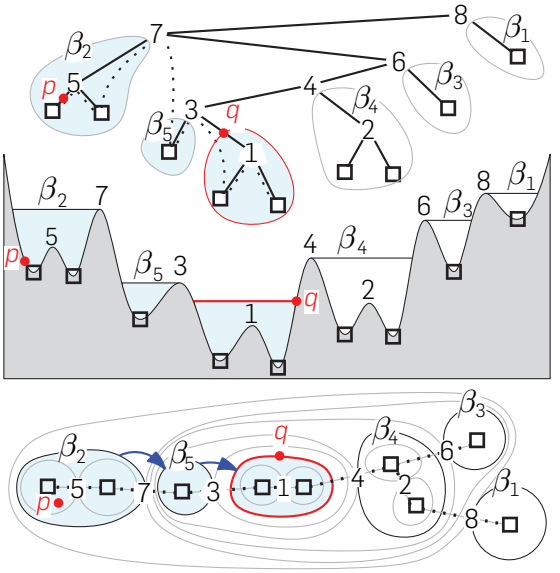
For any $t \geq 0$, the fill rate of β at time t is the direct rain reaching β plus the water spilling into β from its tributaries that are full. That is,

$$F_\beta(t) = F_\beta(0) + \sum_{\alpha\in\mathcal{T}_\beta} S_\alpha(t)\phi(\mathrm{Sd}(\alpha),\beta). \quad (5)$$

Fill and spill volumes. For a depression β, we similarly define $\mathcal{F}_\beta, \mathcal{S}_\beta : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ as *fill- and spill-volume* functions of β, that is, $\mathcal{F}_\beta(t)$ tells how much water has arrived in depression β by time t, and $\mathcal{S}_\beta(t)$ tells how much water has spilled from β by time t.

$$\mathcal{F}_\beta(t) = \int_0^t F_\beta(x)dx \quad \text{and} \quad \mathcal{S}_\beta(t) = \int_0^t S_\beta(x)dx.$$

Figure 4. Example terrain and point-flood query (p, q). Sinks are marked with squares, and saddles are marked with labels 1–8 indicating the saddle elevation. Dotted lines indicate spill events. Top: Merge tree with tributaries of q, β_1-β_5 marked. Middle: Terrain seen from the side. Bottom: Terrain seen from above.



By definition of the spill rate, letting β' be the sibling depression of β, we have

$$\mathcal{S}_\beta(t) = \begin{cases} 0 & \text{if } t < \tau_\beta \vee \tau_{\beta'} \leq \tau_\beta. \\ \mathcal{F}_\beta(t) - \text{Vol}(\beta) & \text{if } t > \tau_\beta \wedge \tau_{\beta'} > \tau_\beta. \end{cases}$$

4. TERRAIN-FLOOD QUERIES

In this section, we describe an algorithm for answering a terrain-flood query. That is, given a rain distribution $\mathcal{R}$ and a volume ψ, determine which vertices of $\mathbb{M}$ will be flooded if a volume of ψ rain falls according to the distribution $\mathcal{R}$. A key idea of the algorithm is that if a vertex v is flooded, then all points lying in the depression β_v are also flooded, so we do not have to process the vertices lying in β_v. We just need to know how much (if any) water will spill to its downstream tributaries. Using this simple observation, we compute the flooded vertices of $\mathbb{M}$ for a given $\mathcal{R}$ as follows.

Overview of the algorithm. As a preprocessing step, we construct the merge tree T, and in doing so, we augment T with a linear-size data structure so that for each point q (i) the edge of T containing q and (ii) $\text{Vol}(\beta_q)$ can each be computed in $O(\log n)$ time.

For a given rain distribution $\mathcal{R}$, we first compute how much rain directly falls in each maximal depression initially. For a maximal depression β, let $\hat{\mathcal{R}}(\beta)$ be the rate of rain falling on vertices in β which are not contained in any other maximal depression. Using the recurrence

$$\mathcal{R}(\alpha) = \hat{\mathcal{R}}(\alpha) + \sum_{(\alpha,\beta)\in\mathsf{T}} \mathcal{R}(\beta),$$

$\mathcal{R}(\alpha)$ for all maximal depressions $\alpha \in \mathsf{T}$ can be computed in $O(|\mathcal{R}| + m)$ time.

Next, we process the vertices in descending height order, and at each vertex v, we maintain the following:

- The set of *active depressions* that are not completely filled depressions in the $h(v)$-sublevel set
- For each active depression α (i) the fill volume $\mathcal{F}_\alpha$, that is, the volume of water in α, and (ii) the set of edges crossing into α, denoted $\boldsymbol{E}(\alpha)$, and finally
- For each edge $e \in \boldsymbol{E}(\alpha)$, the volume of rain flowing along e denoted by $\Lambda(e)$

As we sweep the vertices from top to bottom, we will find the height at which depressions are flooded. At this point, we mark the corresponding depression as flooded and do not consider any vertices contained in the flooded depression.

We now describe in detail how we process each vertex.

Processing a nonsaddle vertex. Let α_v be the smallest maximal depression containing β_v. A key observation is that $\mathcal{F}_\alpha$ is the same as $\mathcal{F}_{\alpha_v}$. Therefore, it does not change at a nonsaddle vertex and thus has already been computed at an earlier step. Then, if $\text{Vol}(\beta_v) \leq \mathcal{F}_{\beta_v}$, that is, β_v is flooded, we mark all vertices contained in β_v as flooded and remove β_v from the set of active depressions. If β_v is not flooded, for each edge (v, w) in T, we compute the water flowing along it as:

$$\Lambda(v, w) = \lambda(v, w)\left(\mathcal{R}(v)\psi + \sum_{(u,v)\in\mathbb{E}} \Lambda(u, v)\right). \tag{6}$$

Processing a saddle vertex. If v is a nonflooded saddle vertex delimiting two maximal depressions α, α', in addition to the above process, we must also compute the volume of rain in each of the two depressions. To do so, partition the edges $\boldsymbol{E}(\beta_v)$ into the two sets $\boldsymbol{E}(\alpha)$ and $\boldsymbol{E}(\alpha')$ and compute the volume of rain crossing into α (resp. α') as $\sum_{e\in\boldsymbol{E}(\alpha)} \Lambda(e)$ (resp. $\sum_{e\in\boldsymbol{E}(\alpha)} \Lambda(e)$.) See Figure 5 for an example. These sums can be computed in a total of $O(n \log n)$ time summed over all saddles. Using this value along with the value of $\mathcal{R}(\alpha)$ (resp. $\mathcal{R}(\alpha')$) in (7), we can compute $\mathcal{F}_\alpha$ (resp. $\mathcal{F}_{\alpha'}$).

Then, the volume of water in a depression β is the amount falling directly in it, plus the amount of water flowing into it,

$$\mathcal{F}_\beta = \psi\,\mathcal{R}(\beta) + \sum_{e\in\boldsymbol{E}(\beta)} \Lambda(e). \tag{7}$$

Finally, we use this value along with the depression volumes to check if α or α' is flooded. Note that because v is not flooded, at most, one of these depressions will be flooded. If one is flooded, without loss of generality, let it be α. In this case, mark α as flooded, and add α' to the set of active depressions. Then, the volume of rain spilling from α into α' will be $\mathcal{F}_\alpha - \text{Vol}(\alpha)$. Let $\lambda'(v, w)$ be the modified flow function, computed as if the flooded neighboring vertices have a height of ℓ. Then, we update the flow of water along the edges from v as follows:

$$\Lambda(v, w) = \lambda'(v, w)\left(\mathcal{R}(v) + \sum_{(u,v)\in\mathbb{E}} \Lambda(u, v) + (\mathcal{F}_\alpha - \text{Vol}(\alpha))\right). \tag{8}$$

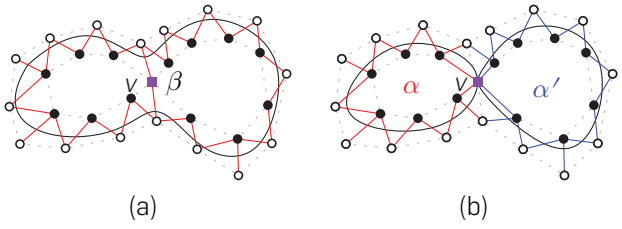
If neither α nor α' is full, add them both to the set of active depressions.

THEOREM 4.1. *Given a triangulation of $\mathbb{M}$ of $\mathbb{R}^2$ with n vertices, a height function $h : \mathbb{M} \to \mathbb{R}$ that is linear on each face of $\mathbb{M}$, a rain distribution $\mathcal{R}$, and a volume of rain ψ, the flooded vertices of $\mathbb{M}$ can be computed in $O(n \log n)$ time.*

5. POINT-FLOOD QUERIES

Given a rain distribution $\mathcal{R}$, a query point $q \in \mathbb{M}$, and a rain volume ψ, the point-flood query asks whether the point q is flooded if a volume ψ rain falls with distribution $\mathcal{R}$. Of course, the terrain-flood query procedure described in Section 4 can answer this query, but our goal is to answer this query more

Figure 5. (a) A single active depression β with active edges marked in red. (b) Saddle vertex v (marked in purple) delimits two maximal depressions α and α'. The active edges are partitioned into two sets: those crossing into α (resp. α') marked in red (resp. blue); the edges connecting to v in (a) are no longer active (corresponding to upslope edges), and downslope edges from v are now active and partitioned accordingly.



efficiently. We first discuss an algorithm for the MFD model and then describe how the running time can be further improved for the SFD model. We also briefly discuss a variant of the point-flood query, the *flood-time query*, which asks how much it must rain before a query point $q \in \mathbb{M}$ is flooded.

5.1. Point-flood query under MFD

Under the MFD model, the algorithm exploits two observations. First, we need not compute the fill volume of all maximal depressions. In particular, suppose q lies in a maximal depression β and there are two children depressions β_1 and β_2 of the sibling depression β' of β. We do not have to compute the fill volume of β_1 and β_2. It suffices to compute if β' fills and how much, if any, water spills from β' to the depression β_q. In fact, the only depressions we need to consider are the tributaries of q. Second, we introduce the notion of tributary graph that describes how water flows between the tributaries of q. The tributary graph is used to quickly compute the fill and spill volumes of the tributaries of q.

Using these ideas, we describe an $O(nm)$-size data structure for the MFD model that answers a point-flood query in $O(|\mathcal{R}|k + k^2)$ time, where m is the number of sinks in $\mathbb{M}$ and k is the number of tributaries of q.

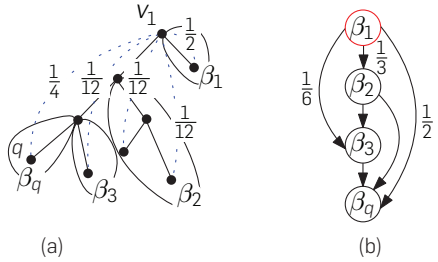
Tributary graph. For a point $q \in \mathbb{M}$, the tributary graph $\mathsf{G}[q] = (\mathsf{X}_q, \mathsf{E}_q)$ is a directed acyclic graph. X_q is the depression β_q plus its tributaries. For a pair of depressions $\alpha, \beta \in \mathsf{X}_q$, we add the directed edge (α, β) to E_q if $h(\mathrm{Sd}(\alpha)) > h(\mathrm{Sd}(\beta))$ and $\phi_\alpha(\mathrm{Sd}(\alpha), \beta) > 0$ (if $\beta = \beta_q$; then, by $\mathrm{Sd}(\beta_q)$, we mean q.) Recalling that ϕ_α is the flow function when the depression α is flooded, we set the weight of the edge (α, β) to be $\phi_\alpha(\mathrm{Sd}(\alpha), \beta)$. For the MFD model, this can be computed as

$$w(\alpha, \beta) = \frac{\phi(\mathrm{Sd}(\alpha), \beta)}{\sum_{(\alpha,\gamma)\in\mathsf{E}_q} \phi(\mathrm{Sd}(\alpha), \gamma)}, \tag{9}$$

that is, the weights are normalized so that the weighted outdegree of each node in $\mathsf{G}[q]$ is 1. See Figure 6 for an example.

Overview of algorithm. It is expensive to compute the fill and spill volume functions $\mathcal{F}_\beta$ and $\mathcal{S}_\beta$ exactly for each tributary of q, so we define slightly different functions $\tilde{\mathcal{F}}_\beta, \tilde{\mathcal{S}}_\beta : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ for all $\beta \in \mathsf{X}_q$. They are fill, spill volume functions under the assumption that every tributary of β_q fills before its sibling, that is, water spills from a tributary β to various sinks in the sibling β' of β; note that β' is a depression containing q.

Figure 60 (a) A merge tree T, with tributaries (β_1, β_2, β_3) of q delimited and flow from v_1 to each sink s, $\phi(v_1, s)$, marked. (b) Tributary graph G[q], with the edge weights from β_1.



We define $\tilde{\mathcal{F}}_\beta, \tilde{\mathcal{S}}_\beta$ recursively using the tributary graph $\mathsf{G}[q]$ as follows:

$$\begin{aligned} \tilde{\mathcal{F}}_\beta &= F_\beta(0)\psi + \sum_{(\alpha,\beta)\in\mathsf{E}_q} \tilde{\mathcal{S}}_\alpha w(\alpha, \beta), \\ \tilde{\mathcal{S}}_\beta &= \max\{0, \tilde{\mathcal{F}}_\beta - \mathrm{Vol}(\beta)\}. \end{aligned} \tag{10}$$

For any given ψ, $\tilde{\mathcal{F}}_{\beta_q} < \mathrm{Vol}(\beta_q)$ if and only if $\mathcal{F}_{\beta_q} < \mathrm{Vol}(\beta_q)$. So the point-flood query can be answered by computing $\tilde{\mathcal{F}}_{\beta_q}$ and returning yes if this quantity is at least $\mathrm{Vol}(\beta_q)$.

Preprocessing step. In the preprocessing step, we construct the merge tree T and preprocess it so that for a point $q \in \mathbb{M}$, (i) the edge of T containing q and (ii) $\mathrm{Vol}(\beta_q)$ can be computed in $O(\log n)$ time.

Additionally, for each vertex $v \in \mathbb{M}$ and for each of $O(m)$ maximal depressions β, we store the value of $\phi(v, \beta)$. Actually, we need to store only nonzero values; in practice, the number of such pairs is much smaller than mn.

Preprocessing takes $O(n \log n + nm)$ time, and the size of the data structure is $O(nm)$.

Query procedure. For a query rain distribution $\mathcal{R}$ and a query point, we first find the edge e of T containing q and $\mathrm{Vol}(\beta_q)$. Given e, we compute the tributaries of q in $O(k)$ time by traversing T upward from e. We now construct the tributary graph $\mathsf{G}[q] = (\mathsf{X}_q, \mathsf{E}_q)$ in $O(k^2)$ time, using the precomputed values of $\phi(v, \beta)$ in (9).

To compute $\tilde{\mathcal{F}}_q$ for all depressions $\beta \in \mathsf{X}_q$, we first compute $F_\beta(0)$ for each $\beta \in \mathsf{X}_q$ using the formula

$$F_\beta(0) = \sum_{u:\mathcal{R}(u)>0} \mathcal{R}(u)\phi(u, \beta)$$

and then use the recurrence 10. The total time spent by the query procedure is $O(|\mathcal{R}|k + k^2)$. Putting everything together, we obtain the following:

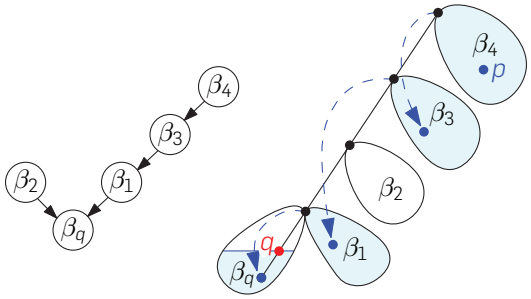
THEOREM 5.1. *Given a triangulation of $\mathbb{M}$ of $\mathbb{R}^2$ with n vertices, a height function $h : \mathbb{M} \to \mathbb{R}$ that is linear on each face of $\mathbb{M}$, a data structure of size $O(mn)$ can be constructed in $O(n \log n + mn)$ time so that a point-flood query can be answered in $O(|\mathcal{R}|k + k^2)$ time, where $|\mathcal{R}|$ is the complexity of the query rain distribution, k is the number of maximal depressions containing the query point, and m is the number of sinks in the terrain $(\mathbb{M}, h)$.*

5.2. Point-flood query under SFD

If the water flows according to an SFD model, point-flood queries can be answered even more efficiently. Under the SFD model, a key observation is that the tributary graph $\mathsf{G}[q]$ is a tree because each tributary has out degree 1; see Figure 7. To see why, note that for each vertex v, water flows from v to exactly one sink γ. Importantly, each tributary α upon becoming full will spill to a single sink γ, that is, $\phi_\alpha(\mathrm{Sd}(\alpha), \gamma) = 1$, and for all other sinks, this will be 0. Letting β be the tributary containing γ, the edge (α, β) will have unit weight in $\mathsf{G}[q]$, and there will be no other edges from α.

Single-point source. First consider the case when rain falls only at a single point p (contained in tributary, say β, of q). As $\mathsf{G}[q]$ is a tree, there is a unique path π from β to β_q. When a tributary becomes full, the water begins spilling to the next

Figure 7. Left: G[q] is a tree under the SFD model. Right: When rain falls at $p \in \beta_4$, the tributaries along the path from β_4 to β_q in G[q] fill and spill in order.



tributary in π. For q to become flooded, all the tributaries in π must be flooded. We can answer the point-flood query by simply following the tributaries π, pushing excess water to the next tributary until either q is flooded or we come to a tributary that does not get filled. See Figure 7. However, if q has k tributaries, this algorithm takes $O(k)$ time, and in the worst case, $k = \Omega(n)$.

The query can be expedited using a well-known data structure called a *heavy-path tree decomposition* on T. In this data structure, T is partitioned into *heavy*-paths, such that every path intersects $O(\log n)$ heavy-paths. By precomputing prefix sums of tributary volumes along each heavy-path, we can process in amortized $O(1)$ time all the tributaries in π intersecting a given heavy-path. Therefore, point-flood queries for a single-point source can be answered in $O(\log n)$ time. We can again use the heavy-path decomposition data structure to add the volumes of tributaries, but the running time now depends on the number of tributaries in which rain is falling.

Region source. This approach can be extended to work for rain falling in multiple tributaries. When paths from two of these tributaries intersect, both spilling into a common tributary, we simply add the spill volume from both. See Figure 8.

THEOREM 5.2. *Given a triangulation of $\mathbb{M}$ of $\mathbb{R}^2$ with n vertices, 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 $O(n \log n)$ time so that a point-flood query under the SFD model can be answered in $O(|\mathcal{R}| + k \log n)$ time, where $|\mathcal{R}|$ is the complexity of the query rain distribution and k is the number of tributaries of q on which it is raining.*

5.3. Flood-time query

Given a rain distribution $\mathcal{R}$ and a query point $q \in \mathbb{M}$, we can also ask how much it must rain before q becomes flooded. Now, instead of just maintaining the spill and fill volumes for a fixed volume of rain ψ, we must maintain the functions for spill and fill rates as defined in (2) and (5). Under the SFD model, the algorithm described above can be extended to answer the flood-time queries without increasing the space or time complexity. The main idea is that given the rates at the predecessors of a node α in G[q], the

Figure 8. Top: Under the SFD model, water spilling from two tributaries intersects at a single downstream tributary. Bottom: Under the MFD model, (a fraction of) water spilling from two tributaries may intersect at many downstream tributaries.

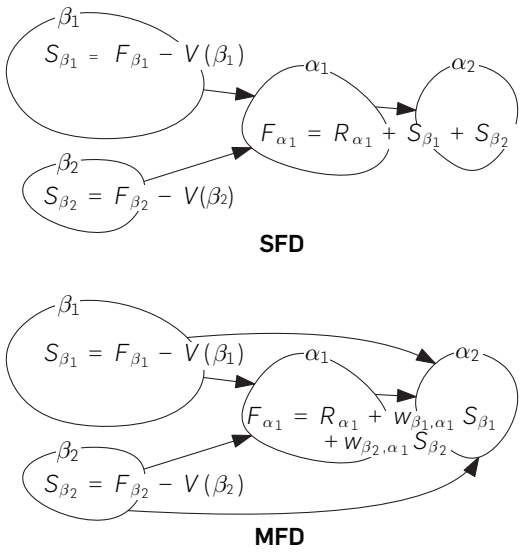
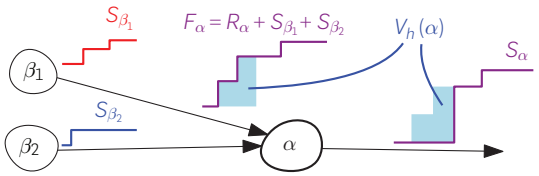


Figure 9. The fill rate F_α is computed from the spill rates S_{β_1} and S_{β_2}; the spill rate S_α is computed from F_α.



fill and spill rates at α can be computed in $O(\log n)$ amortized time. See Figure 9.

However, under the MFD model, computing spill and fill rates becomes significantly more complex as the water spilling from a tributary may split and fill multiple downstream tributaries. We can, however, do better than the naive approach adapting ideas from the algorithm for the SFD model. The main idea is that by using matrix multiplication, the downstream effect of spilling from multiple tributaries can be computed in a single step. If the product of two $k \times k$ matrices can be computed in $O(k^\omega)$ time, a flood-time query can be answered in $O(|\mathcal{R}|k + k^\omega + k^2 \log n)$, where $|\mathcal{R}|$ is the complexity of the query rain distribution and k is the number of tributaries of q.

6. EXPERIMENTS

In this section, we present experiments we have conducted on real terrain data to demonstrate the efficiency of these algorithms and compare qualitatively the flooding under SFD and MFD models.

We have implemented the terrain-flood algorithm, described in Section 4, in C++, as well as a version of the point-flood algorithm.

We study the performance of the algorithms on three publicly available grid DEMs:

- The *Indiana dataset*, a 0.89 mi^2 model of a suburban area 0.5 mi northeast of Holland, IN, USA
- The *Philadelphia dataset*, a 225 km^2 model of an urban area in the northwest area of Philadelphia, PA, USA
- The *Norway dataset*, a 10,000 km^2 model of a mountainous region located in the Jotunheimen National Park, Norway

For the SFD model, water flows from a vertex to its lowest neighbor. For the MFD model, water flows from a vertex to all downslope neighbors with the relative rates proportional to the gradient of the slope.

SFD vs. MFD flooding. We considered the rain distribution $\mathcal{R}$ to be rain falling: (i) on a single vertex or (ii) uniformly over a region.

Our experiments show that when rain is falling at a single point, the areas that are flooded under the SFD and MFD models can be quite different. Under the MFD model, some large areas may become flooded that would not under the SFD model (Figure 10). As we increase the region on which rain is falling, we still see differences in the areas flooded, although they may be less pronounced (Figure 11). For example, the same general regions may be flooded, but under the MFD model, more water might end up in one location as opposed to that in SFD model, or water may reach more depressions. When we expand the rain distribution to be falling over the whole terrain, the regions that are flooded tend to be very similar.

Another difference in the two models, irrespective of the area of the region where rain is falling, is how water flows over the terrain. Under the SFD model, water flows along disjoint paths, whereas under the MFD model, it spreads more on the terrain (Figures 10 and 11). We illustrate these observations here with a few examples.

For the case when rain falls at a single point, we computed the flooded areas with rain volume of 10^5 m^3 on the Indiana dataset and 10^7 m^3 on the Norway dataset. Figure 10 shows the terrain-flood query for two single point rain distributions under both the SFD and MFD models. In Figure 10(a), we see that under the SFD model water follows a single path from p north east, first filling a large region before spilling and filling a series of smaller regions as the water flows west toward a feature corresponding to a river. In Figure 10(b), under the MFD model, the water splits at p and fills a number of depressions to the south west of p. We additionally note that under the MFD model in (b) water spreads out more and flows across a wider path between full depressions.

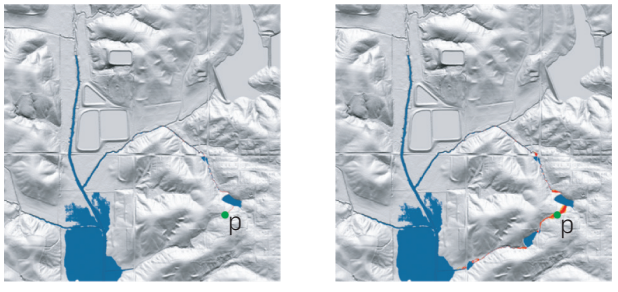
For the case where rain is falling uniformly over a small square, we set the rain distribution to be uniform over a square of size 1 km × 1 km and set $\psi = 10^6$ m^3 and computed the flooded area for the Norway dataset, under both SFD and MFD models. Figure 11 shows the queries along with enlarged images of the region on which it is raining. We see that although similar areas become flooded, water spreads out across the peak more under the MFD model, and a larger fraction of the water flows to the southwest. In contrast, under the SFD model, the rain follows narrow bands outside of the rain region, and a larger fraction of the water flows to the north.

Performance of algorithms. Building the merge tree and preprocessing it to compute the depression volume of any point as well as to perform lowest common ancestor queries took an average of 1.33 s over 5 trials for the Indiana dataset containing 10^6 vertices.

We also ran tests over the Philadelphia dataset, taking subregions with 1.6×10^7, 9×10^6 and 10^6 vertices, and on the Norway dataset. Our experiments show that in practice, the preprocessing time is near linear in the size of the terrain. Although preprocessing does require sorting the nodes, which takes $O(n \log n)$ time in the worst case, in practice, the constant on this term is much smaller than that of the linear steps in the preprocessing.

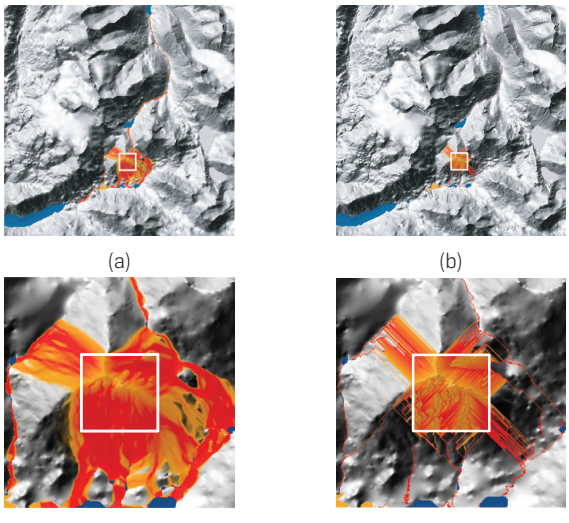
We examined the running time of the terrain-flood query on the Indiana dataset as we change the amount of

Figure 10. 10^5 m^3 (resp. 10^7 m^3) of rain falls at p in the Indiana dataset. The flooded regions of Σ are marked in blue, with regions that water flows over marked in red. Water flows according to SFD in (a) and MFD in (b).



(a) (b)

Figure 11. 100 m of rain falls uniformly over the square outlined in white in the Norway dataset. (a) Water flows according to MFD, (b) water flows according to SFD. (c) and (d) show a 3 km × 3 km area around the region it is raining on.



(a) (b) (c) (d)

rain ψ. Increasing the volume of rain first increases the running time until it reaches a peak and then decreases, becoming very fast with the largest volumes of rain. When a small amount of rain is falling in a small area, water reaches very few depressions and thus only a small portion of the merge tree is explored by the algorithm. As the volume and area of rain increases, the algorithm explores larger portions of the merge tree leading to an increase in the running time. However, once the volume of rain increases further, large depressions get filled and the algorithm succeeds in pruning large portions of the merge tree. Roughly speaking, the running time of the algorithm is proportional to the number of depressions that are partially filled.

Finally, we examined the running time of the terrain-flood query on the Indiana and Philadelphia datasets as we change the number of points with positive rain in $\mathcal{R}$, raining uniformly over squares of varying sizes. Although the running time increases with the number of vertices with positive rainfall, it grows slower than the linear dependence indicated by the worst-case time analysis.

7. RELATED WORK

The flood-risk problem has been widely studied in multiple research communities, and many different approaches have been proposed to tackle this problem. One such approach, coming from the hydrology community, simulates fluid dynamics, using nonlinear partial differential equations such as the Navier-Stokes equations.[7, 14] They often account for additional factors, such as the effects of different terrain types and drainage networks. Although these models tend to be the most accurate, naive applications are computationally expensive and many techniques such as multiresolution representation of the terrain and approximate computation have been proposed to expedite the computation.[7, 14, 25] Not withstanding much work on to reducing the computational cost of these methods, these algorithms are hard to scale to large terrains.

Recently, machine-learning–based approaches have been proposed for predicting flood risk.[10, 24] These approaches are relatively fast, while maintaining a reasonable level of predictive power. However, these methods generally are used for predicting fluvial floods (i.e., flooding of rivers) and rely on stream gauges or other sensors already being installed in the area of interest. Tehrany et al.[24] tested the efficacy of various support vector machine (SVM) kernels at predicting the overall flood hazard of points on a terrain using historical flood events and a number of terrain features such as slope, altitude, surface runoff, and distance from a river. Chang et al.[10] used self-organizing maps and neural networks to forecast the flood inundation in the near future (1–3 h) given the current inundated areas.

There has been extensive work on modeling water flow on a terrain in the GIS community.[2–6, 9, 11, 15, 17, 19–21] These approaches use simpler models, focusing on the geometry of the terrain. These tend to be more computationally efficient and suitable for large datasets. However, the simplifying assumptions mean that they may not be as accurate as PDE-based models in all situations. For example, they do not take into account absorption of water into the ground and are thus more suitable for flash floods wherein most of the flooding occurs over a shorter timespan. Liu and Snoeyink[17] (see also Arge et al.[6]) proposed an $O(n \log n)$-time algorithm under the single-flow direction (SFD) model that computes the fill times of all depressions assuming rain is falling at a constant rate on the entire terrain.

Arge et al.[4] described an $O(n \log n)$-time algorithm, under the SFD model, to compute the set of flooded vertices when a given volume of rain $\psi \geq 0$ falls on a given region of the terrain.

8. CONCLUSION

In this paper, we have presented efficient algorithms for a few flood-risk queries: the *terrain-flood* query that asks which vertices of a terrain will be flooded and the *point-flood* query that asks if a given point will be flooded. The work is only a small step toward performing flood-risk analysis in real time over a large terrain, and there are many open questions:

Can these algorithms be extended to incorporate uncertainty in data as well as in the model? There have been some preliminary results for uncertainty of terrain height under the SFD model,[21] but this problem remains largely open.

The flooding model described in this paper depends only on the elevation of the terrain data. In reality, there are other factors that affect flooding such as terrain type and permeability as well as drainage networks. Can a model be developed that incorporates additional terrain data? Furthermore, can historical flood data be incorporated into this model to more accurately compute flood risk?

The flooding model described here also assumes that water flows only along edges of the terrain and that water flows instantaneously to the sinks. Although these assumptions are reasonable in some settings (e.g., flash floods and high-resolution terrain models), can a model be developed that incorporates the velocity of the water and allows for channel flow?

Acknowledgments

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. Work by Rav is partially supported by the Innovation Fund Denmark. C

References

1. 3Di Water Management. https://3diwatermanagement.com, 2019
2. Agarwal, P.K., Arge, L., Yi, K. I/O-efficient batched union-find and its applications to terrain analysis. In *Proceedings of the 22nd Annual Symposium on Computational Geometry* (2006), 167–176.
3. Arge, L., Chase, J., Halpin, P., Toma, L., Vitter, J., Urban, D., Wickremesinghe, R. Efficient flow computation on massive grid terrain datasets. *GeoInformatica 4*, 7 (2003), 283–313
4. Arge, L., Rav, M., Raza, S., Revsbæk, M. I/o-efficient event based depression flood risk. In *Proceedings of the 19th Workshop on Algorithm Engineering and Experiments* (2017), 259–269.
5. Arge, L., Revsbæk, M. I/O-efficient contour tree simplification. In *International Symposium on*

Algorithms and Computation (2009), 1155–1165.
6. Arge, L., Revsbæk, M., Zeh, N. I/O-efficient computation of water flow across a terrain. In *Proceedings of the 26th Annual Symposium on Computational Geometry* (2010), 403–412.
7. Bates, P.D., De Roo, A. A simple raster-based model for flood inundation simulation. *J. Hydrol. 1-2*, 236 (2000), 54–77.
8. Carr, H., Snoeyink, J., Axen, U. Computing contour trees in all dimensions. *Comput. Geomet. 2*, 24 (2003), 75–94.
9. Carr, H., Snoeyink, J., Panne, M. Flexible isosurfaces: Simplifying and displaying scalar topology using the contour tree. *Comput. Geomet. 1*, 43 (2010), 42–58.
10. Chang, L.-C., Shen, H.-Y., Chang, F.-J. Regional flood inundation nowcast using hybrid som and dynamic neural networks. *J. Hydrol.*, 519 (2014), 476–489.
11. Danner, A., Mølhave, T., Yi, K., Agarwal, P., Arge, L., Mitásová, H. Terrastream: From elevation data to watershed hierarchies. In *Proceedings of the 15th Annual ACM International Symposium on Advances in GIS* (2007), 28.
12. Edelsbrunner, H., Harer, J., Zomorodian, A. Hierarchical morse complexes for piecewise linear 2-manifolds. In *Proceedings of the 17th Annual Symposium on Computational Geometry* (2001), 70–79.
13. Fathom Global. https://www.fathom.global/fathom-global, 2019.
14. Ghimire, B., Chen, A.S., Guidolin, M., Keedwell, E.C., Djordjević, S., Savić, D.A. Formulation of a fast 2d urban pluvial flood model using a cellular automata approach. *J. Hydroinform. 3*, 15 (2013), 676–686.
15. Jenson, S., Domingue, J. Extracting topographic structure from digital elevation data for geographic information system analysis. *Photogramm. Eng. Rem. Sens. 11*, 54 (1988), 1593–1600.
16. Kreveld, M., Oostrum, R., Bajaj, C., Pascucci, V., Schikore, D. Contour trees and small seed sets for isosurface traversal. In *Proceedings of the 13th Annual Symposium on Computational Geometry* (1997), 212–220.
17. Liu, Y., Snoeyink, J. Flooding triangulated terrain. In *Proceedings of the 11th International Symposium on Spatial Data Handling* (2005), 137–148.
18. Lowe, A., Agarwal, P.K. Flood-risk analysis on terrains under the multiflow-direction model. In *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, (ACM, 2018), 53–62.
19. O'Callaghan, J., Mark, D. The extraction of drainage networks from digital elevation data. *Comp. Vis. Graph. Image Process. 3*, 28 (1984), 323–344.
20. Quinn, P., Beven, K., Chevallier, P., Planchon, O. The prediction of hillslope flow paths for distributed hydrological modelling using digital terrain models. *Hydrol. Process. 1*, 5 (1991), 59–79.
21. Rav, M., Lowe, A., Agarwal, P. Flood risk analysis on terrains. In *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in GIS* (ACM, 2017), 36.
22. SCALGO. www.scalgo.com, 2019.
23. Tarasov, S., Vyalyi, M. Construction of contour trees in 3d in o(n log n) steps. In *Proceedings of the 14th Annual Symposium on Computational Geometry* (1998), 68–75.
24. Tehrany, M.S., Pradhan, B., Mansor, S., Ahmad, N. Flood susceptibility assessment using gis-based support vector machine model with different kernel types. *Catena*, 125 (2015), 91–101.
25. Volp, N., Van Prooijen, B., Stelling, G. A finite volume approach for shallow water flow accounting for high-resolution bathymetry and roughness data. *Water Resour. Res. 7*, 49 (2013), 4126–4135.

Aaron Lowe and Pankaj K. Agarwal ([aaron, pankaj]@cs.duke.edu), Duke University, Durham, NC, USA.

Mathias Rav (mathias@scalgo.com), SCALGO, Aarhus, Denmark.