Computing representative networks for braided rivers

Kleinhans, M.; van Kreveld, M.J.; Ophelders, T.A.E.; Sonke, W.M.; Speckmann, B.; Verbeek, K.A.B.

Published in: Proc. 33rd International Symposium on Computational Geometry (SoCG)

DOI: 10.4230/LIPIcs.SoCG.2017.48

Published: 01/06/2017

Document Version
Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

- A submitted manuscript is the author's version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

Citation for published version (APA):

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain.
- You may freely distribute the URL identifying the publication in the public portal.

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.
Computing Representative Networks for Braided Rivers

Maarten Kleinhans¹, Marc van Kreveld², Tim Ophelders³, Willem Sonke⁴, Bettina Speckmann⁵, and Kevin Verbeek⁶

1 Faculty of Geosciences, Utrecht University, Utrecht, The Netherlands
m.g.kleinhans@uu.nl
2 Dept. of Information and Computing Sciences, Utrecht University, Utrecht, The Netherlands
m.j.vankreveld@uu.nl
t.a.e.ophelders@tue.nl
w.m.sonke@tue.nl
5 Dept. of Mathematics and Computer Science, TU Eindhoven, Eindhoven, The Netherlands
b.speckmann@tue.nl
k.a.b.verbeek@tue.nl

Abstract
Drainage networks on terrains have been studied extensively from an algorithmic perspective. However, in drainage networks water flow cannot bifurcate and hence they do not model braided rivers (multiple channels which split and join, separated by sediment bars). We initiate the algorithmic study of braided rivers by employing the descending quasi Morse-Smale complex on the river bed (a polyhedral terrain), and extending it with a certain ordering of bars from the one river bank to the other. This allows us to compute a graph that models a representative channel network, consisting of lowest paths. To ensure that channels in this network are sufficiently different we define a sand function that represents the volume of sediment separating them. We show that in general the problem of computing a maximum network of non-crossing channels which are \( \delta \)-different from each other (as measured by the sand function) is NP-hard. However, using our ordering between the river banks, we can compute a maximum \( \delta \)-different network that respects this order in polynomial time. We implemented our approach and applied it to simulated and real-world braided rivers.

1998 ACM Subject Classification F.2.2 Analysis of Algorithms and Problem Complexity

Keywords and phrases Braided rivers, Morse-Smale complex, persistence, network extraction, polyhedral terrain

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.48

© Maarten Kleinhans, Marc van Kreveld, Tim Ophelders, Willem Sonke, Bettina Speckmann, Kevin Verbeek; licensed under Creative Commons License CC-BY


Leibniz International Proceedings in Informatics Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany
Geomorphology is the study of the shape of natural terrains and the processes that create them. One of these processes is erosion due to water flow. In mountainous areas, valleys are nearly always shaped by rivers, which slowly transport solid material downstream. Due to gravity, water usually follows the direction of steepest descent, although inertia may result in deviations. The combination of terrain shape and water flow gives rise to various computational problems that have been studied in geomorphology, geographic information science (GIS), and computational geometry.

One prominent problem is the computation of drainage networks, also referred to as flows. Here computations are based on elevation data only and the shape of the terrain is used to determine where rivers will form (see an extensive overview). A second problem of interest concerns local minima. Due to erosion local minima are more rare in natural terrains than local maxima. Minima in terrains are the bottoms of pits that will be filled with water. When a pit overflows at the lowest saddle surrounding it, the saddle may be eroded and eventually both pit and saddle will be removed simultaneously. For this reason certain terrain types do not have local minima and minor local minima are often measurement errors. Such errors are clearly undesirable when studying flow on terrains and hence, minor local minima are removed by computational means. A third commonly studied problem deals with watersheds. Watershed boundaries are steepest ascent paths following the ridges of mountains.

Braided rivers. The usual models (as discussed above) for water flow in terrains allow rivers to merge, which is natural because side valleys join main valleys. And clearly, if water always follows the direction of steepest descent, a river cannot split (except due to degeneracies). Yet splitting happens in deltas and various river types, in particular braided river systems (see Figure 1 for an example). Such systems have islands called bars ranging in length from about one water depth to ten times the overall river width, separating different channels of the same river over their length after which the channels confluence. Bifurcations are currently still poorly understood in geomorphology. They are quite dynamic in the sense that erosion or sedimentation leads to perpetual changes in the division of water and sediment at bifurcations, which affects downstream morphological development of channels and bars locally and over surprisingly long distances downstream. This makes braided river systems highly dynamic on the seasonal time scale. Within a year the appearance and hydrological functioning is also highly variable due to changing water levels during floods and low flow periods that emerge and submerge bars.

Many fundamental questions remain how rivers form shallow, ecologically valuable shoals, carve deep channels for shipping, collapse banks with property on it, and flood built-up areas. Part of the reason for slow progress is that sediment transport is a nonlinear function of flow velocity, leading to self-amplifying changes in depth and width of channels. This leads to changes in water and sediment conveyance, which in turn affects downstream channels that split around other bars and so on ad nearly infinitum. We know this quasi-quantitatively from increasing gridded data of bed elevation and flow properties collected with advanced remote sensing techniques, in numerical two-phase modelling and in controlled laboratory experiments with small-scale braided rivers. However, how small changes at

---

1 We use ‘rivers’ as unifying terminology for flowing water, including erosive rivulets, streams, brooks, and large sediment-laden rivers that build their own landscapes through sedimentation.
bifurcations and confluences propagate and grow or decay through the channel network and how that leads to changes in the larger-scale network structure remains unknown. A main reason is that there is currently no technique to extract morphologically meaningful networks, nor to rigorously connect the networks at different water levels [12, 18].

Modeling braided rivers, where channels can both split and merge, is considerably more complex than modeling standard drainage networks, where all rivers flow only downhill and do not bifurcate. In this paper we initiate the study of braided rivers from the perspective of computational geometry and topology.

**Modeling braided rivers in a nutshell.** To model a braided river we first need a representation of the basic geometry, independent of water level. We hence use the elevation of the river bed as a starting point. In meandering rivers the so-called *thalweg* is often used as a basic representation of the river. The thalweg is defined as the deepest part of a continuous channel. In a meandering river that is simply one linear feature, which is usually not the channel center line but more sinuous as it follows the deeper pools in the outer bends. We are striving for a similar representation for braided rivers, consisting of linear features along *lowest paths* in each channel. These linear features can merge and bifurcate, that is, they form a planar graph or network. The use of graphs to model and analyze braided rivers was recently pioneered in [18]. In a river, channels are deepest where water flows (or used to flow) the fastest. This is due to the non-linearity of sediment transport: when velocity doubles, sediment transport increases eight to perhaps hundredfold, which leads to the erosion of channels and deposition in bars. We define lowest paths intuitively as the paths that do not go higher than they need to go to connect the endpoints. To find lowest paths we use a descending quasi Morse-Smale complex [10, 24]. We show that lowest paths must always lie on this complex, except for, possibly, the ends of the path.

A representative network for a braided river should not necessarily contain all possible channels. Topologically speaking a tiny local maximum in the river bed creates two channels. We can use persistence to simplify our input and avoid such situations. But still, too many channels might remain. We would hence like to select a set of channels which are sufficiently different from each other. We model how different two channels are with a function (the *sand function*) that relates to the volume of sediment the river has to move before the two channels become one. More volume needs more time to be removed [13]. A bar of very small volume separating two channels requires insignificant time to be removed, so the channels are not significantly different. But a large bar with a large volume may require multiple...
floods to be shaved off or cut through by a new channel, meaning that the two channels separated by this bar are significantly different.

Our objective now is to compute a representative network of channels that is optimal in some sense. We require (i) each channel to be (mostly) on the descending quasi Morse-Smale complex (that is, locally lowest), (ii) any two channels to be sufficiently different (specified by a parameter \( \delta \) and the sand function), and (iii) the representative network to be maximum. Unfortunately, the exact formulation of this problem is NP-hard. We deal with this by computing a striation: a left-bank-to-right-bank sequence of non-crossing paths for the whole braided river. We provide three different heuristics to compute a striation, each with various strengths and weaknesses. We then limit the representative network to select channels only from the striation. To make this selection, we compute the sand function between every two channels. For this we compute a monotone isotopy between any two channels, to ensure that only sand between the channels is measured, and without multiplicity. We require this isotopy to be consistent with the striation. We then present three different models to compute the sand function along a single matching path of the isotopy, which is then integrated over the entire isotopy to compute the sand function between two channels. The isotopy that minimizes the sand function is chosen. We show that the resulting sand function can be computed efficiently between any two channels of the striation. Finally, we can use a simple greedy algorithm to select the representative network from the striation.

Organization. Section 2 gives the definitions and the problem statement. Section 3 describes the quasi Morse-Smale complex and shows that lowest paths lie mostly on this complex. Section 4 shows NP-hardness of computing a representative network and introduces three ways of computing a striation. Section 5 gives three ways of defining the sand function and algorithms to compute it. Section 6 describes how to compute a representative network if a striation and a sand function are given. Section 7 reports on experiments of an implementation applied to data from a numerical model and to data collected from the real-world Waimakariri river (see Fig. 1). Finally, a discussion is given in Section 8.

## 2 Definitions and problem statement

Let \( G = (V, E) \) be a triangulation of a topological disk \( M \) in the plane, and let \( h: M \to \mathbb{R} \) be the height of the points in \( M \), where the edges (respectively triangles) of \( G \) interpolate \( h \) linearly between its vertices (respectively edges). So \( G \) can be viewed as a simplicial 2-complex in \( \mathbb{R}^3 \) by adding \( h \) as a third dimension. Furthermore, let \( \sigma \in V \) be a source and \( \tau \in V \) be a sink, both on the boundary \( \partial M \). The source and sink are assumed to be vertices for simplicity, but our approach can be extended to the case where \( \sigma \) and \( \tau \) are connected components on the boundary of \( M \). We refer to \( (G, h, \sigma, \tau) \) as a river, and we refer to the volume \( \{(x, y, z) \mid (x, y) \in M, z \in \mathbb{R}, z \leq h(x, y)\} \) as sand. We define the amount of sand above a point \( (x, y, z) \) as \( \max(0, h(x, y) - z) \).

Let \( \pi^+_{\partial M} \) and \( \pi^-_{\partial M} \) (respectively clockwise and counterclockwise) be the two paths from \( \sigma \) to \( \tau \) along the boundary of \( M \). We call a path \( \pi \) from \( \sigma \) to \( \tau \) semi-simple if it has no self-intersections, but it may coincide with itself, see the red path in Fig. 2. In such self-coinciding parts, these parts are symbolically separated. Let \( P \) be a set of semi-simple, pairwise non-crossing paths from \( \sigma \) to \( \tau \) along edges of \( G \). For two semi-simple paths \( \pi_0 \) and \( \pi_1 \) in \( P \), let \( D(\pi_0, \pi_1) \) be the region bounded by and including \( \pi_0 \) and \( \pi_1 \). So two semi-simple paths \( \pi_0 \) and \( \pi_1 \) from \( \sigma \) to \( \tau \) have no proper crossings if and only if \( D(\pi^+_{\partial M}, \pi_0) \subseteq D(\pi^+_{\partial M}, \pi_1) \) or \( D(\pi^-_{\partial M}, \pi_1) \subseteq D(\pi^-_{\partial M}, \pi_0) \).
A homotopy \( \eta : [0,1]^2 \to M \) from \( \pi_0 \) to \( \pi_1 \) is a continuous map, such that \( \eta(p,0) = \pi_0(\alpha_\eta(p)), \eta(p,1) = \pi_1(\beta_\eta(p)), \eta(0,t) = \sigma \) and \( \eta(1,t) = \tau \), where reparameterizations \( \alpha_\eta \) and \( \beta_\eta : [0,1] \to [0,1] \) are continuous non-decreasing surjections aligning \( \pi_0 \) and \( \pi_1 \). We refer to \( (\alpha_\eta, \beta_\eta) \) as the matching \( \pi_0 \) and \( \pi_1 \) given by \( \eta \). We can equip a homotopy \( \eta \) with a height function \( \zeta : [0,1]^2 \to \mathbb{R} \) and define the surface \( \Sigma_\eta : [0,1]^2 \to \mathbb{R}^3 \) as \( \Sigma_\eta(p,t) = (\eta(p,t), \zeta(p,t)) \). Define the volume above \( \Sigma_\eta \) as the total volume of sand above the points of \( \Sigma_\eta \) (counted with multiplicity) given by

\[
vol(\Sigma_\eta) = \int \int_{[0,1] \times [0,1]} \max(0, h(\eta(p,t)) - \zeta(p,t)) \| \frac{\partial \eta}{\partial p} \times \frac{\partial \eta}{\partial t} \| \, dp \, dt.
\]

Generally, we will choose \( \eta(p,t) \) in such a way that it does not surpass the height of \( \eta(p,0) \) or \( \eta(p,1) \), so as to measure at least the volume of sand ‘above’ an extremal path (\( \pi_0 \) or \( \pi_1 \)).

We measure the similarity between two paths using a sand function \( d : \mathcal{P} \times \mathcal{P} \to \mathbb{R} \), and we say a path \( \pi_0 \) is \( \delta \)-dissimilar to \( \pi_1 \) if and only if \( d(\pi_0, \pi_1) \geq \delta \). The function \( d \) will generally not be a metric, since it is generally not symmetric, does not satisfy the triangle inequality, and \( d(\pi_0, \pi_1) \) can be 0 for distinct paths \( \pi_0 \) and \( \pi_1 \). Intuitively, we define \( d(\pi_0, \pi_1) \) in such a way that measures the volume of sand that lies between \( \pi_0 \) and \( \pi_1 \). Since the height of both paths may vary along the length of the river, it is unclear how to define this volume in a natural way. We define \( d(\pi_0, \pi_1) = vol(\Sigma_\eta) \), so the sand measured by \( d \) depends largely on the homotopy \( \eta \) and the corresponding height function \( \zeta \). We discuss how to choose \( \eta \) between \( \pi_0 \) and \( \pi_1 \) in Section 5.

Our goal is to compute a network whose paths represent channels in a river. Essentially, such paths minimize the distance spent at high elevations. We define the cost of a path as its lexicographic height [21]. For a path \( \pi : [0,1] \to M \), define \( \pi_h : [0,1] \to M \times \mathbb{R} \) to be the path \( (\pi(p), h(\pi(p))) \) over the terrain, and let \( \rho(\pi, z) \) be the length of the path \( \pi_h \) that has height at least \( z \). We say a path \( \pi_0 \) is lower than \( \pi_1 \) if and only if there exists a \( z^* \in \mathbb{R} \), such that for all \( z \geq z^* \), \( \rho(\pi_0, z) = \rho(\pi_1, z) \) and for all \( z > 0 \), there is some \( z' \in (z^* - \varepsilon, z^*) \) with \( \rho(\pi_0, z') < \rho(\pi_1, z') \). A path is lowest if no lower paths exist. We motivate this choice by the property of Lemma 2 that lowest paths follow steepest descent – a property often assumed for water flow as well [26]. Instead of stopping in local minima, lowest paths can proceed by taking a steepest descent from a saddle point in reverse (note that this is not a steepest ascent path from the minimum).

Call a subset \( \Pi \subseteq \mathcal{P} \) of paths a \( \delta \)-network if no pair of paths in \( \Pi \) has proper crossings, and \( d(\pi_0, \pi_1) \geq \delta \) if the lexicographic height of \( \pi_0 \in \Pi \) is at least that of \( \pi_1 \in \Pi \). Intuitively, a representative \( \delta \)-network is one that contains as many lowest paths as possible. More precisely,
if $\Pi$ and $\Pi'$ are $\delta$-networks, then $\Pi'$ is better than $\Pi$ if there exists some $k \leq \min\{|\Pi|+1,|\Pi'|\}$, such that for each $i < k$, the $i$-th lowest path of $\Pi$ and that of $\Pi'$ are equally low, and either $|\Pi| < k$, or the $k$-th lowest path of $\Pi'$ is lower than that of $\Pi$. A $\delta$-network is representative if no better $\delta$-networks exist.

Assumptions and problem statement. Assume that a polyhedral terrain is given: a triangulation with $n$ vertices that have a height, and linear interpolation over edges and triangles. We assume that all vertices have a different height and any two edges incident to the same vertex have different slope. The latter assumption ensures that steepest descent and steepest ascent over edges is unique from every vertex. We artificially connect all vertices of the boundary of the terrain to an extra vertex $v_\infty$ that is higher than all vertices on the boundary. All local minima on the boundary will stay local minima, whereas all local maxima on the boundary become regular. We do not need a geometric embedding for this modification.

Given such a modified terrain, a source $\sigma$, a sink $\tau$, and a difference parameter $\delta$, we study the problem of computing a representative $\delta$-network over the edges of the triangulation for various choices of the sand function $d$.

3 Morse-Smale complex and lowest paths

An important topological tool for computing a representative $\delta$-network, and lowest paths in particular, is the Morse-Smale complex. Here we briefly introduce some of the concepts related to (quasi) Morse-Smale complexes that are relevant to our work. We do so in a simplified manner to make the paper more accessible (full technical details are given in [10]). We furthermore specify the relation between Morse-Smale complexes and lowest paths.

3.1 Morse-Smale complex

Let $M$ be a smooth, compact 2-dimensional manifold without boundary, and let $h: M \to \mathbb{R}$ be a height function on $M$. A point $p$ on $M$ is critical with respect to $h$ if all partial derivatives vanish at $p$. Otherwise, $p$ is called regular. There are three types of critical points: (local) minima, (local) maxima, and saddle points. For each regular point $p$ we define the path of steepest ascent (or steepest descent) as the path that follows the gradient of $h$ at $p$. These paths are also known as integral lines and are open at both ends, with at each end a critical point. Using these integral lines, we can subdivide $M$ as follows: two regular points $p$ and $q$ belong to the same cell if the integral lines through $p$ and $q$ end at the same critical points on both sides. The resulting complex is known as the Morse-Smale complex, or MS-complex in short. Note that if one of the endpoints of an integral line is a saddle point, then the corresponding cell is 1-dimensional. We refer to 2-dimensional cells of the MS-complex as MS-cells, 1-dimensional cells as MS-edges, and the vertices simply correspond to the critical points. It can be shown ([10]) that every MS-cell is a quadrilateral with a minimum, a saddle, a maximum, and again a saddle along the boundary of the cell in that order.

Our input consists of a triangulation, and all paths must follow the edges of the triangulation. Therefore, the MS-complex as defined above is not directly useful for our purpose. Instead, we use a quasi MS-complex as defined in [10]. A quasi MS-complex has the same combinatorial structure as an ordinary MS-complex, but it can be required to follow the edges of a triangulation. Let $v$ be a vertex of the triangulation, and let $S(v)$ be the edge star of $v$ consisting of the set of edges incident to $v$. Note that, if we order the edges of $S(v)$ around $v$, then the endpoints of two neighboring edges must be connected by an edge in the triangulation. The lower edge star $S^\downarrow(v)$ consists of the subset of edges whose
endpoints are lower than \( v \) with respect to \( h \). Symmetrically, we can define the upper edge star \( S^\uparrow(v) = S(v) \setminus S^\downarrow(v) \) (recall that we have no horizontal edges). The lower edge star can naturally be subdivided into wedges of consecutive edges in \( S^\downarrow(v) \) separated by edges in \( S^\uparrow(v) \). We can now classify points based on the wedges in its lower edge star: minima have 0 wedges, regular points have 1 wedge but not the whole edge star, saddles have 2 or more wedges, and maxima have one complete wedge \( (S(v) = S^\downarrow(v)) \). Given these definitions, we can construct a quasi MS-complex as follows. From every saddle point \( v \) we construct a steepest descent path in every wedge of \( S^\downarrow(v) \) until it reaches a minimum. Similarly, we construct a steepest ascent path in every wedge of \( S^\uparrow(v) \) until it reaches a maximum. This construction can lead to crossings, as a steepest descent path may cross a steepest ascent path (two steepest descent paths may merge, but will never cross). Therefore, to ensure the correct combinatorial structure, the construction of a path must also end if it encounters an already constructed path. The resulting subdivision is a quasi MS-complex and thus depends on the order in which the paths are constructed. Since, for our purposes, the steepest descent paths are more important, we require that all steepest descent paths are constructed first. In fact, we do not need the steepest ascent paths, and hence we will completely omit these paths from the construction. The resulting complex is commonly referred to as a descending (quasi) MS-complex [24]. The cells of a descending MS-complex are bounded by alternating minima and saddle points, and every cell contains exactly one maximum. In the remainder of this paper we refer to the descending quasi MS-complex as constructed above simply as the MS-complex, unless stated otherwise. The same rule applies to the components of the complex, namely the MS-cells and MS-edges.

The quasi MS-complex and therefore the descending quasi MS-complex can be computed in \( O(n \log n) \) time [10].

3.2 Lowest paths

The following lemmas state that the lowest paths lie mostly on the MS-complex. The proofs can be found in the full version of the paper. We also sketch how lowest paths can be computed.

▶ Lemma 1. Let \( \pi \) be the lowest path between two vertices \( u \) and \( v \) in \( G \). Then the highest point of \( \pi \) is at \( u \), \( v \), or a saddle point.

▶ Lemma 2. Let \( u \) and \( v \) be two adjacent vertices on a lowest path \( \pi \) with \( h(u) > h(v) \). Then the edge \((u,v)\) must be the edge of steepest descent among the edges in the wedge of \( S^\downarrow(u) \) that contains \((u,v)\).

▶ Lemma 3. Let \( \pi \) be the lowest path between two vertices \( u \) and \( v \) in \( G \), where both \( u \) and \( v \) lie on MS-edges. Then all vertices of \( \pi \) lie on MS-edges.

▶ Lemma 4. Let \( \pi \) be the lowest path between two vertices \( u \) and \( v \) in \( G \), and let \( u' \) and \( v' \) be the first vertices on an MS-edge encountered by following the steepest descent path from \( u \) and \( v \), respectively. Then \( \pi \) is the concatenation of the steepest descent path from \( u \) to \( u' \), the lowest path from \( u' \) to \( v' \), and the steepest descent path from \( v \) to \( v' \) in reverse.

We can compute lowest paths efficiently using the MS-complex. Specifically, we construct a lowest path tree from the source vertex as follows. We start with a disconnected set of vertices consisting of the source and all minima. Then we add all saddle points in increasing order of height. For every saddle, we add the saddle point and the MS-edges to adjacent minima in the MS-complex. However, if two (or more) minima are already in the same connected
component in the lowest path tree, then we remove the first edges of the corresponding MS-edges (the edges incident to the saddles), except for the first edge that descends the steepest. After all saddle points are added, we compute the steepest descent edge for all vertices internal to MS-cells. This results in the correct lowest path tree by Lemma 3 and Lemma 4. The method runs in $O(n + m \log m)$ time using sorting, where $m$ is the number of critical vertices.

### 4 Striation

Before we specify the sand function needed to construct a $\delta$-network, we can consider the complexity of the problem of computing a representative $\delta$-network for a generic sand function. Let $\pi_0$ and $\pi_1$ be two semi-simple non-crossing paths from $\sigma$ to $\tau$. For any reasonable sand function $d$, the value of $d(\pi_0, \pi_1)$ should measure only sand in the region $D(\pi_0, \pi_1)$ in between the paths. To enforce this, we can restrict the domain of the homotopy $\eta$ from $\pi_0$ to $\pi_1$ to $D(\pi_0, \pi_1)$. Furthermore, we want to ensure that each volume of sand is measured at most once. This can be achieved by restricting $\eta$ to be an isotopy, as well as monotone.

Now consider an input terrain consisting of a sequence of pyramids with different heights as shown in Figure 3, where all non-peak vertices are at height $0$. Let $\pi_0$ and $\pi_1$ be two paths from source to sink at height $0$ and let $P$ be the set of pyramids in $D(\pi_0, \pi_1)$. We say that a sand function $d$ is well-behaved if $d(\pi_0, \pi_1) = \sum_{p \in P} \text{vol}(p)$ where $\text{vol}(p)$ is the sand volume of the pyramid $p$. It is easy to see that computing a representative $\delta$-network for a terrain of this type is NP-hard if the sand function is well-behaved, by reduction from Partition (see Fig. 3). We note that all sand functions that we use are well-behaved.

To make the problem tractable, we put a restriction on the paths that can be used in a representative $\delta$-network. It is not sufficient to restrict the paths to be monotone, since the NP-hardness reduction still works for monotone paths. As a stronger restriction we can specify a monotone isotopy $\eta$ between the boundaries $\pi_{\partial M}^-$ and $\pi_{\partial M}^+$. We require every path in a $\delta$-network to be a level curve of $\eta$. This ensures that the candidate paths cannot cross each other or themselves. The resulting $\delta$-network then strongly depends on the choice of $\eta$. Naturally we want to choose $\eta$ in such a way that many low paths are included as candidates. To that end we use the MS-complex. Instead of completely specifying $\eta$, we specify only a discrete subset thereof consisting of suitable candidate paths. We define a striation $S$ as an

---

2 That is, a homotopy whose intermediate curves have no self-intersections.

3 That is, for $t \leq t'$, $\eta(\cdot, t)$ and $\eta(\cdot, t')$ have no proper crossings and $\eta(\cdot, t) \subseteq D(\eta(\cdot, 0), \eta(\cdot, t'))$. 
Ordered set of non-crossing paths $S = \{\pi_0, \ldots, \pi_r\}$ from $\sigma$ to $\tau$ with $\pi_0 = \pi_{\partial M}^-$ and $\pi_r = \pi_{\partial M}^+$. Every path in a striation must be composed of MS-edges and between every two consecutive paths $\pi_i$ and $\pi_{i+1}$ in a striation there can be at most one MS-cell and possibly several one-dimensional features. The one-dimensional features arise from overlapping MS-edges or from the way the striation is computed. Note that every striation can be completed to a monotone isotopy between $\pi_{\partial M}^-$ and $\pi_{\partial M}^+$.

**Computing a striation.** The hardness result of the previous section also directly implies that computing a striation that includes a representative $\delta$-network is NP-hard. Therefore we consider several heuristics. For every heuristic we require that the lowest path between source and sink is in the striation. This is possible, since by Lemma 3, the lowest path is composed of MS-edges.

Two of our three heuristics for computing the striation use the persistence of local maxima. The persistence of a local maximum is the difference in height to a saddle with which it is paired, and a local maximum is paired with the lowest saddle on a highest path to a higher maximum. It can be computed in linear time from the contour tree, which can be computed in $O(n + m \log m)$ time \cite{7,8} when there are $m$ critical points. All local maxima except for $v_\infty$ will be paired and have their persistence defined.

**Iterated lowest path.** As a first step we compute the lowest path $\pi$ from source to sink. We then subdivide $G$ along $\pi$ into two parts: $D_1 = D(\sigma, \pi_{\partial M}^-)$ and $D_2 = D(\pi_{\partial M}^+, \tau)$. Next, we recursively apply the algorithm in $D_1$ and $D_2$. The obtained striations $S_1$ and $S_2$ can then be concatenated to obtain the final striation $S = \{S_1, \pi, S_2\}$. Finally we can add $\pi_{\partial M}^-$ and $\pi_{\partial M}^+$ to the striation. The recursion stops when $G$ contains at most one MS-cell.

We have to be careful when computing $\pi$, since $\pi$ must be distinct from $\pi_{\partial M}^-$ and $\pi_{\partial M}^+$.
More precisely, $\pi$ must have an MS-cell on both sides. Since $\pi$ is also a path in $D_1$ and $D_2$, we must often compute the second or third lowest path. We can do so by computing the lowest paths through all edges that are not in the lowest path tree. The lowest path that is not one of the boundary paths can then easily be obtained.

**Highest persistence first.** The first path $\pi$ is obtained by computing the lowest path from source to sink that passes through the maximum $q$ with the highest persistence (excluding $v_\infty$). Since $\pi$ actually consists of two lowest paths (from source to $q$, and from $q$ to sink), $\pi$ has the form of a path $\pi'$ with a special vertex $v$ from which there is a path to $q$ and back to $v$ (Lemma 4). We now subdivide $G$ as follows. Let $c$ be the MS-cell containing $q$, and let $u_1$ and $u_2$ be the first and last vertices of $\pi$ that are on the boundary of $c$, respectively (see Fig. 4). Furthermore, let $\pi_{ccw}$ and $\pi_{cw}$ be the paths between $u_1$ and $u_2$ along the boundary of $c$ in clockwise and counterclockwise direction, respectively. We can now obtain the path $\pi_i$ as the concatenation of the subpath of $\pi$ from $\sigma$ to $u_1$, the path $\pi_{ccw}$, and the subpath of $\pi$ from $u_2$ to $\tau$. Similarly, we can obtain $\pi_{i+1}$ by replacing $\pi_{ccw}$ by $\pi_{cw}$ in $\pi_i$. If $u_1 = u_2$, then either $\pi_i$ or $\pi_{i+1}$ may backtrack from $u_1$. In that case we can replace the respective path with $\pi'$. The paths $\pi_i$ and $\pi_{i+1}$ subdivide $G$ into three parts (see Fig. 5): $D_1 = D(\pi_i, \partial M), D_2 = D(\pi_{i+1}, \pi_i)$, and $D_3 = D(\pi_{cw}, \pi_{i+1})$. Since $D_2$ contains only one MS-cell, we recurse only in $D_1$ and $D_3$ to obtain $S_1$ and $S_3$. The final striation then consists of $S = \{S_1, \pi_{i+1}, \pi_i, S_3\}$. During recursive calls we do not recompute the persistence of maxima, but use the persistence computed initially.

**Hybrid.** We first compute the lowest path $\pi$ from source to sink. Next we pick the maximum $q$ with the highest persistence among all maxima for which $\pi$ contains an MS-edge on the boundary of the corresponding MS-cell. This heuristic then proceeds in the same way as the highest persistence first heuristic, using the lowest path through $q$.

It is easy to verify that all heuristics produce a striation and that this striation includes the lowest path from source to sink (none of the chosen paths can cross the lowest path). The complexity of the striation – the summed complexities of its paths – is $O(nm)$ in all cases, where $m$ is the number of local maxima. The first heuristic may seem the most natural. However, lowest paths computed in recursive steps will often stay close to the original lowest path and differ only in bars with low persistence. It may make more sense to deviate first in the bar with highest persistence, as this may lead directly to a path that is $\delta$-dissimilar to the lowest path. This is what the second heuristic is designed for. However, the lowest path through the maximum with highest persistence might require a long backtracking path to reach this maximum. This will force many paths to go around possibly the wrong side of a bar. Note that, in a sense, this problem is dual to the problem of the first heuristic. The final heuristic, a hybrid of the first two, tries to alleviate this problem by requiring the chosen maximum to be close to the lowest path.

**5 Sand function**

Before we can compute a representative $\delta$-network from a striation, we need to define the sand function and show how to compute it for two paths. Let $\pi_i$ and $\pi_j$ ($i < j$) be two paths in the striation. To compute $d(\pi_i, \pi_j)$ we first need to specify a monotone isotopy $\eta$ between $\pi_i$ and $\pi_j$. For consistency, we require $\eta$ to be monotone with respect to the striation. An isotopy $\eta$ is *striation monotone* with respect to a striation $S = \{\pi_0, \ldots, \pi_r\}$ if for every $k$ ($i < k < j$) there exists a $t_k$ such that $\eta(\cdot, t) \subseteq D(\pi_i, \pi_k)$ for all $t \leq t_k$ and $\eta(\cdot, t) \subseteq D(\pi_k, \pi_j)$ for all $t \geq t_k$. Intuitively, for every path $\pi_k$ in the striation between $\pi_i$ and $\pi_j$ there must exist a level curve of $\eta$ that matches $\pi_k$, and other level curves cannot
(a) water level model, (b) water flow model, (c) symmetric flow model.

We now show how to compute \( h(\eta(p, t)) \) between two matching curves \( \pi_i \) and \( \pi_j \). This function can then be extended to a sand function \( d(\pi_i, \pi_j) \) from \( \pi_i \) to \( \pi_j \) as described in Section 2. We consider three different models (see Fig. 6):

**Water level model.** This sand function simply measures the amount of sand above \( h(\eta(p, 0)) \).

Intuitively, the sand function measures the minimum amount of sand that needs to be removed such that \( \pi_i \) and \( \pi_j \) merge if the water level is at the height of \( \pi_i \). Formally, we set \( \zeta(p, t) = h(\eta(p, 0)) \).

**Water flow model.** This sand function measures the amount of sand that needs to be removed such that the function \( h(\eta(p, \cdot)) \) becomes non-increasing. Formally, we set \( \zeta(p, t) = \min_{t' \leq t} h(\eta(p, t')) \).

**Symmetric flow model.** This sand function measures the amount of sand that needs to be removed such that \( h(\eta(p, t')) \) is unimodal. Formally, we set \( \zeta(p, t) = \max(\min_{t' \leq t} h(\eta(p, t')) \), \( \min_{t' \geq t} h(\eta(p, t')) \) \).

Note that the sand functions in the water level model and the water flow model are not symmetric, whereas the sand function in the symmetric flow model is symmetric. In general we do not assume that the sand function is symmetric.

Finally, we need to compute the isotopy \( \eta \) between two paths \( \pi_i \) and \( \pi_j \). In general we define the sand function as \( d(\pi_i, \pi_j) = \inf_\eta \text{vol}(\Sigma_\eta^i) \), where \( \zeta \) is defined by the model as described above. Below we describe how to compute \( d(\pi_i, \pi_j) \) only for the water flow model, but it is straightforward to extend the same approach to the other models.

Following the definition of the water flow model, we say that \( \pi_i \) is similar to \( \pi_j \) if there exists a striation monotone isotopy \( \eta \) such that all matching curves are non-increasing in \( h \). Note that \( d(\pi_i, \pi_j) = 0 \) if \( \pi_i \) is similar to \( \pi_j \). To be able to argue the correctness of our computation, we introduce a different but related definition. We say that \( \pi_i \) is pseudo-similar to \( \pi_j \) if, for every point \( q \) in \( D(\pi_i, \pi_j) \), there exists a point \( p \in \pi_i \) and a striation monotone path \( \pi \) from \( p \) to \( q \) such that \( \pi \) is non-increasing in \( h \). Omitted proofs can be found in the full version of the paper.

**Lemma 5.** If \( \pi_i \) is pseudo-similar to \( \pi_j \), then \( d(\pi_i, \pi_j) = 0 \).

**Computing the sand function.** We now show how to compute \( d(\pi_i, \pi_j) \) efficiently in the water flow model. The algorithm can operate directly on the MS-complex. First consider two consecutive paths \( \pi_i \) and \( \pi_{i+1} \) in the striation. Let \( B(\pi_i, \pi_{i+1}) \) and \( T(\pi_i, \pi_{i+1}) \) be the sets of critical points that are bounding the MS-cell \( c_i \) between \( \pi_i \) and \( \pi_{i+1} \) and are on \( \pi_i \).
and \(\pi_{i+1}\), respectively. Now let \(h^*\) be the maximum height among all points in \(B(\pi_i, \pi_{i+1})\) and assign \(h^*\) to \(c_i\). We claim that the volume of sand in \(c_i\) above \(h^*\) equals \(d(\pi_i, \pi_{i+1})\). For non-consecutive paths \(\pi_i\) and \(\pi_j\), we need to propagate the heights. After assigning \(h^*\) to \(c_i\), we also lower the height of all points in \(T(\pi_i, \pi_{i+1})\) to \(h^*\) (points with lower height than \(h^*\) keep their height). We continue the propagation until we reach \(\pi_j\). In case there is a one-dimensional feature between two consecutive paths, then we also need to propagate the height over this one-dimensional feature separately from propagating over MS-cells. However, note that one-dimensional features do not add any volume to the sand function. This process assigns certain height values \(h_j^*\) to each of the MS-cells \(c_k\) between \(\pi_i\) and \(\pi_j\), and \(d(\pi_i, \pi_j) = \sum_{k=i}^{j-1} \text{vol}(c_k, h_k^*)\), where \(\text{vol}(c_k, h_k^*)\) is the volume of sand in \(c_k\) above \(h_k^*\).

**Lemma 6.** If \(D(\pi_i, \pi_j)\) is altered by lowering every MS-cell \(c_k\) in \(D(\pi_i, \pi_j)\) to \(h_k^*\), then \(\pi_i\) is pseudo-similar to \(\pi_j\).

**Proof.** For the sake of contradiction, assume that we can make \(\pi_i\) similar to \(\pi_j\) by removing less than \(\sum_{k=i}^{j-1} \text{vol}(c_k, h_k^*)\) volume of sand from \(D(\pi_i, \pi_j)\). By Lemma 5 we can then also make \(\pi_i\) pseudo-similar to \(\pi_j\) by removing the same volume of sand. Now there must be some point \(q\) in some MS-cell \(c_k\) (if \(q\) is on an MS-edge, then let \(k\) be as small as possible) that has height higher than \(h_k^*\) in the altered \(D(\pi_i, \pi_j)\). We now show by induction on \(k\) that there is no striation monotone non-decreasing path from \(q\) to \(\pi_i\), refuting that \(\pi_i\) is pseudo-similar to \(\pi_j\). If \(k = i\), then any non-decreasing path from \(q\) to \(\pi_i\) must end at some point \(q'\) on the boundary of \(c_i\). But since the height of \(q'\) can be at most the maximum height of all points in \(B(\pi_i, \pi_{i+1})\), which is \(h^*_i\), this is not possible. If \(k > i\), then any non-decreasing path from \(q\) to \(\pi_i\) must hit \(\pi_k\) at some point \(q'\) at a height above \(h_k^*\). If \(q' \in \pi_i\), then there must exist a point among \(B(\pi_k, \pi_{k+1})\) that is on \(\pi_i\) and has height above \(h_k^*\). However, this is not possible by construction. Otherwise, \(q'\) lies on some MS-cell \(c_l\) with \(l < k\). By construction \(h_l^* \leq h_k^*\) and thus \(q'\) has height higher than \(h_k^*\). The result now follows by induction.

Lemma 6 and Lemma 7 prove that we can compute \(d(\pi_i, \pi_j)\) as \(\sum_{k=i}^{j-1} \text{vol}(c_k, h_k^*)\). To compute the values \(h_k^*\) we simply need to propagate the height values on the MS-complex by following the striation. To compute \(\text{vol}(c_k, x)\) efficiently for some height \(x\), we can preprocess each MS-cell. Note that \(\text{vol}(c_k, x)\) is a monotone function of \(O(n_k)\) complexity, where \(n_k\) is the number of vertices in \(c_k\). After computing this function in \(O(n_k \log n_k)\) time using a plane sweep, we can query \(\text{vol}(c_k, x)\) for any height \(x\) in \(O(\log n_k)\) time. The total preprocessing time is \(O(n \log n)\), after which the sand function from any \(\pi_i\) to any \(\pi_j\) can be determined in \(O([i-j] \log n) = O(m \log n)\) time.

To compute the sand functions for other models, we only need slight modifications. For the water level model we need the following modification: when we propagate a height \(h_k^*\) to all points in \(T(\pi_i, \pi_{i+1})\), we also update heights lower than \(h_k^*\) to \(h_k^*\). The symmetric flow model requires the propagation in both directions, that is from \(\pi_i\) to \(\pi_j\) and from \(\pi_j\) to \(\pi_i\). The height assigned to an MS-cell is then the maximum height assigned by any direction of the propagation. The correctness proofs are very similar to those of the water flow model.

### 6 Representative network

To obtain our representative network, we sort the \(O(m)\) paths of our striation \(S = \{\pi_0, \ldots, \pi_r\}\) based on lexicographic height. Each path \(\pi_i\) contains \(O(n)\) edges. We preprocess each path in \(O(n \log n)\) time to compute its height profile function [21]. After this, we can compare
the lexicographic heights of two striation paths in $O(n)$ time. Hence, sorting the paths by lexicographic height takes $O(mn \log n + mn \log m) = O(mn \log n)$ time. By our non-degeneracy assumptions, all paths have different lexicographic height. Assume $S$ is the sorted set of paths.

To compute the representative network, we initialize the representative network as an empty set $R$ of paths. We iterate over the paths in $S$ in order of increasing lexicographic height, and we add a path $\pi$ to $R$ if each path $\pi'$ already in $R$ has $d(\pi, \pi') \geq \delta$. This can be tested in $O(m \log n)$ time by testing only the two paths of $R$ in between which $\pi$ would be inserted. Overall, this yields an $O(mn \log n)$ running time given the striation. By construction we obtain a representative $\delta$-network, that is, no better $\delta$-network exists.

### 7 Experimental results

We performed experiments on two data sets: a numerically modeled river and a real-world braided river. To do so we implemented the highest persistence first heuristic for the striation and the sand function in the water flow model. The numerically modeled river was created by a state-of-the-art model suite that is used in the civil engineering and fluvial and coastal morphology disciplines worldwide, indicating its usefulness and quality [23]. The real-world case is an iconic braided gravel river in New Zealand called Waimakariri, one of the largest in the world of this type, that was the first with spatial cover of the bed through remote sensing techniques [11].

Figure 7 shows our results for the numerically modeled river. The representative network is capturing most channels of the river. Channels that are added in the denser network cross big bars and are in most cases located in small tie-channels that formed at water levels when water was spilling over the bar. These are essential in the natural braiding dynamics in that these can be the locations where bars split during floods. Furthermore, the sparser network also avoids deep short channels which are connected on one or no end. Some of these formed as filling lows between two merging bars rather than active channels, and the fact that they only partake in the dense network when connected at small tie-channels is evidence that the network method represents the river channel network in these important aspects.

Figure 8 shows a representative network for the Waimakariri. This complex topography has many large remnant channels that became inactive while smaller channels may be
Growing. The network captures some smaller channels and leaves out the unconnected remnant channels. The network sometimes loops to flow upstream, meaning that such channels should either not be connected or that minor tie-channels were missed, which can now for the first time be investigated through comparison of sparser and denser networks.

8 Discussion and future work

The representative $\delta$-networks computed using our approach already look very promising, even though we have implemented only one striation heuristic and only one sand function. The next step is a thorough testing on different river bed terrain data using each combination of a striation and a sand function, and with different values of $\delta$. The output can be further assessed by geomorphologists, for example by comparing the computed networks with the corresponding actual river networks at different water levels. Other striations and other sand functions can also be considered, and may give better results in certain scenarios.

In particular, our striation heuristic may give counter-intuitive results when $M$ includes the banks of the river. To illustrate this, consider the two high mountains on both sides of the river banks in Figure 9. It can occur that such mountains are the most significant maxima, and cause the majority of the river to go over the banks, instead of between the mountains. For this reason, we may want to base the significance of maxima also on their distance to the main channel of the river. Our hybrid heuristic already tries to address this issue, but a more refined approach may be needed.

In the future we also want to consider time-varying data on rivers, both as a tool to improve the results by eliminating features that do not persist over time, and as a way to analyze how the river structure evolves over time. However, to perform such an analysis efficiently, we may need to simplify our model.

References

1 Pankaj Agarwal, Mark de Berg, Prosenjit Bose, Katrin Dobrint, Marc van Kreveld, Mark Overmars, Marko de Groot, Thomas Roos, Jack Snoeyink, and Sidi Yu. The complexity


