Embedding Rivers in Polyhedral Terrains *

Marc van Kreveld
Dept. of Information and Computing Sciences
Utrecht University
The Netherlands
marc@cs.uu.nl

Rodrigo I. Silveira
Dept. of Information and Computing Sciences
Utrecht University
The Netherlands
rodrigo@cs.uu.nl

ABSTRACT

Data conflation is a major issue in GIS: spatial data obtained from different sources, using different acquisition techniques, needs to be combined into one single consistent data set before the data can be analyzed. The most common occurrence for hydrological applications is conflation of a digital elevation model and rivers. We assume that a polyhedral terrain is given, and a subset of its edges are designated as river edges, each with a flow direction. The goal is to obtain a terrain where the rivers flow along valley edges, in the specified direction, while preserving the original terrain as much as possible.

We study the problem of changing the elevations of the vertices to ensure that all the river edges become valley edges, while minimizing the total elevation change. We show that this problem can be solved using linear programming. However, several types of artifacts can occur in an optimal solution. We analyze which other criteria, relevant for hydrological applications, can be captured by linear constraints as well, in order to reduce such artifacts. We implemented and tested the approach on real terrain and river data, and describe the results obtained with different variants of the algorithm. Moreover, we give a polynomial-time algorithm for river embedding for the special case where only the elevations of the river vertices can be modified.

Categories and Subject Descriptors

F.2.2 [**Nonnumerical Algorithms and Problems**]: Geometrical problems and computations; G.1.6 [**Optimization**]: Linear programming

General Terms

Algorithms, Experimentation, Theory

*This research was funded by the Netherlands Organisation for Scientific Research (NWO) under BRICKS/FOCUS grant number 642.065.503, and under the Free Competition project GOGO.

SCG'09, June 8–10, 2009, Aarhus, Denmark.

Keywords

polyhedral terrain, river network, linear programming

1. INTRODUCTION

Data used in Geographic Information Systems (GIS) can be acquired in various ways and has widely differing characteristics. Even for the same type of data, different acquisition techniques give rise to different properties of the data. For example the extent of a forest can be obtained from land surveying, which is precise and reliable, but it can also be obtained from satellite imagery, which is less precise and less reliable, because it depends on classifications of pixels. On the other hand, land surveying is more expensive and therefore, less frequently performed. This implies that available data can easily be outdated. As a second example, consider a data set with the boundaries of the states of the USA and a data set with the rivers of the USA. These data sets have different origins, but if they are used in conjunction, one should be aware that often, the boundary of a state follows part of a river. If the river data set has a lower resolution, then it seems that a state also has small patches of land on the other side of the river.

These two examples illustrate the issue of spatial data integration in GIS. *Data conflation* is the compilation or reconciliation of two spatial data sets covering overlapping regions [2, 20]. When two data sets refer to the same area, data conflation may be necessary to obtain data that is consistent, without implicit or explicit errors.

In this paper we discuss a data conflation problem for a digital elevation model and a river data set. In particular, we assume that the digital elevation model is given as a polyhedral terrain and the river data set as a forest of trees embedded in the plane, each with a flow direction. We wish to conflate these two data sets, which means that the rivers should follow edges of the terrain, these edges should indeed be valleys with respect to the incident triangles, and the direction of flow should be correct. To accomplish this, we may need to add edges to the terrain, move terrain and river edges, and/or change the elevations of vertices in the terrain.

Motivation and background The importance of terrain and river conflation lies in hydrology and erosion modeling on terrains. Water can flow through the surface and on the surface, and in the latter case, it may flow freely or in rills, streams or rivers. One of the reasons for erosion is the sediment transportation capacity of flowing water, which leads to soil loss.

A standard assumption of water flow on a terrain is that

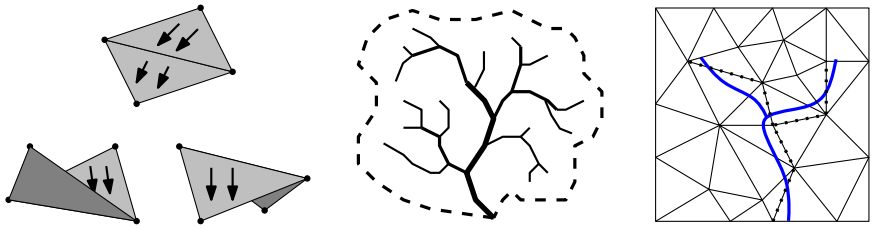

Figure 1: Left: three types of edges: normal or transfluent, valley or cofluent, and ridge or difluent. Center: a river network and its drainage basin (dashed). Right: the terrain, input rivers (blue), and snapped rivers (dotted).

water follows the direction of steepest descent. When a polyhedral terrain is used as a digital elevation model, the places where water collects are the so-called cofluent or valley edges [6], see Fig. 1. Whereas most edges receive water from one incident triangle and pass it on to the other, valley edges receive water from both incident triangles, after which the water follows the downhill direction on that edge. Hence, valley edges are the natural way to represent rivers (for brevity we use *rivers* for any flowing stream on a terrain, be it a rill, gully, stream, creek, brook, tributary, or river). Rivers typically form a network that is a tree rooted at a local minimum or the boundary of the terrain, see Fig. 1. The part of the terrain that drains into a particular river is the drainage basin, or watershed, of that river.

Most hydrological modeling research assumes a gridded digital elevation model (DEM). Each grid cell drains to one of the eight neighbors, the one that gives the steepest descent. This is called the D8 model, and allows the definition of accumulation of water, drainage networks, and drainage basins. World's most used GIS, ArcGIS, contains tools for this in its Spatial Analyst extension. It also contains tools to correct DEMs hydrologically. A DEM is hydrologically correct if it has a connected drainage structure and a correct representation of ridges and streams. It implies that there are no spurious pits: local minima that are caused by insufficiently dense sampling, inaccuracies in elevation measurement, or rounding of elevations [8, 9, 14].

Correction is done by changing elevations of the grid cells. There are essentially three approaches to ensure that given rivers correspond to those according to the terrain shape. The simplest and most used method is that of *stream burning*: lower only the elevations of cells through which the river passes. Secondly, one can specify a buffer around the river and also allow a change of elevation of cells in the buffer. This allows a more smooth incorporation of rivers, and is the approach in the ArcGIS Spatial Analyst extension and in AGREE [7]. Thirdly, one can (potentially) allow all cells of the DEM to change elevation. The ANUDEM system allows this [8, 9, 10]. The three approaches are compared experimentally in [1].

To ensure that spurious local minima are removed, one can employ *pit filling* and *breaching*. Pit filling is to increase the elevation of a local minimum and some grid cells around it until no local minimum remains [11]. Breaching is to lower elevations of grid cells along a path from the local minimum to an even lower point in the neighborhood [15].

Once a hydrologically correct terrain is obtained, further analysis can be performed to compute hydrographs, determine runoff, or to simulate the hydrological cycle (including rainfall, evaporation, overland flow, subsurface flow, ...). Clearly, many geographic aspects influence hydrological modeling, like soil type, vegetation, precipitation intensity, and terrain shape.

While hydrological modeling has mostly assumed a gridded representation of elevation, research on extracting drainage networks, ridges, and watersheds has also assumed a polyhedral terrain (called a TIN representation in GIS) as the source data [4, 17, 22, 24]. Furthermore, several terrain correction methods on polyhedral terrains have been studied, including spurious pit removal and drainage component minimization [5, 13, 21]. Using river data to extend the hydrologic quality of terrain data is considered in [17, 19, 23]. These methods let the triangulation have edges where the given rivers run, but do not guarantee that these edges are valley edges.

With the rise of LiDAR (Light Detection and Ranging) for elevation data acquisition, irregular point clouds and triangulations thereof are becoming more common as elevation models, creating a need for algorithms that work on polyhedral terrain representations. Furthermore, for hydrologic simulations, triangulations are considered superior to DEMs, as shown by the experiments of Vivoni et al. [23].

Our approach and results We study polyhedral terrain (TIN) correction based on additional river data. We will compute a polyhedral terrain with minimally changed elevations of vertices while ensuring that given rivers flow along valley edges, and in the right direction.

We first assume that a polyhedral terrain is given where a subset of the edges are marked as *river edges*, and a flow direction is specified. We study the problem of changing the elevations of potentially all vertices to let all of these river edges become valley edges of the terrain, while minimizing the sum of elevation changes of the vertices. We show in Section 2 that this problem can be solved by linear programming. However, an optimal solution may contain artifacts like sharp folds that were not present before. Therefore we analyze how they can be avoided while still using linear programming, and what other hydrological constraints can be incorporated.

To test our approach, we implemented it and run it on independent terrain and river data from British Columbia. We describe the data, the preparation of the data, and the experimental objectives in Section 3. Then we evaluate the results and visualize various attributes like the change in elevation in different parts of the terrain, and the change in slope.

A special case of the river embedding problem is one where

Terrain criteria	Model with LP?	Exclusions for feasibility
Valley & flow constraints for rivers	Yes	None (flat terrain works)
Strict decline along rivers	Yes	No directed cycles & wingtips of river edges on rivers
Valley edges in the strict sense	Yes	Wingtips of river edges on rivers
Contiguous flow through rivers	No (approx. yes)	Non-river edge connections between rivers
Limit edge slope change	Yes	Edges incident to river vertices
Limit triangle slope change	No (approx. yes)	Triangles edge- or vertex-adjacent to rivers
Local minima removal	No	n/a

Table 1: Terrain criteria that can be represented as linear constraints or not, and when such constraints can make the LP infeasible.

we may only change the elevations of vertices incident to river edges, the *river vertices*. This is the stream burning setting that was discussed for DEMs. This restricted problem can obviously still be solved using linear programming, but this does not give a strongly polynomial-time solution (it does give an expected subexponential time solution [12, 16]). We show that this special case—with some additional assumptions—can be solved in strongly polynomial time in Section 4.

2. CONSTRAINTS FOR TERRAIN CORRECTION

In this section we study the criteria that make river edges to be valley edges on a terrain. We also study other criteria that influence the morphological quality. The objective is to formulate the criteria as much as possible as linear constraints, so that standard linear programming software like CPLEX can be used for terrain correction.

First we introduce some notation. We are given a triangulation $\mathcal{T}$ with heights, a subset of its edges that are marked to be river, each with a flow direction. Each vertex v_i has an initial height h_i^0. The goal is to determine a new height h_i for each vertex v_i that minimizes the sum of the absolute difference between the new height and the original height of each vertex: $\min \sum_i |h_i - h_i^0|$. By using the standard transformation $h_i = h_i^0 + z_i^+ - z_i^-$, where $z_i^+, z_i^- \geq 0$, the objective function becomes linear: $\min \sum_i (z_i^+ + z_i^-)$. In Subsection 2.1 we show that turning river edges into valley edges with the proper flow direction is an instance of linear programming (LP). In Subsection 2.2 we analyze what other criteria can be incorporated with an LP approach.

2.1 Constraints for embedding rivers

Each edge e of $\mathcal{T}$ is incident to two triangles. We call the two vertices of such a triangle that are not incident to e its *wingtips*. If e is marked as a river edge, then two constraints need to be imposed to ensure that e is a valley edge, and another constraint is needed to ensure the correct flow direction.

Let H be the plane containing e and the horizontal line perpendicular to e and through some point on e. Then e is a (weak) valley edge if and only if both wingtips lie above or on the plane H. For both triangles incident to e, this constraint is a linear inequality on the three heights of the vertices involved, called *valley constraint*. The flow direction involves only the two heights of the vertices incident to e, and obviously specifies that the vertex to which the flow goes is at least as low as the other vertex. This constitutes the *flow constraint*.

THEOREM 1. *Given a polyhedral terrain with n vertices, of which a subset of the edges are marked as river edges with a flow direction, the problem of minimizing the total height change of all vertices, while embedding the river edges as valley edges with the specified flow direction, is a linear programming problem with $O(n)$ variables and constraints.*

Observe that the polyhedral terrain where all vertices have the same height is a feasible solution, regardless of which edges are specified to be river. In principle, we prefer strict inequalities for the valley and flow constraints. If the river edges do not form a directed flow cycle and no triangle of $\mathcal{T}$ incident to a river edge has its wingtip at a river vertex as well, then a solution exists even with strict inequalities (although an optimal solution is not well-defined when we use strict inequalities). We can simply lower all vertices that are endpoints of river edges to some very low height which is the same for all of these vertices. This makes all river edges to be valley edges in the strict sense (essentially, this is what stream burning does on a gridded DEM). Then we can modify the heights slightly to ensure the flow direction in the strict sense as well.

2.2 Constraints for other terrain criteria

We analyze other terrain criteria that can be enforced using—hopefully linear—constraints involving the heights of the vertices. The purpose is two-fold. Firstly, it may help to limit artifacts like substantial slope change of triangles that can arise when we embed river edges in the terrain. Secondly, it may be possible to ensure other useful characteristics of terrains with respect to flow.

A summary of criteria and whether they can be represented by linear constraints is presented in Table 1. The table also specifies whether the linear constraints for a terrain criterion can make the LP infeasible (in combination with the valley and flow constraints). Infeasible LPs are not useful because the purpose is to generate an improved terrain. To ensure feasibility, we may need to adapt the terrain or exclude constraints for certain edges and triangles.

Strict decline along rivers We can specify that every terrain edge that is part of a river has a decline (slope) in its flow direction of at least some constant. This gives a linear inequality involving only the elevations of the river vertices. Directed cycles of river edges will make the LP infeasible, and possibly also triangles that are incident to a river edge and another river vertex (the wingtip of a river edge meets the same river again). To deal with infeasibility of the second type, we can add one Steiner point on an edge, splitting two triangles into two subtriangles each.

Valley edges in the strict sense To ensure that the flow

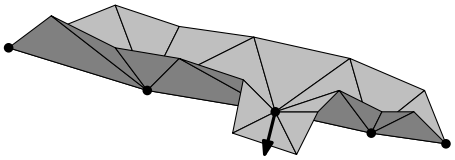

Figure 2: Water can escape from a sequence of valley edges at a leak vertex.

direction on a triangle incident to a river edge is actually into the river and not parallel to it, we would need a strict inequality in the valley constraint. We can specify a decline (slope) of at least some constant in the direction perpendicular to the river edge using a linear constraint. As long as wingtips of river edges do not lie on a river vertex, this cannot lead to infeasibility.

Contiguous flow through rivers Although standard drainage extraction methods for terrains will select all valley edges as part of the drainage network, it is not guaranteed that the water in a river will flow from its source to its outlet even if all river edges are valley and the flow direction is ensured. It can happen that another, more steeply descending flow direction exists from a vertex of the river, see Fig. 2. Such a vertex is called a *leak vertex*. Since water is assumed to follow the direction of steepest descent, the water in the river will leave its intended course, while at the next river edge, a "new" river will start. This implies that hydrologic analyses using accumulation of water will give incorrect results.

To ensure that water actually flows through the entire river from its source down, we must guarantee that the river edge e down from a vertex v is steeper than any other edge or segment over a triangle leaving v. While the condition that one edge is steeper than another is easily captured with a linear constraint, this is not the case for segments over triangles. Moreover, a triangle $\triangle uvw$ may exist where $v\vec{u}$ and $v\vec{w}$ descend mildly, whereas a segment from v to some point p on $\overline{uw}$ descends steeply, see Fig. 3. This can happen if the angle of $\triangle uvw$ at v is close to π, and therefore, p can be much closer to v than u or w. Requiring that the steepest sloping descent on a triangle is at most some value cannot be represented in a linear constraint.

We can give linear constraints that are only slightly stronger and do guarantee that the river edge e down from v is the steepest one. Let $s \leq 0$ be the downward slope of e, whose value depends linearly on the elevations of its endpoints. Let $t = \triangle uvw$ be some triangle not incident to e, and let α be the angle at v (see Fig. 3). We can require with two linear constraints that the slopes of $v\vec{u}$ and $v\vec{w}$

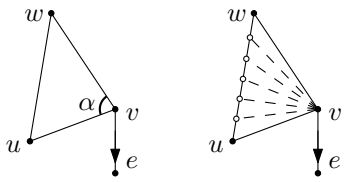

Figure 3: Using imaginary points on the opposite edge to ensure contiguous flow with mildly stronger constraints.

are at least $s \cdot \cos(\alpha/2)$, which ensures that the steepest slope occurring on t is at least s. To make such a constraint only slightly stronger than needed, we must make α small. This is easily done by using imaginary points along $\overline{uw}$ (see Fig. 3), and using linear constraints for the imaginary edges to these imaginary points as well. Instead of α, we use the much smaller angles between consecutive edges from $\overline{vu}$, the imaginary edges, and $\overline{vw}$. By increasing the number of imaginary points, we decrease the angle that we can use, and these constraints together become only slightly stronger than needed to ensure steepest descent from v along e. We do not need to generate constraints for the triangles incident to e, because the valley and flow constraints together ensure that no segment on these triangles is steeper down than e.

Preserve edge and triangle type Requiring that the flow situation of triangles and edges (see Fig. 1) does not change can be specified using linear constraints. For edges, we already showed that being valley is captured by two linear constraints, and obviously the same is true for being ridge. Also, the property for a transfluent edge that it receives flow from the one incident triangle and passes it on to the other is captured by two linear constraints. Note, however, that the property of being a transfluent edge is not strict enough: it leaves open which incident triangle is the one that gives flow to the edge, leading to a disjunction. Similarly, we cannot use linear programming to require that certain edges are *not* valley edges.

Some other related property we can maintain is the inflow and outflow edges incident to any triangle, because these are the same constraints.

Thirdly, edges on a terrain can be convex or reflex (or flat), which can be represented by a linear constraint for each edge. Therefore this property can also be maintained.

Elevation of lakes, drainage basins Often the hydrologic data contains lakes in the river. We can easily specify equal elevation of the vertices of the terrain that bound a lake, or specify an exact elevation of a lake if it is known, using linear constraints.

Drainage basins (watersheds) are usually found by tracing paths of steepest ascent from certain points in the terrain, in particular, from river junctions, passes, and local minima [17]. Maintaining drainage basins is important for hydrological correctness. However, it seems impossible to formulate the preservation of drainage basins by linear constraints. We noted that we can maintain ridges, which are part of drainage basin boundaries, using linear constraints. Furthermore, we do not expect drastic changes to drainage basins when we embed rivers, because most changes made will be local to the river edges, and drainage basin boundaries are mostly not close to rivers.

Local minima removal We cannot use a linear programming approach for spurious pit removal. To make sure that a vertex is not a local minimum, we would have to require that at least one of the neighbors is lower. These are linear constraints, but it is a disjunction instead of a conjunction so we cannot use linear programming. Furthermore, local minima removal while minimizing total elevation change was shown to be NP-hard [21].

Limit slope change One method to avoid artifacts on a terrain caused by changing elevations of some vertices is

river name	# terrain triangles	# river edges	river length	# river edges wrong flow	# river edges not valley
Goat	37,713	140	13.49 km	32 (22.86%)	47 (33.57%)
Khutze	216,648	407	34.16 km	71 (17.44%)	121 (29.73%)
Kiltuish	146,091	281	22.42 km	54 (19.22%)	78 (27.76%)
Kowesas	244,598	321	38.07 km	56 (17.45%)	108 (33.64%)
Paril	20,960	89	8.85 km	12 (13.48%)	24 (26.97%)
Triumph	20,382	104	11.37 km	28 (26.92%)	15 (14.42%)

Table 2: Original terrain and river data.

Goat River				Kowesas River			
variant	total abs. change	# vertices moved	average change	variant	total abs. change	# vertices moved	average change
VF	169 m	75	2.26 m	VF	246 m	146	1.69 m
VFR 10%	203 m	125	1.62 m	VFR 10%	281 m	247	1.14 m
VFRS 10%	219 m	131	1.67 m	VFRS 10%	297 m	264	1.12 m
VFR 1°	13,065 m	313	41.74 m	VFR 1°	159,201 m	761	209.20 m
VFRS 1°	44,225 m	1,582	27.96 m	VFRS 1°	2,720,426 m	23,912	113.77 m

Table 3: Changes in two of the rivers by five variants of the algorithm.

Goat River				Kowesas River			
variant	# LM	# FE	# LV	variant	# LM	# FE	# LV
— (Initial)	142	36	32	— (Initial)	465	77	80
VF	130	87	7	VF	456	151	23
VFR 10%	112	0	10	VFR 10%	408	0	20
VFRS 10%	110	0	7	VFRS 10%	407	0	19
VFR 1°	178	0	61	VFR 1°	615	0	221
VFRS 1°	121	0	12	VFRS 1°	477	0	39

Table 4: Artifacts created for two of the rivers, by five variants of the algorithm. Abbreviations used: LM: local minima, FE: flat river edges, LV: leak vertices.

to limit the change in slope of the edges. This will cause a propagation of an elevation change over a neighborhood, and therefore give a natural way to incorporate necessarily elevation changes to embed rivers. Restricting the change in slope of every edge to, say, 50% of the original slope is easily captured in a linear constraint. However, enforcing the constraint for river edges and edges incident to triangles that are incident to river edges can make the LP infeasible. We also note that (nearly) flat edges must stay (nearly) flat with such conditions, potentially causing propagation of a change in elevation over a large distance when a part of the terrain is nearly flat. There are various practical alternatives with which a change of slope of edges can be limited, but we will not go into them here.

The slope on a triangle is the maximum slope of any segment on it. The *aspect* is the compass direction in which the maximum downward slope is attained. Limiting the slope change of a triangle cannot be done with a linear constraint, but we can limit the slope change in a set of directions with a set of linear constraints, by which we limit the change in aspect as well. Again there are various options useful in practice that we do not discuss here.

3. EXPERIMENTS

The approach described in the previous section has been implemented and tested on real terrain and river data. The data corresponds to an area of British Columbia, with heights ranging from 0 to 1,600 m. We obtained a raster terrain model and a vector stream network from the Canadian Council On Geomatics.[1] The original terrain data was a 1:50,000 raster DEM, whereas the water streams were covered at a 1:20,000 scale (very similar data was used in [17] for extracting watersheds). We converted the raster terrain into a polyhedral terrain using ArcGIS, allowing a maximum vertical error of 5 m. The stream network of the area contained 11 named rivers, of which six were selected for our tests, including the three longest ones. Each original river, a polygonal line in 2D, was snapped to the terrain edges and processed as to avoid triangles with two river edges. The latter was done by selecting the other (third) edge every time two edges were part of the river. This adaptation resolved many causes for infeasibility of the resulting LP.

We tested five different variants of our linear programming algorithm. VF includes only the basic (weak) valley and flow constraints. VFR 10% adds a river decline constraint requiring each river edge to have a slope of at least 10% of the average river slope. This is to avoid flat river edges. The third variant, VFRS 10% adds a fourth constraint indicating that the slope of non-flat terrain edges cannot change by more than 50%. To guarantee a feasible LP, edges where one of the endpoints is a river vertex are excepted. Then we try two *extreme* additional variants: VFR 1° and VFRS 1° are the same as before, but now require each river edge to

[1]Available at `http://www.geobase.ca`.

have a vertical slope equivalent to at least 1°. This is clearly a very strong constraint: its only purpose is to understand how the algorithm propagates the change away from the river, in case a lot of change is needed. For each river and variant, we generate a linear pogram that is solved using CPLEX 9.1. We analyze two aspects of the solutions: the total elevation change, and the presence of certain artifacts.

The basic facts about the six rivers are shown in Table 2. It is worth noting the large percentage of river edges that violate one of the two basic constraints in the original data. Table 3 summarizes the change incurred by each variant. Due to space limitations, we only show the tables for two rivers. It can be seen that the first three variants incur a relatively small change, on average less than 2 m (averaged over the vertices with non-zero change).

If the only goal were to minimize the total elevation change, one would always choose VF, except for the fact that it can create artifacts. This is shown in Table 4. In particular, VF creates a large number of flat river edges. For the rivers studied, at least 40% of the edges become flat. VFR 10% resolves these artifacts with only a small additional total elevation change. In all cases, since the changes by VFR 10% were rather small and local, VFRS 10% produced similar results. The importance of the slope preservation constraint becomes evident in the more extreme case of VFR 1°. For the rivers tested, the least-cost way to create a river with 1° decline is always to create a dam around it. This is clearly visible in Fig. 4. Such a solution is unacceptable for practical applications, because the terrain shape suggests that there are two more rivers, one on either side of the dam. The variant VFRS 1° spreads the change towards the sides of the river, producing a larger total change, but a more natural terrain because there is still only one river in the valley. Table 4 also shows clearly that VFR 1° increases the number of artifacts considerably, whereas VFRS 1° is able to prevent many of them.

From the experiments we can conclude that the first three variants give valid rivers with relatively small and local elevation change. The extent of the change can be seen in Fig. 5. It is interesting to note that the number of local minima and leak vertices, which are not addressed explicitly, often also decreases. The most extreme case of Fig. 5, VFRS 1°, is shown in more detail in Fig. 6. We emphasize that this figure shows an extreme setting: the purpose is to understand how the algorithm propagates the change towards the sides of the river, in case a major change is needed. In most practical situations we expect that such major changes are not needed.

Finally, even though the goal of our tests was not to assess the running time of the methods, the methods appear suitable for real-world instances. For our largest instance (Kowesas River), CPLEX took less than 50 seconds to compute the solution to VFRS 1°, running on a Core 2 Duo 2.66Ghz computer with 2GB memory. The solution to the more realistic VFRS 10% variant was computed in only 7 seconds.

4. STRONGLY POLYNOMIAL-TIME ALGORITHM

Since it is an open problem whether a strongly polynomial-time algorithm for linear programming exists (expected sub-exponential is known [12, 16]), it is relevant if a strongly polynomial-time algorithm exists for some version of our river embedding problem. In this section we show that this is the case if the elevations of only the river vertices can be changed. This case is related to the stream burning approach mentioned for DEMs in the introduction. However, we may also raise river vertices, and we minimize total change. Note that although the corresponding LP is sparse, inequalities can have three variables (due to the introduction of z_i^+ and z_i^-), while a strongly polynomial-time solution for LP exists when inequalities have at most two variables [3, 18].

To guarantee feasibility, we will assume that no triangle has three river vertices, nor two river vertices that are not endpoints of the same river edge. In this setting, different rivers do not influence each other, and we can concentrate on one river only. We ignore river merging, although our algorithm can be adapted to handle this. Let $v_1, \ldots, v_n$ be the river vertices of one river, listed from (intended) source to sink. We process the river top-down. After processing v_i, we will know *for every possible elevation of* v_i, the minimum total displacement of $v_1, \ldots, v_i$ that makes $\overline{v_1v_2}, \ldots, \overline{v_{i-1}v_i}$ valley edges with correct flow. This is described by a cost function $C_i(h)$, where h is the height of v_i. The minimum attained by $C_n(h)$ is the minimum cost needed. It seems that the problem can be solved with dynamic programming, but the function $C_n(h)$ can have exponentially many breaks, which causes complications.

4.1 Properties of the cost function

Assume that the river edges between $v_1, \ldots, v_i$ have been processed, and we want to turn $\overline{v_iv_{i+1}}$ into a valid river edge as well. Assume we know the cost function $C_i(h)$, and assume further that $C_i(h)$ is convex. Based on this information we will compute $C_{i+1}(h)$. Note that $C_{i+1}(h)$ depends on the coordinates of v_{i+1}, the coordinates of v_i, the coordinates of the wingtips of the two triangles incident to the river edge $\overline{v_iv_{i+1}}$, and the cost function $C_i(h)$. It depends on the coordinates of other vertices only via the cost function $C_i(h)$. Denote by h_i^* the height of v_i that minimizes the total height change of $v_1, \ldots, v_i$, so $C_i(h)$ attains its minimum at h_i^*.

We first look at the situation of edge $\overline{v_iv_{i+1}}$ only, ignoring the other edges from v_1 to v_i. Consider the combinations of heights of v_i and v_{i+1} that make $\overline{v_iv_{i+1}}$ a valid edge: these are the ones that satisfy the flow and valley constraints. We can view this in the (h_{i+1}, h_i)-plane, where h_{i+1} gives the height of v_{i+1} and h_i gives the height of v_i. The height value pairs (h_{i+1}, h_i) that result in a valid edge define a feasible region F_{i+1} in the (h_{i+1}, h_i)-plane. The flow constraint (in the weak sense) gives the inequality $h_{i+1} \leq h_i$. The two (weak) valley constraints give rise to various possible shapes of F_{i+1}, but F_{i+1} is always unbounded.

We introduce a function $H_i(h)$, which gives the optimal height of v_i as a function of the height h of v_{i+1}. $H_i(h)$ implicitly considers all the displacement needed to have a valid river from v_1 to v_{i+1}, given that v_{i+1} is at height h. If no valid river from v_1 to v_{i+1} exists when v_{i+1} is at height h, then $H_i(h)$ is undefined.

The shape of $H_i(h)$ depends on the feasible region F_{i+1} and the optimal height h_i^* for v_i, see Fig. 7. For a given height h for v_{i+1}, the cheapest option—if possible—is to have v_i at h_i^*. However, the combination may not be feasible (Fig. 8 shows an example). In that case, since we assumed

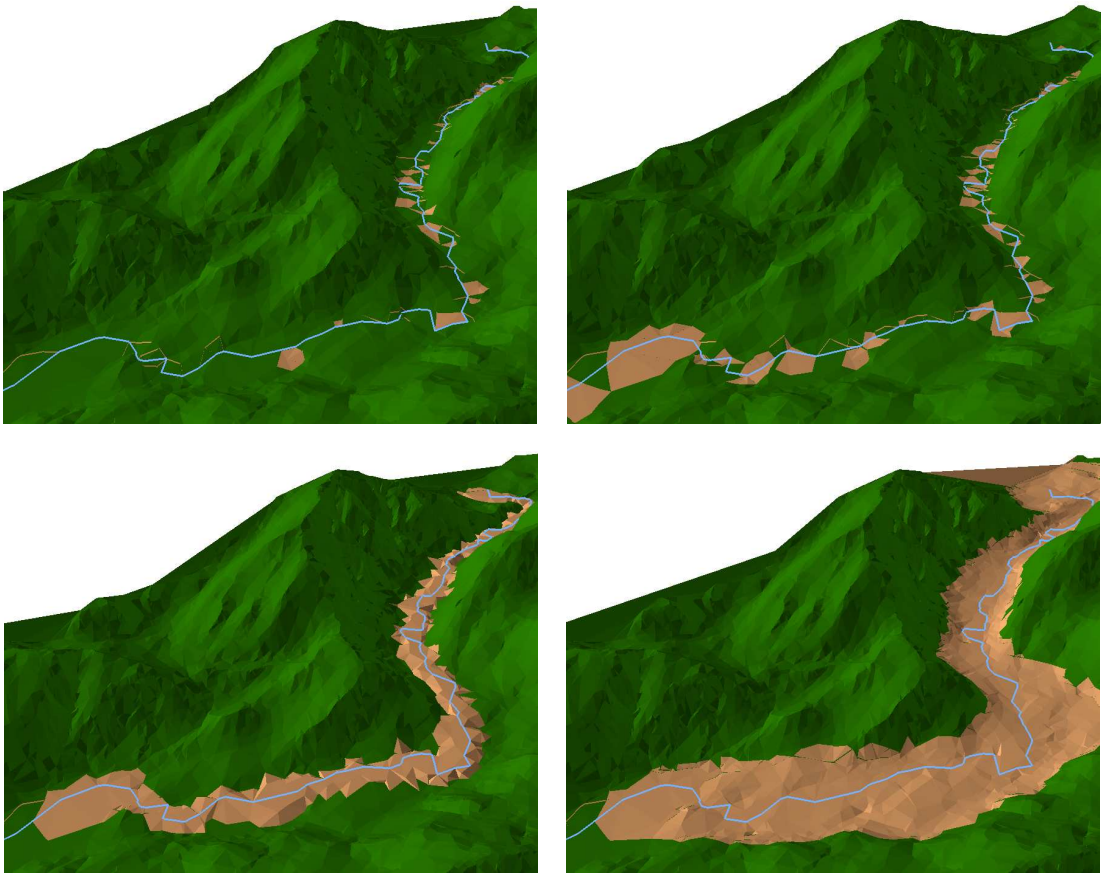

Figure 4: Part of the Goat River after applying four variants of the algorithm. Changed parts are shown in brown (light). From left to right, top to bottom: VF, VFR 10%, VFR 1°, VFRS 1°.

that $C_i(h)$ is convex, the best option is the height closest to h_i^* that is in the feasible region. This implies that H_i is composed of parts of the boundary of F_{i+1} and possibly a horizontal segment in the interior of F_{i+1} where $H_i(h) = h_i^*$. If such a horizontal segment exists, we define h_{i+1}^l and h_{i+1}^r to be the smallest and largest values of the height of v_{i+1} such that $(h_{i+1}^l, h_i^*) \in F_{i+1}$, resp. $(h_{i+1}^r, h_i^*) \in F_{i+1}$. If the horizontal segment is not bounded to the left, h_{i+1}^l is undefined, and if $(h_i = h_i^*) \cap F_{i+1} = \emptyset$, then h_{i+1}^l and h_{i+1}^r are both undefined. Since F_{i+1} is the intersection of three halfplanes, there are only a few possible shapes for $H_i(h)$ (see Fig. 7). Clearly, $H_i(h)$ has constant description size.

We define $C_{i+1}(h)$ as a function of $C_i(h)$. The function $C_{i+1}(h)$ is essentially a combination of the local cost plus the previous function after some transformation. Recall that h_{i+1}^0 denotes the initial height of vertex v_{i+1}. We have:

$$C_{i+1}(h) = |h - h_{i+1}^0| + C_i(H_i(h))$$

From now on we discuss the case of the shape shown in Fig. 7(a). The other shapes are similar but easier, as Fig. 7(a) is the only case that can cause $C_{i+1}(h)$ to have twice the number of break points in $C_i(h)$. We also assume that $h_{i+1}^r < h_{i+1}^0$; again the other cases are analogous. We have:

$$H_i(h) = \begin{cases} h_i^* - \lambda' \cdot (h_{i+1}^l - h) & h < h_{i+1}^l \\ h_i^* & h_{i+1}^l \le h \le h_{i+1}^r \\ h_i^* + \lambda'' \cdot (h - h_{i+1}^r) & h_{i+1}^r < h \end{cases}$$

for some $\lambda' > 0$ and $\lambda'' < 0$ defined by the slopes in $H_i(h)$.

To ease the notation, let $\tilde{C}_i(h) = C_i(h_i^* - \lambda' \cdot (h_{i+1}^l - h))$ and $\hat{C}_i(h) = C_i(h_i^* + \lambda'' \cdot (h - h_{i+1}^r))$.

We now define the cost function C_i recursively, with $C_1(h) = |h_1^0 - h|$ as the base case. For $1 \le i \le n-1$, we note that $C_{i+1}(h)$ is defined as:

$$C_{i+1}(h) = \begin{cases} (h_{i+1}^0 - h) + \tilde{C}_i(h) & h < h_{i+1}^l \\ (h_{i+1}^0 - h) + C_i(h_i^*) & h_{i+1}^l \le h \le h_{i+1}^r \\ (h_{i+1}^0 - h) + \hat{C}_i(h) & h_{i+1}^r < h \le h_{i+1}^0 \\ (h - h_{i+1}^0) + \hat{C}_i(h) & h_{i+1}^0 < h \end{cases} \quad (1)$$

The next lemma shows that C_i is convex, so C_i has only one minimum, and it specifies where this minimum occurs. Similar lemmas exist for the other cases (shape of $H_i(h)$ and relative position of h_{i+1}^0).

LEMMA 1. *$C_{i+1}(h)$ is a convex piecewise linear function that has its minimum either at the break point $h = h_{i+1}^0$, or at a break point of $\hat{C}_i(h)$. Moreover, if it occurs at a break point of $\hat{C}_i$, it is at a point where the value of the slope of $\hat{C}_i(h)$ changes from below 1 to at least 1.*

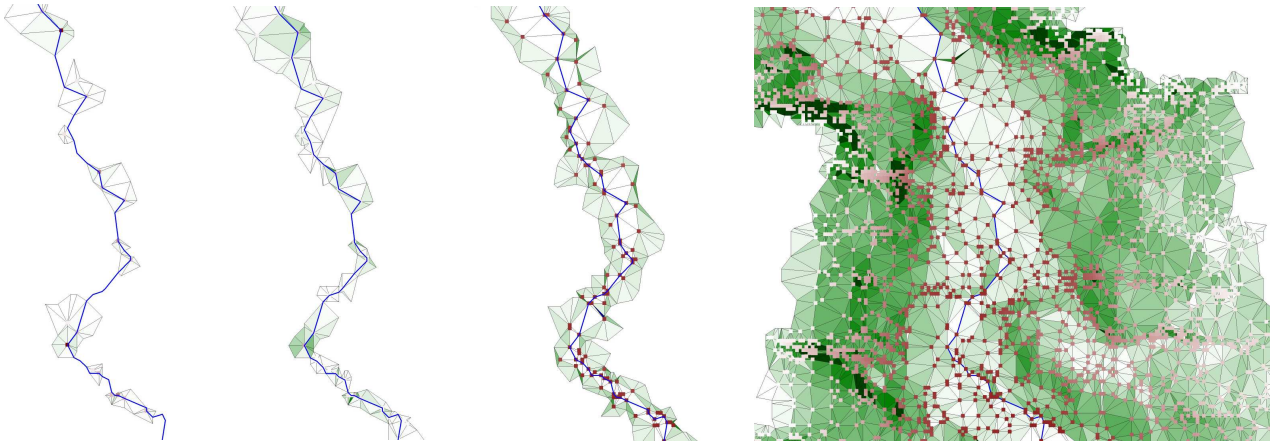

Figure 5: Extent of change for four different variants, on a portion of the Kowesas River. Shades of green indicate change in triangle slope (the darker, the larger), shades of red show vertex elevation change. From left to right: VF, VFR 10%, VFR 1°, VFRS 1°.

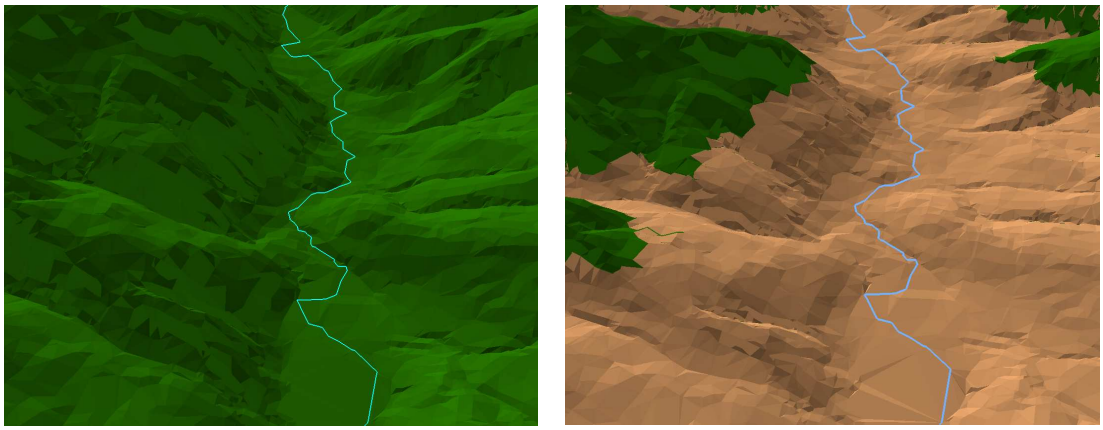

Figure 6: Terrain view of a portion of the Kowesas River, corrected using the VFRS 1° variant (the portion shown corresponds to the rightmost figure in Fig. 5). Left: original terrain. Right: terrain computed by the VFRS 1° variant, with changed areas colored.

PROOF. We prove this by induction. The base case corresponds to $C_1(h) = |h_1^0 - h|$ and clearly holds. For the general case, suppose that $C_i(h)$ is a convex piecewise linear function with its only minimum at h_i^*. We will show that the result holds also for $C_{i+1}(h)$.

If $C_i(h)$ is piecewise linear and convex, it is easy to verify that $\tilde{C}_i(h)$ and $\hat{C}_i(h)$ are also piecewise linear and convex, with their minima at h_{i+1}^l and h_{i+1}^r, respectively. Therefore $C_{i+1}(h)$, as a whole, is piecewise linear, and any minimum must occur at a break point. The break points in $C_{i+1}(h)$ all correspond to breaks of $C_i(h)$, except for $h = h_{i+1}^0$. It is easy to see that there is a break at $h = h_{i+1}^0$ because there is a slope difference of exactly 2 between the functions immediately to the left and right of h_{i+1}^0.

Moreover, we can observe the following about each of the four pieces of C_{i+1}. The first piece has negative (increasing) slope, because $\tilde{C}_i(h)$ is always to the left of its minimum. As h approaches h_{i+1}^l, $\tilde{C}_i(h)$ decreases (it gets closer to its minimum), and at the same time (because $\tilde{C}_i(h)$ is convex), the slope increases. The second part has slope exactly -1. Regarding the third and fourth parts, $\hat{C}_i(h)$ is always evaluated for $h > h_{i+1}^r$, thus as h increases, $\hat{C}_i(h)$ also increases, and so does its slope. Let $t(h)$ denote the slope of $\hat{C}_i$ at h. It follows from Equation (1) that the slope of $C_{i+1}(h)$ for $h_{i+1}^r \leq h$ is $(t(h) - 1)$ up to h_{i+1}^0 and $(t(h) + 1)$ afterwards. Hence, the minimum must either be at $h = h_{i+1}^0$ or at the break point where $(t(h) - 1)$ goes from negative to positive, or equivalently, when $t(h)$ goes from < 1 to ≥ 1. Note that the break point can be at h_{i+1}^r. □

4.2 The algorithm

Based on the structure of C_i, we describe an algorithm to determine $C_n(h_n^*)$, the optimum cost. We continue assuming that the shape of $H_i(h)$ is as in Fig. 7(a), and $h_{i+1}^r < h_{i+1}^0$.

Unfortunately, computing an explicit representation for each function $C_i(h)$ cannot be done efficiently, because C_i can have an exponential number of break points. This is because when the shape of $H_i(h)$ is as shown in Fig. 7(a), two copies of $C_i(h)$, scaled and translated, may be needed to define $C_{i+1}(h)$. This would cause C_{i+1} to have roughly twice as many break points as C_i. However, we can use the recursive definition of Equation (1) to store a constant-size

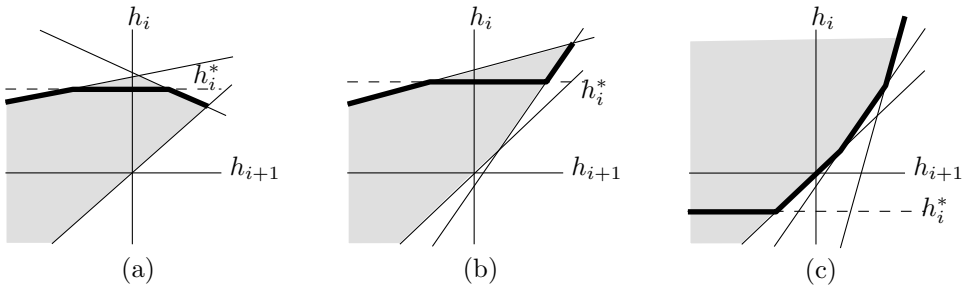

Figure 7: Three different possible shapes for $H_i(h)$ (drawn with thick edges). Case (a) is the most challenging one, because part of C_i is used twice in C_{i+1}.

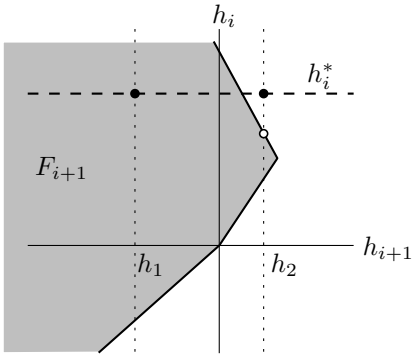

Figure 8: The pair (h_1, h_i^*) is feasible, whereas (h_2, h_i^*) is not. The white circle shows the height for v_i with lowest cost if v_{i+1} has height h_2.

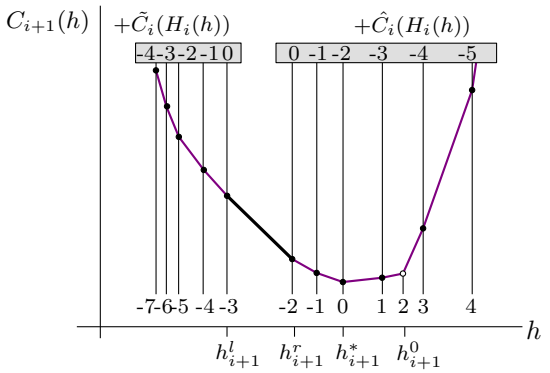

Figure 9: Mapping break points of C_{i+1}. The break point that corresponds to h^0_{i+1} (in white) is the only one that is not mapped to one in C_i.

representation of $C_{i+1}(h)$ based on $C_i(h)$, which allows us to evaluate $C_{i+1}(h)$ in $O(n)$ time.

We describe the algorithm recursively. The basic idea is as follows. Assume we can evaluate C_i at any h, and that we also know h_i^*. Then, using Equation (1), we can determine the value of $C_{i+1}(h)$ for any h. We only need the values of h^0_{i+1}, λ', λ'', h^l_{i+1}, and h^r_{i+1}. The initial height h^0_{i+1} is part of the input, and λ', λ'' depend only on $H_i(h)$. Finally, h^l_{i+1}, and h^r_{i+1} can be computed based on F_{i+1} and h_i^*, where h_i^* is assumed to be known. Note that even though Equation (1) defines C_{i+1} in terms of $\tilde{C}_i$ and $\hat{C}_i$, when more convenient we will refer to C_i instead. It remains to describe how h^*_{i+1} is determined.

Lemma 1 states that h^*_{i+1} occurs at h^0_{i+1} or at a break point of $\hat{C}_i$. Since $\hat{C}_i$ is a linearly transformed version of a part of C_i, if we can evaluate C_i at its break points efficiently, we can find h^*_{i+1} by a binary search on the slope of $\hat{C}_i$ to find the two consecutive segments whose slopes are < 1 and ≥ 1. Let the break points of C_i be numbered as follows: break point 0 corresponds to h_i^*, the break points to the right of h_i^* are numbered increasingly $(1, 2, \dots)$, and the ones to its left decreasingly $(-1, -2, \dots)$.

More precisely, we define our inductive hypothesis as follows: assume that for any break point number of C_i, we can compute, in $O(n)$ time, its h-coordinate and the value of C_i at that break point. Clearly, its minimum h_i^* can be obtained by simply asking for break point 0. Note that if we know this for C_i, we also know it for $\tilde{C}_i$ and $\hat{C}_i$.

Now we want to be able to answer the same type of query for C_{i+1}. In other words, we need to know how to map break point numbers from C_{i+1} to C_i, see Fig. 9. The break point number where the slope of $\hat{C}_i$ changes from < 1 to ≥ 1 can be found by a binary search as described before. In the same way, we find the two consecutive break points of C_i between which h^0_{i+1} occurs. This allows us to determine h^*_{i+1}, and furthermore, how to map break point numbers of C_{i+1} to the ones of C_i. By definition, break point 0 of C_{i+1} corresponds to h^0_{i+1} or to the break point where the slope changes from < 1 to ≥ 1. The other mappings are straightforward: based on the two break point numbers around h^0_{i+1}, the mappings for h^l_{i+1} and h^r_{i+1} can be deduced. Following the structure of Equation (1), any other break point can be mapped.

For the running time, we note that for each of the $O(n)$ river edges we spend $O(n^2)$ time to find h_i^* and the breaks around h^0_{i+1}. This is because each binary search, among $O(2^n)$ break points, takes $O(n)$ time, and each query to the previous C_i also takes $O(n)$ time. Therefore the total running time is $O(n^3)$. This time bound assumes that we can handle basic operations on integers of up to exponential size in $O(1)$ time. In the standard models of computation, we should charge $O(n)$ time for operations on exponential-size integers. This leads to a running time of $O(n^4)$. Although we did not explicitly describe it, our algorithm can handle merging of rivers as well.

THEOREM 2. *The minimum elevation change of river vertices on a terrain, to satisfy the valley and flow constraints, can be determined in $O(n^4)$ time, where n is the number of river vertices.*

5. CONCLUSIONS AND FUTURE WORK

We discussed correcting a polyhedral terrain so that a known river follows valley edges: edges whose local shape is like those of rivers. We formulated the problem of minimizing the total elevation change to achieve this (and other terrain corrections) as a linear program, which makes it easy to implement and fast in practice. Experiments show that the approach works very well. Depending on the exact criteria, vertices need be moved only a little, and the new terrain is hydrologically more correct. A special case was shown to be solvable in strongly polynomial time.

To obtain hydrologically correct terrains, spurious local minima must be removed by other means than linear programming, and one must ensure that the watershed boundaries are correct. Further research is needed to determine how and when local minima are removed, and experiments may reveal whether a change in watershed boundaries really occurs in terrain correction methods. Algorithmically, we expect that our strongly polynomial-time algorithm can be made more efficient.

Acknowledgements The authors thank Oswin Aichholzer, Thomas Hackl, Mike McAllister and Ross Purves for comments and interesting discussions.

6. REFERENCES

[1] J.N. Callow, K.P. Van Niel, and G.S. Boggs. How does modifying a DEM to reflect known hydrology affect subsequent terrain analysis? *J. of Hydrology*, 332:30–39, 2007.

[2] C.-C. Chen and C.A. Knoblock. Conflation of geospatial data. In S. Shekhar and H. Xiong, editors, *Encyclopedia of GIS*, pages 133–140. Springer, 2008.

[3] E. Cohen and N. Megiddo. Improved algorithms for linear inequalities with two variables per inequality. *SIAM J. Comput.*, 23(6):1313–1347, 1994.

[4] M. de Berg, P. Bose, K. Dobrint, M. van Kreveld, M. Overmars, M. de Groot, T. Roos, J. Snoeyink, and S. Yu. The complexity of rivers in triangulated terrains. In *Proc. 8th Canad. Conf. Comput. Geom.*, pages 325–330, 1996.

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

[6] A.U. Frank, B. Palmer, and V.B. Robinson. Formal methods for the accurate definition of some fundamental terms in physical geography. In *Proc. 2nd Int. Symp. on Spatial Data Handling*, pages 583–599, 1986.

[7] R. Hellweger. Agree.aml. Center for Research in Water Resources, The University of Texas at Austin, 1996.

[8] M.F. Hutchinson. Calculation of hydrologically sound digital elevation models. In *Proc. 3th Int. Symp. on Spatial Data Handling*, pages 117–133, 1988.

[9] M.F. Hutchinson. A new procedure for gridding elevation and streamline data with automatic removal of spurious pits. *J. of Hydrology*, 106:211–232, 1989.

[10] M.F. Hutchinson. ANUDEM Version 5.1. The Australian National University, Canberra, Australia, 1994.

[11] S.K. Jenson and J.O. Dominique. Extracting topographic structure from digital elevation data for geographic information systems. *Photogrammatic Engineering and Remote Sensing*, 54:1593–1600, 1988.

[12] G. Kalai. A subexponential randomized simplex algorithm (extended abstract). In *Proc. 24th ACM Symposium on Theory of Computing*, pages 475–482, 1992.

[13] Y. Liu and J. Snoeyink. Flooding triangulated terrain. In P.F. Fisher, editor, *Developments in Spatial Data Handling, Proc. 11th Int. Sympos.*, pages 137–148, Berlin, 2004. Springer.

[14] D.R. Maidment. GIS and hydrologic modeling. In M.F. Goodchild, B.O. Parks, and L.T. Steyaert, editors, *Environmental modeling with GIS*, pages 147–167. Oxford University Press, New York, 1993.

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

[16] J. Matousek, M. Sharir, and E. Welzl. A subexponential bound for linear programming. *Algorithmica*, 16:498–516, 1996.

[17] M. McAllister and J. Snoeyink. Extracting consistent watersheds from digital river and elevation data. In *Proc. ASPRS/ACSM Annu. Conf.*, 1999.

[18] N. Megiddo. Towards a genuinely polynomial algorithm for linear programming. *SIAM J. Comput.*, 12(2):347–353, 1983.

[19] E.J. Nelson, N.L. Jones, and W. Miller. Algorithm for precise drainage-basin delineation. *J. Hydraulic Engineering*, 120:298–312, 1994.

[20] A. Saalfeld. Conflation: automated map compilation. *Int. J. of Geographical Information Systems*, 2:217–228, 1988.

[21] R.I. Silveira and R. van Oostrum. Flooding countries and destroying dams. In *Proc. 10th Workshop on Algorithms and Data Structures*, volume 4619 of *Lecture Notes in Computer Science*, pages 227–238. Springer, 2007.

[22] D.M. Theobald and M.F. Goodchild. Artifacts of TIN-based surface flow modelling. In *Proc. GIS/LIS*, pages 955–964, 1990.

[23] E.R. Vivoni, V.Y. Ivanov, R.L. Bras, and D. Entekhabi. Generation of triangulated irregular networks based on hydrological similarity. *J. Hydrologic Engineering*, 9:288–302, 2004.

[24] S. Yu, M. van Kreveld, and J. Snoeyink. Drainage queries in TINs: from local to global and back again. In M.J. Kraak and M. Molenaar, editors, *Advances in GIS research II: Proc. of the 7th Int. Symp. on Spatial Data Handling*, pages 829–842, 1997.