Cite as: Barnes, Lehman, Mulla. "An Efficient Assignment of Drainage Direction Over Flat Surfaces In Raster Digital Elevation Models". Computers & Geosciences. Vol 62, Jan 2014, pp 128–135. doi: "10.1016/j.cageo.2013.01.009".

An Efficient Assignment of Drainage Direction Over Flat Surfaces in Raster Digital Elevation Models

Richard Barnes[a,*], Clarence Lehman[c], David Mulla[c]

[a] *Ecology, Evolution, & Behavior, University of Minnesota, USA*
[b] *College of Biological Sciences, University of Minnesota, USA*
[c] *Soil, Water, and Climate, University of Minnesota, USA*

Abstract

In processing raster digital elevation models (DEMs) it is often necessary to assign drainage directions over flats—that is, over regions with no local elevation gradient. This paper presents an approach to drainage direction assignment which is not restricted by a flat's shape, number of outlets, or surrounding topography. Flow is modeled by superimposing a gradient away from higher terrain with a gradient towards lower terrain resulting in a drainage field exhibiting flow convergence, an improvement over methods which produce regions of parallel flow. This approach builds on previous work by Garbrecht and Martz (1997), but presents several important improvements. The improved algorithm guarantees that flats are only resolved if they have outlets. The algorithm does not require iterative application; a single pass is sufficient to resolve all flats. The algorithm presents a clear strategy for identifying flats and their boundaries. The algorithm is not susceptible to loss of floating-point precision. Furthermore, the algorithm is efficient, operating in $O(N)$ time whereas the older algorithm operates in $O(N^{3/2})$ time. In testing, the improved algorithm ran 6.5 times faster than the old for a 100×100 cell flat and 69 times faster for a 700×700 cell flat. In tests on actual DEMs, the improved algorithm finished its processing 38–110 times sooner while running on a single processor than a parallel implementation of the old algorithm did while running on 16 processors. The improved algorithm is an optimal, accurate, easy-to-implement drop-in replacement for the original. Pseudocode is provided in the paper and working source code is provided in the Supplemental Materials.

Keywords: flat resolution; terrain analysis; hydrology; drainage network; modeling; GIS

1. Background

A digital elevation model (DEM) is a representation of terrain elevations above some common base level, usually stored as a rectangular array of floating-point or integer values. With improvements in remote sensing, LIDAR, and computer performance, DEMs have increased in resolution from thirty-plus meters in the recent past to the sub-meter resolutions becoming available today. Increasing resolution has led to increased data sizes: current data sets are on the order of gigabytes and increasing, with billions of data points. While computer processing and memory performance have increased appreciably, legacy equipment and algorithms suited to manipulating smaller DEMs with coarser resolutions make processing improved

*Corresponding author, 321-222-7637. ORCID: 0000-0002-0204-6040
Email addresses: rbarnes@umn.edu (Richard Barnes), lehman@umn.edu (Clarence Lehman), mulla003@umn.edu (David Mulla)

data sources costly, if not impossible. Therefore, improved algorithms are needed. This paper presents one such algorithm.

DEMs may be used to estimate a region's hydrologic and geomorphic properties, including soil moisture, terrain stability, erosive potential, rainfall retention, and stream power. Many algorithms for extracting these properties require (1) that every cell within a DEM have a defined flow direction and (2) that by following flow directions from one cell to another, it is always possible to reach the edge of the DEM. These requirements are confounded by the presence of depressions and flats within the DEM.

Flats are mathematically level regions of the DEM with no local gradient. Although flats may occur naturally, their presence in a DEM is also frequently the result of technical issues in the DEM's collection and processing, such as from biased terrain reflectance, conversions from floating-point to integer precision, noise removal, low vertical resolution, or low horizontal resolution, among other possibilities. [Nardi et al., 2008; Garbrecht and Martz, 1997] Flats may also be produced when depressions—inwardly-draining regions of the DEM which have no outlet, also known as pits—are filled in. Depression-filling algorithms often increase the size and number of flats in a DEM, with one study finding 28–162% more cells in flats after depressions were filled. [Nardi et al., 2008] In mountainous terrain the total number of cells in flats may be small, while in level or agricultural terrain the number may be a significant fraction of the DEM.

Ultimately, it is inevitable that the DEM will have flats. The algorithm by Garbrecht and Martz [1997]—henceforth G&M—described below presents one means of resolving them. G&M is an improvement over previous algorithms (see Tribe [1992]), offering a simple method to produce realistic flow patterns over flat terrain. More recent approaches using breadth-first search [Liang and MaCkay, 2000], priority-first search in weighted-graphs [Jones, 2002], cellular automata [Coppola et al., 2007], and variable elevation increments [Jana et al., 2007], among other methods (see Zhang and Huang [2009]), may provide superior results but are often difficult to describe, implement, and test, leading to low adoption. Soille et al. [2003] is the only work of which the authors are aware which directly improves on G&M (by eliminating the need for iterative applications of the algorithm). However, the method is described in terms of field-specific knowledge and little explanation of implementation is provided.

While most algorithms can be improved, G&M's direct use in a number of studies (e.g. Alsdorf [2003]; Clarke et al. [2008]; Clennon et al. [2007]; Coe et al. [2008]; Fang et al. [2010]; Brardinoni and Hassan [2006]; Kite [2001]; Lin et al. [2006]; Phillips and Slattery [2007]; White et al. [2004]), its inclusion in DEM processing packages such as TOPAZ [Garbrecht and Martz, 1997] and TauDEM, as well as its inclusion as a preprocessing step in a number of other algorithms (e.g. Mackay and Band [1998]; Miller [2003]; Stepinski and Vilalta [2005]; Tarboton and Ames [2001]; Toma et al. [2001]; Turcotte et al. [2001]; Vogt et al. [2003]; Wallis et al. [2009]; Zhang and Huang [2009]), make it an important target for improvement.

This paper presents easy-to-implement improvements to G&M which realize substantial gains in efficiency and accuracy.

2. The Algorithm

2.1. Overview

As noted by Garbrecht and Martz [1997], a flat will always be surrounded by two types of terrain: higher and lower. This is true even at the edges of the DEM because the improved algorithm assumes that the DEM's edge cells direct their flow outwards. It is natural to suppose that water near lower terrain will flow towards that lower terrain, whereas water near higher terrain will flow away from that higher terrain. This reasoning can be applied inductively to the entirety of a flat.

Using only the gradient away from higher terrain results in convergent, inward flow, as shown in Fig. 1e. Taken alone, this gradient would be unsuitable for further DEM processing because such flow does not in general drain from the flat. Using only the gradient towards lower terrain results in parallel flows which are guaranteed to drain the flat, as shown in Fig. 2f. Taken alone, this gradient would be unsuitable for further DEM processing because parallel flow patterns are uncommon in nature and, when present, tend to quickly evolve towards convergent flows. However, when these gradients combine, they produce a realistic, convergent flow pattern which is guaranteed to drain the flat.

If the flat is not adjacent to lower terrain, it cannot drain and therefore cannot be resolved by either the improved algorithm described in this paper or by G&M. However, unlike G&M, the improved algorithm will flag such flats and refuse to work on them. Aside from this requirement, the flat may be surrounded by any arbitrary combination of terrain.

Since the algorithm is used as a preprocessing step, its results must be available to subsequent processes. In G&M this is accomplished by adding increments of 10^{-5} to the DEM's elevations. Ideally, these increments are negligably small compared to the vertical resolution of the DEM and are sufficient to direct flow while having no other impacts on the processing of the DEM.

However, the precision of DEMs has increased considerably since G&M was designed and it is now possible to find DEMs which use the full width of single-precision floating-point storage units. In such a case, there is no negligable increment which can be added—every increment is a significant alteration of the DEM. In this case, incrementing a cell to resolve a flat may cause it to rise above surrounding cells which were formerly higher than it was, corrupting important information about the landscape. Ultimately, G&M provides no guarantees regarding its effect on a DEM.

Using double-precision floating-point storage is one solution, but, if a flat is particularly large or the DEM particularly precise, increments may still become significant. Using the larger data type also undesirably increases the storage size of the DEM in all subsequent operations.

Therefore, the improved algorithm generates an *elevation mask* which is used to determine flow directions; the DEM's elevations themselves are left unaltered. If it is necessary to alter the DEM itself, a simple modification to the algorithm—discussed below—performs the alteration using what is guaranteed to be the smallest possible increment.

The algorithm assumes that the edge cells of the DEM which have defined elevations also have or can have defined flow directions—usually such flow is directed outwards, so the DEM drains itself. This assumption means that edge cells need not be treated as special cases.

The algorithm is divided into four steps, which are detailed below and described in pseudocode by Algorithm 1. (1) Each unique flat is identified and its edge cells grouped into two categories: those edge cells adjacent to higher terrain and those edge cells adjacent to lower terrain. (2) The gradient away from higher terrain will be constructed starting with the edge cells adjacent to higher terrain. During this step, the maximal number of increments for each flat will be noted. (3) The gradient towards lower terrain will be constructed starting with the edge cells adjacent to lower terrain. The results of Step (2) are superimposed during this step. (4) The final flow directions are determined using the gradient developed in Steps 2 and 3.

2.2. Step 1: Flat Identification

In order to guarantee that each flat drains and to calculate the gradient away from higher terrain in $O(N)$ time, the algorithm requires that each unique flat be identified.

The algorithm assumes that a separate flow direction algorithm of the user's choice has already been run and has assigned flow directions to each cell. Algorithm 2 provides an example of how this might done. Some cells will have no place to drain to because they are surrounded by cells of the same or higher elevation; these cells will be given a special NoFlow value indicating that they do not have a defined flow direction. Let the data structure holding information regarding flow directions be called *FlowDirections*.

In order to seed the gradient routines, the algorithm requires a list of those cells of the flat which are adjacent to higher and lower terrain. To obtain this, the improved algorithm scans over each cell c of *FlowDirections* (see Algorithm 3) and, where appropriate, adds c to a queue for further processing.

c is a "high edge" cell if (1) c does not have a defined flow direction and (2) c has at least one neighbor n at greater elevation. If these properties are true of c it is added to the first-in, first-out (FIFO) queue *HighEdges*.

c is a "low edge" cell if (1) c has a defined flow direction, (2) c has at least one neighbor n at the same elevation, and (3) this n does not have a defined flow direction. If these properties are true of c it is added to the FIFO queue *LowEdges*.

Since every cell surrounding a flat must be either a low edge or a high edge cell, if none of either are found, then the DEM has no flats. If there are no low edge cells, but there are some high edge cells, then

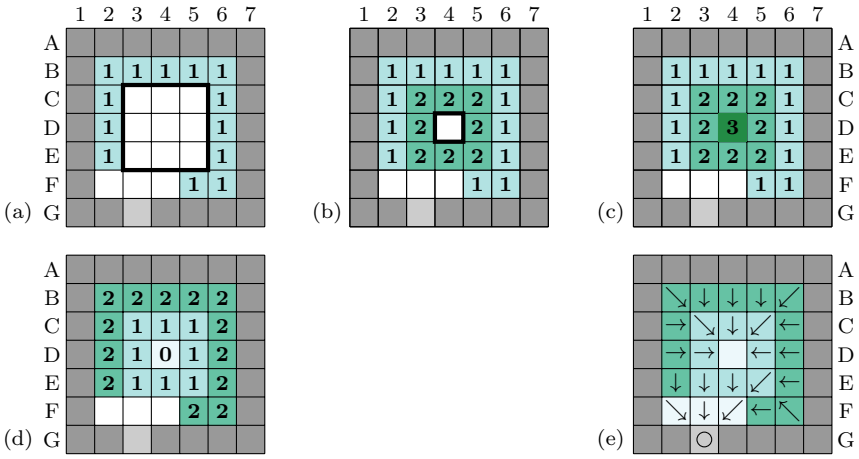

Figure 1: Step 2: gradient away from higher terrain (see §2.3 and Algorithm 5). The number of increments applied to each cell is shown. (a) Numbers of increments after the first iteration. This first set of incremented cells are the edge cells adjacent to higher (but not to lower) terrain; they were found in Step 1 and stored in the queue *HighEdges*. After being incremented, each of these cells will add its unincremented neighbors—those just beyond the thick black line—to the queue. In (b) the neighbors of the edge cells have been popped off of the queue and incremented, advancing the black line. This process continues through (c), which shows the final number of increments, though the gradient is the inverse of what is desired. By noting that at most 3 increments were applied to this flat it is possible to obtain the desired gradient (d) by subtracting each cell of (c) from 3. This is done in Step 3 (Fig. 3). Note that the cells (F2, F3, F4) adjacent to lower terrain are ignored. (e) shows the resulting drainage field. The flow directions of cells E3–E5 will be uniquely determined by a gradient towards lower terrain in Step 3 (see Fig. 3d).

the DEM has flats but they cannot be drained. In both such instances, the algorithm indicates this and proceeds no further.

Otherwise, the algorithm then labels each unique flat (see Algorithm 4). Let the data structure holding these labels be called *Labels*, and let it be initialized to a value NoLabel. To apply the labels, each cell c in the *LowEdges* queue is used as the seed to a flood fill algorithm. If c has not yet been labeled then a new label l is given to it, its elevation e is noted, and all cells which can be reached from c while traversing only cells of elevation e are given the same label l. A FIFO queue could be used for this, or a D8 variant of Heckbert [1990], or one of many other solutions to the "connected components" problem (Grana et al. [2010] provide a good overview). Recursive solutions should be avoided in implementations using languages without proper tail recursion, as stack overflows are likely when processing large DEMs. In the end each flat will have a unique label and all member cells of a flat will bear its label.

The *LowEdges* queue will not contain any non-draining flats; however, such flats must be eliminated from the *HighEdges* queue. To do so, each cell in *HighEdges* is considered and, if its corresponding label is *NoLabel*, it is removed (this is correct since all high edge cells must eventually connect with a low edge cell and each low edge cell has been used as a labeling seed). If any cells are eliminated from *HighEdges*, it means that some of the DEM's flats will not drain.

If a flat does not drain, then it is a depression and may be resolved through depression-filling, among other methods. *Barnes et al.* [in progress] describes an efficient algorithm for doing this in $O(m \log m)$ time, where $m \leq N$; other less efficient solutions operating in $O(N \log N)$ time and higher are described by Wang and Liu [2006] and Planchon and Darboux [2002], respectively.

The results of Step 1 are (1) two queues—*HighEdges* and *LowEdges*—containing all and only the drainable flat cells bordering regions of higher elevation and the flat cells bordering regions of lower elevation, respectively; and (2) a structure *Labels* mapping each flat cell to a label indicating which flat it is a part of. At this point, all flats referred to by cells in either queue are guaranteed to drain following the completion of the algorithm.

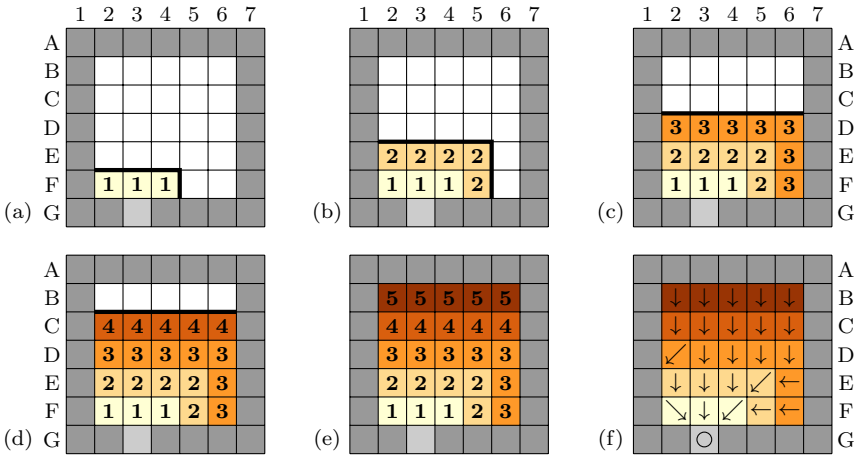

Figure 2: Step 3: gradient towards lower terrain (see §2.4 and Algorithm 6). The number of increments applied to each cell is shown. (a) Numbers of increments after the first iteration. This first set of incremented cells are the edge cells adjacent to lower terrain; they were found in Step 1 and stored in the queue *LowEdges*. After being incremented, each of these cells will add its unincremented neighbors—those just beyond the thick black line—to the queue. In (b) the neighbors of the edge cells have been popped off of the queue and incremented, advancing the black line. This process continues through (c), (d), and (e). (e) shows the final number of increments. (f) shows the resulting drainage field. There is a second part to Step 3, shown below, which combines the gradients.

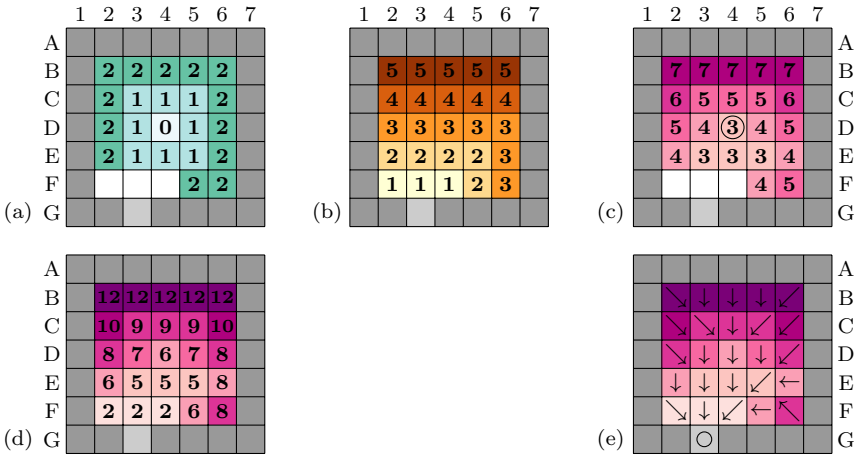

Figure 3: Step 3, continued: combining the gradients (see §2.4 and Algorithm 6). (a) The gradient away from higher terrain resulting from Step 2. (b) The gradient towards lower terrain resulting from the first part of Step 3. (c) Superposition of Steps 2 and Steps 3 in the style of G&M, note that a new flat has been created at cell D4 (circled); this leads to flow direction ambiguity. (d) Superposition as in the improved algorithm. By giving the gradient towards lower terrain twice the weight as the gradient away from higher terrain, all ambiguities are eliminated and convergent flow patterns produced. (e) shows the resulting drainage pattern.

2.3. Step 2: Gradient away from higher terrain

The gradient away from higher terrain is constructed by growing elevated terrain inwards from edge cells adjacent to higher terrain in a breadth-first expansion using a FIFO queue, a process described by Fig. 1 and Algorithm 5.

To begin this process, a special "iteration marker" is put at the end of the queue *HighEdges* developed in Step 1. An iteration counter I is set to 1. Next, each cell c is popped off of the queue in turn and its increment set to I in a structure *FlatMask*. All of c's unincremented neighbors which have the same label as c (implying they are part of the same flat) and have undefined flow directions are added to the end of

the queue. If c is the iteration marker, then a new iteration marker is pushed onto the queue and I is incremented by one.

Additionally, the maximal number of increments applied to each flat must be recorded. Therefore, an array *FlatHeights* with length equal to the number of flats is initialized to 0. When a cell is incremented, the cell's flat is looked up in *Labels* and the flat's corresponding value in *FlatHeights* is updated to I.

If a cell is adjacent to both lower and higher terrain, that cell is considered to be adjacent *only* to the lower terrain and it is not considered in this step. The fulfillment of this constraint is guaranteed by Step 1.

Step 2 terminates when *HighEdges* contains only an iteration marker. The results of are (1) a structure *FlatMask* indicating each cell's D8 distance away from higher ground and (2) an array *FlatHeights* indicating the maximal distance any point in the flat is from higher ground. The values in *FlatMask* are the inverse of the desired gradient, but the values in *FlatHeights* can be used to invert the inverse, thereby producing the intended gradient (note the transition between Fig. 1c and Fig. 1d).

2.4. Step 3: Gradient towards lower terrain

The gradient towards lower terrain is built via backgrowth from outlet(s) in the flat by working inwards from edge cells adjacent to lower terrain using a FIFO queue, a process described by Fig. 2 and Algorithm 6.

This step is analogous to Step 2, with a few twists. First, every cell in *FlatMask* is made negative. Now, any cell which has a value of 0 is not part of a flat and all cells which are negative have yet to be processed. Second, a special "iteration marker" is put at the end of the queue *LowEdges* developed in Step 1. An iteration counter I is set to 1.

As each cell c is popped off of the queue, all of c's unincremented neighbors which have the same label as c and have undefined flow directions are added to the end of the queue. c is then dealt with as specified below. If c is the iteration marker, then a new iteration marker is pushed onto the queue and I is incremented by one.

As each cell is popped, two things happen. If the cell's value in *FlatMask* was incremented by the previous step, then it will have a negative value. The maximal height of the cell's flat—previously stored in *FlatHeights* in Step 2—is added to this value, thereby inverting the gradient away from higher terrain. Finally, the cell's value in *FlatMask* is always incremented by $2I$. This is the twice the gradient towards lower terrain and enforces unambiguous drainage from the flat without the iterative applications required by G&M, as discussed below.

Step 3 terminates when *LowEdges* contains only an iteration marker. The results are that *FlatMask* is the superposition of the gradient away from higher terrain and the gradient towards lower terrain.

2.5. Step 4: Determination of Flow Directions

In G&M, exceptional situations occurred in which the resolution of flats created new flats (as in Fig. 3c); these situations were resolved by iterative application of G&M using ever-smaller increments. However, loss of significance during this procedure could lead to unresolvable flats. In the improved algorithm presented in this paper, the gradient towards lower terrain is given twice the weight—made twice as steep—as the gradient away from higher terrain (see Fig. 3d). The gradient towards lower terrain, which is guaranteed to drain the flat, will then dominate ambiguous situations ensuring that all flats drain and that no iterative applications are necessary—a significant improvement over G&M. Soille et al. [2003] propose a much more complex method of achieving the same end by using an "inverse geodesic distance" as a mask, but provide few implementation details.

Finally, the flow directions algorithm is re-run on the DEM and flat cells with previously undefined flow directions are now defined using the increments stored in *FlatMask*. Because the increments are relative to the base elevation of a flat, cells from flats with differing labels must not be allowed to flow into each other unless their actual elevations permit it (which will only be the case for a flat's low edge cells). Flow directions can be determined using D8, D∞ [Tarboton, 1997], or any other such method.

Alternatively, if it is necessary to alter the DEM itself, as in G&M, the smallest possible increments should be used. In `C` and `C++` this can be done using the NextAfter function defined by the `C99` and `POSIX` standards, which increases or decreases a floating-point number by the smallest possible increment

Algorithm	100^2 cells	400^2 cells	700^2 cells	$1{,}000^2$ cells
Improved	0.004 s	0.06 s	0.18 s	0.41 s
G&M	0.026 s	1.82 s	12.5 s	45.4 s
Speed-up	6.5 x	30 x	69 x	111 x

Figure 4: Run-time comparisons of the improved algorithm versus the old. "Speed-up" indicates how many times faster the improved algorithm ran than G&M. Columns are labeled according to the number of cells in the test flat and run-times are in seconds. For comparison, in a 1m DEM an area of 100 x 100 cells is equivalent to 2.47 acres while an area of 1000 x 1000 cells is equivalent to 1 km^2 or 100 hectares. Tests were performed using `C++` implementations compiled with full optimizations using a 2.8GHz core. The implementations are included in the Supplemental Materials.

in the direction of a second number; this is far preferable to an arbitrarily defined ϵ. Similar functions exist in other languages. It is also possible to check that this process doesn't cause problems: after a cell is incremented, the only neighboring cells which should be lower than it are those which were lower before it was incremented, unless the neighbors in question are part of the same flat.

3. Speed Comparisons

Since the improved algorithm visits each cell only once per step, it therefore operates in $O(N)$ time. In contrast, it can be shown that G&M operates in $O(N^2)$ time for long, narrow flats and in $O(N^{3/2})$ time for square flats. Thus the improved algorithm described here can be orders of magnitude faster for large flats.

The G&M algorithm was tested against the improved algorithm on a DEM consisting of a large, square flat surrounded on all sides by higher elevation, except for a single-cell drainage in the bottom left, three cells in from the side—similar to the setup used in Fig. 1–3. Several test run times were averaged to yield the times shown in Fig. 4 for DEMs of various sizes; standard deviations were small. The improved algorithm performed significantly faster than G&M. The implementations used for this test are included in the Supplemental Materials.

Wallis et al. [2009] produced an MPI-based parallel implementation of G&M for the TauDEM 5 analysis package. This implementation's flat resolution algorithm was tested against the improved algorithm included in the Supplemental Materials. Both implementations were built in `C++` and compiled with full optimizations; runs were performed on a two-node, 8-core, 2.8GHz machine.

In a test on a depression-filled 3 m DEM of Minnesota's Beauford watershed (2418 x 1636 cells, 26% flats), the parallel implementation's fastest time was 20.37 seconds on 15 processors; the improved algorithm ran in 0.53 seconds on a single processor, 38 times faster in terms of elapsed time and 577 times faster in terms of processing time. On a depression-filled 3 m DEM of Minnesota's Steele County (10891 x 13914 cells, 18% flats), the parallel implementation's fastest time was 53.25 minutes on 16 processors; the improved algorithm ran in 29 seconds, 110 times faster in terms of elapsed time and 1763 times faster in terms of total processing time. Both of the tested regions were agricultural landscapes consisting largely of fields.

4. Coda

An improved algorithm has been detailed to replace an existing algorithm by Garbrecht and Martz (1997). The improved algorithm produces the same result, but has a time complexity of only $O(N)$, whereas the old algorithm has a time complexity of $O(N^{3/2})$ or worse; additionally, the improved algorithm performed significantly faster in actual situations. In addition, the improved algorithm does not require iterative application, is guaranteed to work only on flats which can be drained, and is not vulnerable to floating-point significance problems. Additionally, the improved algorithm is easier to implement than the old. Pseudocode is provided below. The Supplemental Materials contain a reference implementation, several DEMs, and the results of running the reference implementation on these DEMs for use in verifying custom implementations. The Supplemental Materials can be found in an archived format at `http://dx.doi.org/10.1016/j.cageo.2013.01.009` and in an interactive format at `https://github.com/r-barnes/Barnes2013-FlatSurfaces`.

This algorithm is implemented in the RichDEM analysis package, which is available at `http://rbarnes.org/richdem` or via email from the authors.

5. Acknowledgments

Funding for this work was provided by the Minnesota Environment and Natural Resources Trust Fund under the recommendation and oversight of the Legislative-Citizen Commission on Minnesota Resources (LCCMR). Supercomputing time and data storage were provided by the Minnesota Supercomputing Institute. Joel Nelson provided LIDAR DEMs and ArcGIS support; Adam Clark provided feedback on the paper. In-kind support was provided by Justin Konen and Tess Gallagher. We are also grateful to the editors and to an anonymous reviewer.

Alsdorf, D., 2003. Water storage of the central amazon floodplain measured with gis and remote sensing imagery. Annals of the Association of American Geographers 93 (1), 55–66. doi: http://dx.doi.org/10.1111/1467-8306.93105

Brardinoni, F., Hassan, M., 2006. Glacial erosion, evolution of river long profiles, and the organization of process domains in mountain drainage basins of coastal british columbia. Journal of geophysical research 111 (F1), F01013. doi: http://dx.doi.org/10.1029/2005JF000358

Clarke, S., Burnett, K., Miller, D., 2008. Modeling streams and hydrogeomorphic attributes in oregon from digital and field data. JAWRA Journal of the American Water Resources Association 44 (2), 459–477. doi: http://dx.doi.org/10.1111/j.1752-1688.2008.00175.x

Clennon, J., King, C., Muchiri, E., Kitron, U., 2007. Hydrological modelling of snail dispersal patterns in msambweni, kenya and potential resurgence of schistosoma haematobium transmission. Parasitology 134 (05), 683–693. doi: http://dx.doi.org/10.1017/S0031182006001594

Coe, J., Kinner, D., Godt, J., 2008. Initiation conditions for debris flows generated by runoff at chalk cliffs, central colorado. Geomorphology 96 (3-4), 270–297. doi: http://dx.doi.org/10.1016/j.geomorph.2007.03.017

Coppola, E., Tomassetti, B., Mariotti, L., Verdecchia, M., Visconti, G., 2007. Cellular automata algorithms for drainage network extraction and rainfall data assimilation. Hydrological sciences journal 52 (3), 579–592. doi: http://dx.doi.org/10.1623/hysj.52.3.579

Fang, X., Pomeroy, J., Westbrook, C., Guo, X., Minke, A., Brown, T., 2010. Prediction of snowmelt derived streamflow in a wetland dominated prairie basin. Hydrology and Earth System Sciences 14 (6), 991–1006. doi: http://dx.doi.org/10.5194/hessd-7-1103-2010

Garbrecht, J., Martz, L., 1997. The assignment of drainage direction over flat surfaces in raster digital elevation models. Journal of hydrology 193 (1-4), 204–213. doi: http://dx.doi.org/10.1016/S0022-1694(96)03138-1

Grana, C., Borghesani, D., Cucchiara, R., 2010. Optimized block-based connected components labeling with decision trees. IEEE Transactions on Image Processing 19 (6), 1596–1609. doi: http://dx.doi.org/10.1109/TIP.2010.2044963

Heckbert, P., 1990. A seed fill algorithm. In: Graphics gems. Academic Press Professional, Inc., pp. 275–277.

Jana, R., Reshmidevi, T., Arun, P., Eldho, T., 2007. An enhanced technique in construction of the discrete drainage network from low-resolution spatial database. Computers and geosciences 33 (6), 717–727. doi: http://dx.doi.org/10.1016/j.cageo.2006.06.002

Jones, R., 2002. Algorithms for using a dem for mapping catchment areas of stream sediment samples. Computers and geosciences 28 (9), 1051–1060. doi: http://dx.doi.org/10.1016/S0098-3004(02)00022-5

Kite, G., 2001. Modelling the mekong: hydrological simulation for environmental impact studies. Journal of Hydrology 253 (1-4), 1–13. doi: http://dx.doi.org/10.1016/S0022-1694(01)00396-1

Liang, C., MaCkay, D., 2000. A general model of watershed extraction and representation using globally optimal flow paths and up-slope contributing areas. International Journal of Geographical Information Science 14 (4), 337–358. doi: http://dx.doi.org/10.1080/13658810050024278

Lin, W., Chou, W., Lin, C., Huang, P., Tsai, J., 2006. Automated suitable drainage network extraction from digital elevation models in taiwan's upstream watersheds. Hydrological processes 20 (2), 289–306. doi: http://dx.doi.org/10.1002/hyp.5911

Mackay, D., Band, L., 1998. Extraction and representation of nested catchment areas from digital elevation models in lake-dominated topography. Water resources research 34 (4), 897–901. doi: http://dx.doi.org/10.1029/98WR00094

Miller, D., 2003. Programs for dem analysis. Landscape dynamics and forest management. Gen. Tech. Rep. RMRSGTR-101CD. USDA Forest Service, Rocky Mountain Research Station, Fort Collins, Colorado, USA.[CD-ROM].

Nardi, F., Grimaldi, S., Santini, M., Petroselli, A., Ubertini, L., 2008. Hydrogeomorphic properties of simulated drainage patterns using digital elevation models: the flat area issue/propriétés hydro-géomorphologiques de réseaux de drainage simulés à partir de modèles numériques de terrain: la question des zones planes. Hydrological Sciences Journal 53 (6), 1176–1193. doi: http://dx.doi.org/10.1623/hysj.53.6.1176

Phillips, J., Slattery, M., 2007. Downstream trends in discharge, slope, and stream power in a lower coastal plain river. Journal of Hydrology 334 (1-2), 290–303. doi: http://dx.doi.org/10.1016/j.jhydrol.2006.10.018

Planchon, O., Darboux, F., 2002. A fast, simple and versatile algorithm to fill the depressions of digital elevation models. Catena 46 (2-3), 159–176. doi: http://dx.doi.org/10.1016/S0341-8162(01)00164-3

Soille, P., Vogt, J., Colombo, R., 2003. Carving and adaptive drainage enforcement of grid digital elevation models. Water Resources Research 39 (12), 1366–1375. doi: http://dx.doi.org/10.1029/2002WR001879

Stepinski, T., Vilalta, R., 2005. Digital topography models for martian surfaces. Geoscience and Remote Sensing Letters, IEEE 2 (3), 260–264. doi: http://dx.doi.org/10.1109/LGRS.2005.848509

Tarboton, D., 1997. A new method for the determination of flow directions and upslope areas in grid digital elevation models. Water resources research 33, 309–319. doi: http://dx.doi.org/10.1029/96WR03137

Tarboton, D., Ames, D., 2001. Advances in the mapping of flow networks from digital elevation data. In: World Water and Environmental Resources Congress. Am. Soc Civil Engrs USA, pp. 20–24. doi: http://dx.doi.org/10.1061/40569(2001)166

Toma, L., Wickremesinghe, R., Arge, L., Chase, J. S., Vitter, J. S., Halpin, P. N., Urban, D., 2001. Flow computation on massive grids. In: Proceedings of the 9th ACM international symposium on Advances in geographic information systems. GIS '01. ACM, New York, NY, USA, pp. 82–87. doi: http://dx.doi.org/10.1145/512161.512180

Tribe, A., 1992. Automated recognition of valley lines and drainage networks from grid digital elevation models: a review and a new method. Journal of Hydrology 139 (1-4), 263–293. doi: http://dx.doi.org/10.1016/0022-1694(92)90206-B

Turcotte, R., Fortin, J., Rousseau, A., Massicotte, S., Villeneuve, J., 2001. Determination of the drainage structure of a watershed using a digital elevation model and a digital river and lake network. Journal of Hydrology 240 (3-4), 225–242. doi: http://dx.doi.org/10.1016/S0022-1694(00)00342-5

Vogt, J., Colombo, R., Bertolo, F., 2003. Deriving drainage networks and catchment boundaries: a new methodology combining digital elevation data and environmental characteristics. Geomorphology 53 (3-4), 281–298. doi: http://dx.doi.org/10.1016/S0169-555X(02)00319-7

Wallis, C., Watson, D., Tarboton, D., Wallace, R., 2009. Parallel flow-direction and contributing area calculation for hydrology analysis in digital elevation models. In: International Conference on Parallel and Distributed Processing Techniques and Applications.

Wang, L., Liu, H., 2006. An efficient method for identifying and filling surface depressions in digital elevation models for hydrologic analysis and modelling. International Journal of Geographical Information Science 20 (2), 193–213. doi: http://dx.doi.org/10.1080/13658810500433453

White, A., Kumar, P., Saco, P., Rhoads, B., Yen, B., 2004. Hydrodynamic and geomorphologic dispersion: scale effects in the illinois river basin. Journal of hydrology 288 (3-4), 237–257. doi: http://dx.doi.org/10.1016/j.jhydrol.2003.10.019

Zhang, H., Huang, G., 2009. Building channel networks for flat regions in digital elevation models. Hydrological Processes 23 (20), 2879–2887. doi: http://dx.doi.org/10.1002/hyp.7378

Appendix A. Pseudocode

Algorithm 1 ResolveFlats: The main body of the algorithm, as described in §2.1. **Upon entry, (1)** *DEM* contains the elevations of every cell or a value NoData for cells not part of the DEM. **(2)** *FlowDirs* contains the flow direction of every cell; cells without a local gradient are marked NoFlow. Algorithm 2 provides an example of how this might be done. **At exit, (1)** *FlatMask* has a value greater than or equal to zero for each cell, indicating its number of increments. These can be used be used in conjunction with *Labels* to determine flow directions without altering the DEM, as exemplified by Algorithm 7, or to alter the DEM in subtle ways to direct flow, as exemplified by Algorithm 8. **(2)** *Labels* has values greater than or equal to 1 for each cell which is in a flat. Each flats' cells bear a label unique to that flat.

```
1: procedure ResolveFlats(DEM, FlowDirs)
Require: The dimensions of DEM must equal those of FlowDirs
2:    Let FlatMask and Labels have the same dimensions as DEM and be initialized to 0
3:    Let LowEdges and HighEdges be two empty FIFO queues
4:
5:    FlatEdges(HighEdges,LowEdges)                          ▷ Algorithm 3
6:    if LowEdges is empty then
7:       if HighEdges is not empty then
8:          There were undrainable flats
9:       else
10:         There were no flats
11:      return
12:
13:   Label ← 1
14:   for all c ∈ LowEdges do
15:      if c is not labeled then
16:         LabelFlats(c, Label)                             ▷ Algorithm 4
17:         Label ← Label + 1
18:
19:   for all c ∈ HighEdges do
20:      if c is not labeled then Remove c
21:   if Any cell was removed from HighEdges then
22:      Not all flats have outlets
23:
24:   Let FlatHeight be an array with length equal to the value of Label
25:   AwayFromHigher(HighEdges, FlatHeight)                  ▷ Algorithm 5
26:   TowardsLower(LowEdges, FlatHeight)                     ▷ Algorithm 6
27:   return FlatMask,Labels
```

Algorithm 2 D8FlowDirs: This is an example of how flow directions could be assigned to cells, as described in §2.2. **Upon entry, (1)** *DEM* contains the elevations of each cell or a value NoData for cells not part of the DEM. **At exit,** *FlowDirs* contains a flow direction for each cell which has a local gradient; those cells without a local gradient are marked NoFlow.

```
1: procedure D8FlowDirs(DEM, FlowDirs)
Require: FlowDirs has the same dimensions as DEM
2:    for all c in DEM do
3:       if DEM(c) = NoData then
4:          FlowDirs(c) ← NoData
5:          repeat loop
6:       e_min ← DEM(c)
7:       n_min ← NoFlow
8:       for all neighbors n of c do
9:          if n ∉ DEM then repeat loop
10:         if DEM(n) < e_min then
11:            e_min ← DEM(n)
12:            n_min ← direction to n
13:      FlowDirs(c) ← n_min
```

Algorithm 3 FlatEdges: This function locates flat cells which border on higher and lower terrain and places them into queues for further processing, as described in §2.2. **Upon entry, (1)** *DEM* contains the elevations of every cell or a value NoData for cells not part of the DEM. **(2)** Any cell without a local gradient is marked NoFlow in *FlowDirs*. **At exit, (1)** *HighEdges* contains all the high edge cells (those flat cells adjacent to higher terrain) of the DEM, in no particular order. **(2)** *LowEdges* contains all the low edge cells of the DEM, in no particular order.

```
1: procedure FlatEdges(HighEdges, LowEdges)
Require: DEM,FlowDirs
2:     for all c in Flowdirs do
3:         if c = NoData then repeat loop
4:         for all neighbors n of c do
5:             if n ∉ DEM then repeat loop
6:             if n = NoData then repeat loop
7:             if c ≠ NoFlow and n = NoFlow and DEM(c) = DEM(n) then
8:                 Push c onto LowEdges
9:                 exit loop
10:            else if c = NoFlow and DEM(c) < DEM(n) then
11:                Push c onto HighEdges
12:                exit loop
```

Algorithm 4 LabelFlats: This flood-fill function gives all the cells of a flat a common label, as described by §2.2. **Upon entry, (1)** *DEM* contains the elevations of every cell or a value NoData for cells not part of the DEM. **(2)** *Labels* has the same dimensions as *DEM*. **(3)** c belongs to the flat which is to be labeled. **(4)** L is a unique label which has not been previously applied to a flat. **(5)** *Labels* has been initialized to zero prior to the first call to this function. **(6)** *Labels* has values greater than or equal to 1 for each processed cell which is in a flat. Each flats' cells bear a label unique to that flat. **At exit, (1)** c and every cell reachable from c by passing over only cells of the same elevation as c (all the cells in the flat to which c belongs) is marked as L in *Labels*. **(2)** *Labels* has been updated to reflect the new labels which have been applied.

```
1: procedure LabelFlats(cell c, label L)
Require: DEM, Labels
2:     Let ToFill be an empty FIFO queue
3:     Push c onto ToFill
4:     E ← DEM(c)
5:     while ToFill is not empty do
6:         c ← pop(ToFill)
7:         if c ∉ DEM then repeat loop
8:         if DEM(c) ≠ E then repeat loop                ▷ c is not a part of the flat
9:         if Labels(c) ≠ 0 then repeat loop             ▷ c is already labeled
10:        Labels(c) ← L
11:        Push all neighbors of c onto ToFill
```

Algorithm 5 AwayFromHigher: This procedure builds a gradient away from higher terrain, as described in §2.3 and Fig. 1. **Upon entry, (1)** Every cell in *Labels* is marked either 0, indicating that the cell is not part of a flat, or a number greater than zero which identifies the flat to which the cell belongs. **(2)** Any cell without a local gradient is marked NoFlow in *FlowDirs*. **(3)** Every cell in *FlatMask* is initialized to 0. **(4)** *HighEdges* contains, in no particular order, all the high edge cells of the DEM which are part of drainable flats. **At exit, (1)** *FlatHeight* has an entry for each label value of *Labels* indicating the maximal number of increments to be applied to the flat identified by that label. **(2)** *FlatMask* contains the number of increments to be applied to each cell to form a gradient away from higher terrain; cells not in a flat have a value of 0.

```
1: procedure AwayFromHigher(HighEdges, FlatHeight)
Require: Labels, FlatMask, FlowDirs
2:     Let FlatMask have the same dimensions as Labels
3:     Let FlatHeight have length equal to Max(Labels)
4:     loops ← 1
5:     Push Marker onto HighEdges
6:     while length(HighEdges) > 1 do
7:         c ← pop(HighEdges)
8:         if c = Marker then
9:             loops ← loops + 1
10:            Push Marker onto HighEdges
11:            repeat loop
12:        if FlatMask(c) > 0 then repeat loop
13:        FlatMask(c) ← loops
14:        FlatHeight(Labels(c)) ← loops
15:        for all neighbors n of c do
16:            if n ∈ Labels and Labels(n) = Labels(c) and FlowDirs(n) = NoFlow then
17:                Push n onto HighEdges
```

Algorithm 6 TowardsLower: This procedure builds a gradient towards lower terrain and combines it with the gradient away from higher terrain, as described in §2.4 and Fig. 2. **Upon entry, (1)** Every cell in *Labels* is marked either 0, indicating that the cell is not part of a flat, or a number greater than zero which identifies the flat to which the cell belongs. **(2)** Any cell without a local gradient is marked NoFlow in *FlowDirs*. **(3)** Every cell in *FlatMask* has either a value of 0, indicating that the cell is not part of a flat, or a value greater than zero indicating the number of increments which must be added to it to form a gradient away from higher terrain. **(4)** *FlatHeight* has an entry for each label value of *Labels* indicating the maximal number of increments to be applied to the flat identified by that label in order to form the gradient away from higher terrain. **(5)** *LowEdges* contains, in no particular order, all the low edge cells of the DEM. **At exit, (1)** *FlatMask* contains the number of increments to be applied to each cell to form a superposition of the gradient away from higher terrain with the gradient towards lower terrain; cells not in a flat have a value of 0.

```
1: procedure TowardsLower(LowEdges, FlatHeight)
Require: Labels, FlatMask, FlowDirs
2:     Make all entries in FlatMask negative
3:     loops ← 1
4:     Push Marker onto LowEdges
5:     while length(LowEdges) > 1 do
6:         c ← pop(LowEdges)
7:         if c = Marker then
8:             loops ← loops + 1
9:             Push Marker onto LowEdges
10:            repeat loop
11:        if FlatMask(c) > 0 then repeat loop
12:        if FlatMask(c) < 0 then
13:            FlatMask(c) ← FlatHeight(Labels(c)) + FlatMask(c) + 2 · loops
14:        else
15:            FlatMask(c) ← 2 · loops
16:        for all neighbors n of c do
17:            if n ∈ Labels and Labels(n) = Labels(c) and FlowDirs(n) = NoFlow then
18:                Push n onto LowEdges
```

Algorithm 7 D8MaskedFlowDirs: This is an example of how the *FlatMask* and *Labels* developed previously may be used to determine flow directions across flats, as described in §2.5. This function will need to be adjusted to fit with the flow metric being used. **Upon entry, (1)** *FlatMask* contains the number of increments to be applied to each cell to form a gradient which will drain the flat it is a part of. **(2)** Any cell without a local gradient has a value of NoFlow in *FlowDirs*; all other cells have defined flow directions. **(3)** If a cell is part of a flat, it has a value greater than zero in *Labels* indicating which flat it is a member of; otherwise, it has a value of 0. **At exit,** every cell whose flow direction could be resolved by this algorithm (all drainable flats) has a defined flow direction in *FlowDirs*. Any cells which could not be resolved (non-drainable flats) are still marked NoFlow.

```
 1: procedure D8MaskedFlowDirs(FlatMask, FlowDirs, Labels)
 2:   for all c in FlowDirs do
 3:     if FlowDirs(c) = NoData then repeat loop
 4:     if FlowDirs(c) ≠ NoFlow then repeat loop
 5:     e_min ← FlatMask(c)
 6:     n_min ← NoFlow
 7:     for all neighbors n of c do
 8:       if n ∉ FlowDirs then repeat loop
 9:       if Labels(n) ≠ Labels(c) then repeat loop
10:       if FlatMask(n) < e_min then
11:         e_min ← FlatMask(n)
12:         n_min ← direction to n
13:     FlowDirs(c) ← n_min
```

Algorithm 8 AlterDEM: This is an example of how the *FlatMask* and *Labels* developed previously may be used to increase the elevation of cells in the original DEM to enforce drainage directions, as described in §2.5. The NextAfter function is defined by the `C99` and `POSIX` standards. A safety check is used to ensure that this doesn't change the original hydrology of the DEM. **Upon entry, (1)** *FlatMask* contains the number of increments to be applied to each cell to form a gradient which will drain the flat. **(2)** Any cell without a local gradient has a value of NoFlow in *FlowDirs*; all other cells have defined flow directions. **(3)** If a cell is part of a flat, it has a value greater than zero in *Labels* indicating which flat it is a member of; otherwise, it has a value of 0. **At exit,** *DEM* has been altered so that the elevation of every cell which was part of a drainable flat is adjusted such that it is guaranteed to drain.

```
 1: procedure AlterDEM(DEM, FlatMask, Labels)
 2:   for all c in DEM do
 3:     if DEM(c) = NoData then repeat loop
 4:     if Label(c) = 0 then repeat loop                ▷ Cell is not part of a flat
 5:     if FlatMask(c) = 0 then repeat loop
 6:     for all neighbors n of c do                     ▷ Note whether c is already higher than n
 7:       Higher(n) ← (DEM(c) > DEM(n))
 8:     for j = 0 → FlatMask(c) do                      ▷ Alter the DEM
 9:       DEM(c) ←NextAfter(DEM(c), ∞)
10:     for all neighbors n of c do                     ▷ Check if alteration was significant
11:       if Labels(n) = Labels(c) then repeat loop
12:       if DEM(c) < DEM(n) then repeat loop
13:       if not Higher(n) then
14:         c was lower than n before, but it isn't now.
15:         The alteration was significant.
16:         This is a problem.
```