International Journal of Digital Earth

ISSN: 1753-8947 (Print) 1753-8955 (Online) Journal homepage: http://www.tandfonline.com/loi/tjde20

Efficient Priority-Flood depression filling in raster digital elevation models

Hongqiang Wei, Guiyun Zhou & Suhua Fu

To cite this article: Hongqiang Wei, Guiyun Zhou & Suhua Fu (2018): Efficient Priority-Flood depression filling in raster digital elevation models, International Journal of Digital Earth, DOI: 10.1080/17538947.2018.1429503

To link to this article: https://doi.org/10.1080/17538947.2018.1429503

Published online: 30 Jan 2018.

Submit your article to this journal

View related articles

View Crossmark data

Full Terms & Conditions of access and use can be found at
http://www.tandfonline.com/action/journalInformation?journalCode=tjde20

INTERNATIONAL JOURNAL OF DIGITAL EARTH, 2018
https://doi.org/10.1080/17538947.2018.1429503

Efficient Priority-Flood depression filling in raster digital elevation models

Hongqiang Wei[a,b], Guiyun Zhou[a,b,c] and Suhua Fu[c,d]

[a]Center for Information Geoscience, University of Electronic Science and Technology of China, Chengdu, People's Republic of China; [b]School of Resources and Environment, University of Electronic Science and Technology of China, Chengdu, People's Republic of China; [c]State Key Laboratory of Soil Erosion and Dryland Farming on the Loess Plateau, Institute of Soil and Water Conservation, Chinese Academy of Sciences, Yangling, People's Republic of China; [d]Faculty of Geographical Science, Beijing Normal University, Beijing, People's Republic of China

ABSTRACT
Depressions in raster digital elevation models (DEM) present a challenge for extracting hydrological networks. They are commonly filled before subsequent algorithms are further applied. Among existing algorithms for filling depressions, the Priority-Flood algorithm runs the fastest. In this study, we propose an improved variant over the fastest existing sequential variant of the Priority-Flood algorithm for filling depressions in floating-point DEMs. The proposed variant introduces a series of improvements and greatly reduces the number of cells that need to be processed by the priority queue (PQ), the key data structure used in the algorithm. The proposed variant is evaluated based on statistics from 30 experiments. On average, our proposed variant reduces the number of cells processed by the PQ by around 70%. The speed-up ratios of our proposed variant over the existing fastest variant of the Priority-Flood algorithm range from 31% to 52%, with an average of 45%. The proposed variant can be used to fill depressions in large DEMs in much less time and in the parallel implementation of the Priority-Flood algorithm to further reduce the running time for processing huge DEMs that cannot be dealt with easily on single computers.

ARTICLE HISTORY
Received 28 August 2017
Accepted 11 January 2018

KEYWORDS
Depression filling; DEM; drainage network

1. Introduction

Raster digital elevation models (DEMs) are widely used by practitioners to automatically derive drainage networks in many scenarios such as hydrological process modeling (Nobre et al. 2011) and soil erosion simulation (Fu et al. 2011). Natural and spurious depressions are commonly found in a DEM (Lindsay and Creed 2006). A depression in a DEM is a group of interconnected cells that are surrounded by higher cells. Cells in a depression present a challenge for the extraction of drainage networks because they disrupt the flow of water toward the border of the DEM. Therefore, depressions are typically removed before the DEM is used to extract drainage networks.

Depressions can be removed by a filling process (Jenson and Domingue 1988; Wang and Liu 2006; Barnes, Lehman, and Mulla 2014; Bai et al. 2015) or a breaching process (Soille 2004) or a combination of them (Lindsay 2016). The breaching-based method has been claimed to be more accurate and the derived flow path patterns are less impacted than by depression-filling method (Soille 2004;

CONTACT Guiyun Zhou zhouguiyun@uestc.edu.cn Center for Information Geoscience, University of Electronic Science and Technology of China, Chengdu 611731, People's Republic of China; School of Resources and Environment, University of Electronic Science and Technology of China, Chengdu 611731, People's Republic of China; State Key Laboratory of Soil Erosion and Dryland Farming on the Loess Plateau, Institute of Soil and Water Conservation, Chinese Academy of Sciences, Yangling 712100, People's Republic of China

© 2018 Informa UK Limited, trading as Taylor & Francis Group

Lindsay and Creed 2006). However, depression filling is used much more frequently than breaching methods among practitioners (Lindsay 2016). Compared with breaching methods, depression filling has a long history of development, produces the same filled DEM regardless of the methods used, and is made widely available in both commercial and open-source software packages, such as ArcGIS, GRASS, TauDEM, RichDEM, etc. Recent developments in depression filling have also substantially improved the algorithm in term of both execution time and the size of the DEM it is able to process (Barnes 2016).

According to Zhou et al. (2017), three types of algorithms have emerged for depression filling. The first type of the algorithm has a time complexity of $O(N^2)$ (Jenson and Domingue 1988; Arge et al. 2003; Zhu, Tian, and Zhao 2006; Gong and Xie 2009). The second type of the algorithm has a time complexity of $O(N^{1.2})$ (Planchon and Darboux 2002; Wallis et al. 2009; Qin and Zhan 2012; Rueda, Noguera, and Martínez-Cruz 2013). The third type of the algorithm is the Priority-Flood algorithm, a concept first proposed in Barnes, Lehman, and Mulla (2014). It has a time complexity of $O(N \log N)$ for floating-point DEMs and $O(N)$ for integer DEMs. The Priority-Flood algorithm outperforms the algorithms of the first two types in terms of both time complexity and memory requirement (Wang and Liu 2006; Zhou, Sun, and Fu 2016).

The Priority-Flood algorithm has been implemented in parallel for it to be able to process large DEMs. Barnes (2016) proposes a very efficient parallel implementation of the Priority-Flood algorithm with almost linear scaling. Zhou et al. (2017) provide both OpenMP- and MPI-based parallel implementations of the Priority-Flood algorithm. Both the parallel algorithms are based on the one-pass implementation of the variant of the Priority-Flood algorithm proposed in Zhou, Sun, and Fu (2016) for processing floating-point DEMs.

While the sequential Priority-Flood algorithm has greatly reduced the running time and memory requirements for depression filling, there is a need to further improve the algorithm for processing large DEMs and for it to be used as an integral part of the parallel implementations. This study builds on the sequential one-pass Zhou variant and proposes an improved variant, which further reduces the running time by 45% on average. The remainder of this paper is organized as follows. Section 2 reviews the one-pass Zhou variant. Our proposed improvement is proposed in Section 3. Section 4 presents the experimental results of the proposed improvement. We conclude the paper in Section 5.

2. Review of the one-pass Zhou variant of the Priority-Flood algorithm

This section reviews the one-pass implementation of the Zhou variant of the Priority-Flood algorithm, henceforth referred to only as the Zhou variant. The Zhou variant of the Priority-Flood algorithm builds on two previous variants, proposed by Wang and Liu (2006) and Barnes, Lehman, and Mulla (2014), respectively. The variant of Wang and Liu marks the emergence of the third type of the algorithm for filling depressions in floating-point DEMs. The variant of Wan and Liu is a best-first search algorithm (Asai and Fukunaga 2017) that minimizes the cost of surface water to spill out to the boundary of the DEM. The variant of Barnes, Lehman, and Mulla (2014) focuses on reducing the processing time for cells located in depressions and makes a distinction between the processing of depression cells and non-depression cells, helping to reveal the underlying mechanism of the Priority-Flood algorithm.

The depression-filling process of the Zhou variant is shown in Figure 1. Like the other two variants, the Zhou variant first pushes all border cells of a DEM into a priority queue (PQ), which is used to find the cell with the lowest spill elevation. The spill elevation of a cell is defined as the lowest elevation for the cell to be lifted to in order to have a spill path. The spill path of a cell is an ordered sequence of cells that ends with a border cell of the DEM. The elevation of a cell in a spill path is lower than or equal to the elevation of the preceding cell. In Figure 1(b), border cells A and H are pushed into PQ. In the Zhou variant, a cell is categorized as either a depression cell or a slope cell. As the name suggests, depression cells are located in depressions and need to be raised for them to have spill paths. In contrast, slope cells are located in slopes and do not need to be raised.

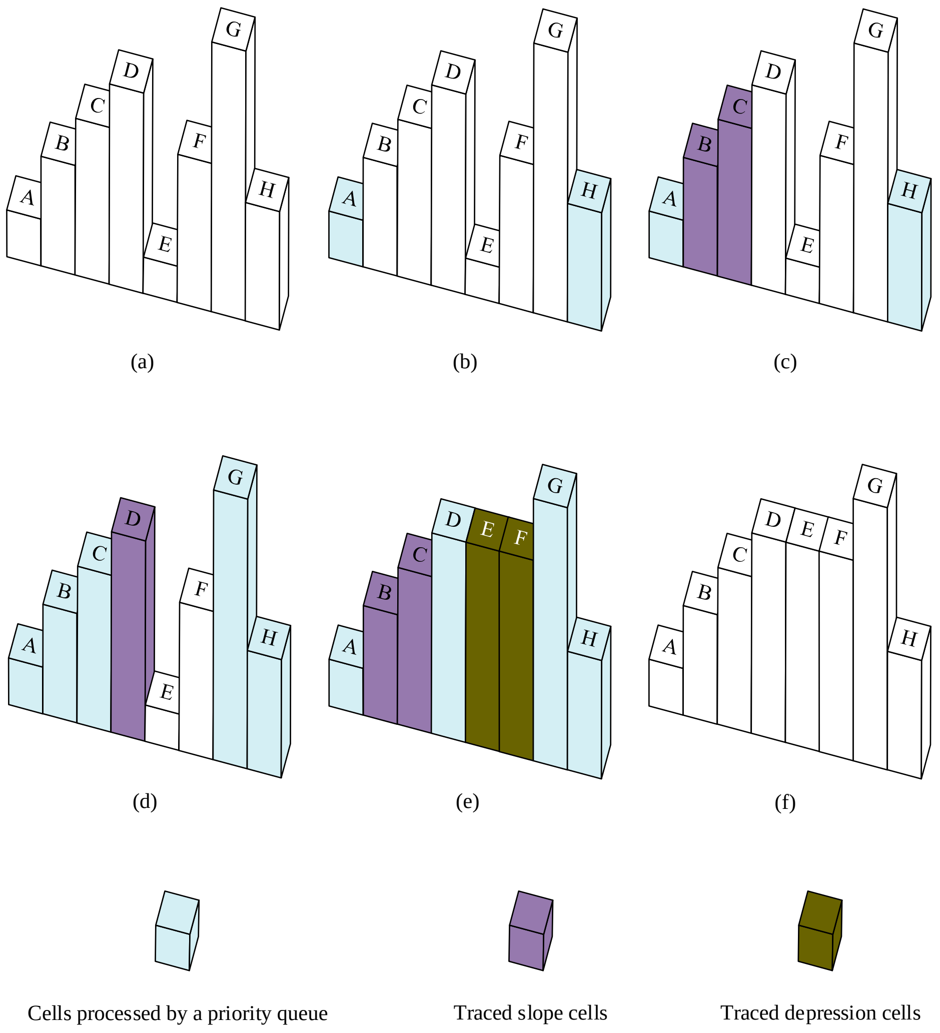

Figure 1. A cross-section of the DEM illustrating the Zhou variant of the Priority-Flood algorithm for filling depressions. (a) Original DEM; (b) border cells *A* and *H* are pushed into a priority queue (PQ); (c) *A* is popped off PQ and slope cells *B*, *C*, and *D* are processed by a region-growing process. Potential spill cell *D* is pushed into PQ; (d) *H* is popped off PQ and slope cell *G* is processed. *G* is also a potential spill cell and pushed into PQ. (e) *D* is popped off PQ and depression cells *E* and *F* are processed by a region-growing process. (f) The filled DEM.

The Zhou variant uses region-growing processes to process both types of cells. In Figure 1(c), *A* has the lowest spill elevation and is popped off PQ. Slope cells *B*, *C*, and *D* are processed using a region-growing process with *A* as the seed cell. Cell *D* has an unprocessed neighbor *E* in another unprocessed depression. In Zhou variant, *D* is called a potential spill cell and is pushed into PQ. In Figure 1(d), *H* has the lowest spill elevation in PQ and is popped off PQ. Another region-growing process is conducted to trace slope cell *G* and *G* is also a potential spill cell and pushed into PQ during the tracing. In Figure 1(e), *D* has the lowest spill elevation in PQ and is popped off PQ. A region-growing process is applied to trace depression cells *E* and *F*. The filled DEM is shown in Figure 1(f). For more details of the Zhou variant, the readers are referred to Zhou, Sun, and Fu (2016).

The primary improvement of the Zhou variant over the variant of Barnes, Lehman, and Mulla (2014) is to process the majority of slope cells using a region-growing process instead of a PQ. Plain (FIFO) queues are used in region-growing processes. Potential spill cells and some interior slope cells are processed using a PQ. A plain queue has an amortized time complexity of $O(1)$ per operation, and a generic floating-point PQ has an amortized time complexity of $O(\log N)$ per operation. Because the time complexity of a generic PQ is greater than that of a plain queue and operations on a PQ involve more memory movement than those on a plain queue, the Zhou variant substantially reduces the running time. It is noteworthy that different implementations of the PQ may result in different running times (Luengo Hendriks 2010; Barnes, Lehman, and Mulla 2014). One of the key parts of the region-growing process for tracing slope cells is to determine whether the currently traced slope cell c (henceforth referred to as the focal cell) should be pushed into PQ or not. In order to make the decision, each unprocessed lower neighbor of c is visited in turn. Suppose that Neighbors (c) is the collection of neighbors of c; SpillEl (c) is the spill elevation of c; and DEM(c) is the original elevation of c. If an unprocessed lower neighbor n of c satisfies the following statement,

$$\exists j \in \mathrm{Neighbors}(n)\colon j \text{ is processed and } \mathrm{SpillEl}(j) < \mathrm{DEM}(n), \tag{1}$$

n can be traced from j as a slope cell and is treated as a processed neighbor. If the Statement (1) is true for all of the unprocessed lower neighbors of c, c does not need to be processed by PQ. If any of the unprocessed lower neighbors cannot be treated as a processed neighbor, c needs to be pushed into PQ so that that neighbor can be raised correctly if it is a depression cell.

The flowchart for tracing slope cells and identifying potential spill cells in the Zhou variant is shown in Figure 2.

3. Proposed variant of the Priority-Flood algorithm

3.1. Proposed variant

Our experiments show that the number of cells processed by the PQ in the Zhou variant and the number of depressions typically differ by one order of magnitude. Considering that a depression typically has only one spill outlet, there is a potential for the number of cells processed by the PQ to be further decreased. Like the Zhou variant, our proposed variant also focuses on floating-point DEMs. On average, our proposed variant reduces the number of cells processed by the PQ by around 70%, which helps to further reduce the running times of the Priority-Flood algorithm.

Our proposed improvements mainly aim to reduce the number of cells processed by the PQ by successively relaxing requirements on unprocessed lower neighbors of the focal cell c. Instead of Statement (1), our first improvement checks whether n satisfies the following statement:

$$\exists j \in \mathrm{Neighbors}(n)\colon j \text{ is processed and } \mathrm{SpillEl}(j) < \mathrm{DEM}(c). \tag{2}$$

Considering that n is lower than c, this statement is more likely to be true than Statement (1), and thus c is less likely to be pushed into PQ. This relaxation of requirement can still ensure that n is correctly raised if it is a depression cell. If j is lower than n, n has a spill path that passes through j and thus is a slope cell. In this case, cell n does not need to be raised. If n is a depression cell and j is higher than n but still lower than c (Figure 3), c is not the spill outlet of the depression where n is located because the spill outlet of a depression is the lowest cell that borders the depression. In this case, both c and j border the depression but j is lower than c, so c is not the spill outlet and does not need to be processed by PQ in order to raise the depression where n is located. Therefore, in all cases, when Statement (2) is true, n has a spill path that is lower than c, which means that the spill elevation of each cell on the spill path, including cell n, is less than that of c.

Our second improvement requires a Boolean matrix S of the size of 5 by 5, with the center element corresponding to c (Figure 3). Because the S matrix covers not only the immediate neighbors of c but also the immediate neighbors of the immediate neighbors of c, using a 5-by-5 matrix instead of a 3-by-3 matrix as the S matrix makes it unnecessary to check whether the array indices are out of bound

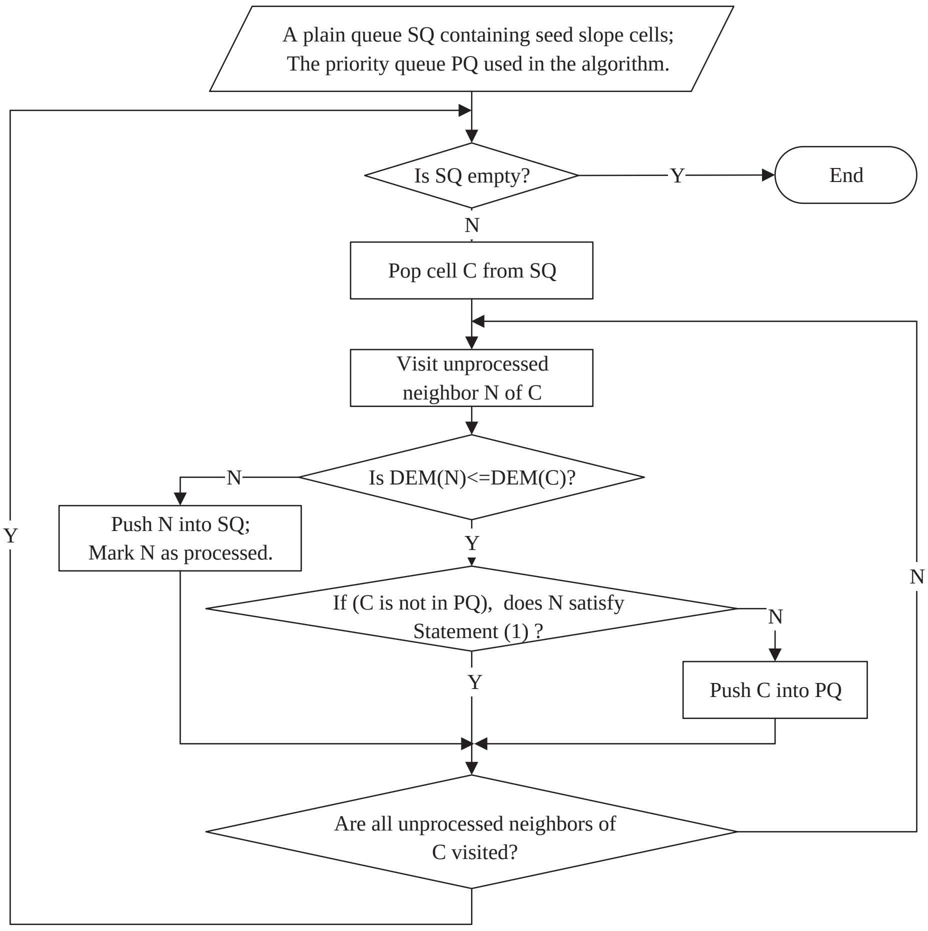
A plain queue SQ containing seed slope cells;
The priority queue PQ used in the algorithm.
Is SQ empty?
Y
End
N
Pop cell C from SQ
Visit unprocessed neighbor N of C
Is DEM(N)<=DEM(C)?
N
Push N into SQ; Mark N as processed.
Y
If (C is not in PQ), does N satisfy Statement (1) ?
N
Push C into PQ
Y
Are all unprocessed neighbors of C visited?
N
Y

Figure 2. The flowchart of the Zhou variant for tracing slope cells and identifying potential spill cells.

or not in the following processing and further helps to reduce running time. $S(n)$ indicates whether a lower unprocessed neighbor n *of* c has a spill path that is lower than c. If n is not the immediate neighbor, $S(n)$ is always false. The matrix S is initialized with false values and updated when the unprocessed lower neighbors of cell c are visited in turn. Our second improvement is to check whether an unprocessed lower neighbor n of c satisfies the following statement:

$$\exists j \in \text{Neighbors}(n): S(j) \text{ is true or}(\, j \text{ is processed and SpillEl}(j) < \text{DEM}(c)). \tag{3}$$

In addition to checking whether n has a lower spill path than c as Statement (2) does, Statement (3) also checks whether the neighbors of n have spill paths that are lower than c. When Statement (3) is true, if n has a lower spill path than c, $S(n)$ is set to be true; if the neighbors of n have lower spill paths than c, since n is also lower than c, n also have a lower spill path than c and c does not need to be processed by PQ and $S(n)$ is set to be true. Because only visited neighbors of c may have true values in S matrix, the requirement that '$s(j)$ is true' basically looks at only the cells that neighbor both c and n. Statement (3) further relaxes requirement on n and is more likely to be true than Statement (2), and thus c is less likely to be pushed into PQ. This further relaxation can still ensure that the unprocessed lower neighbors of c are raised correctly if they are depression cells. In Figure 3, suppose n is the first neighbor of c and the neighbors are visited in the clockwise order. When n is visited, $S(n)$ is false but n satisfies Statement (2), so $S(n)$ is set to be true and n has a lower spill path than c. m does

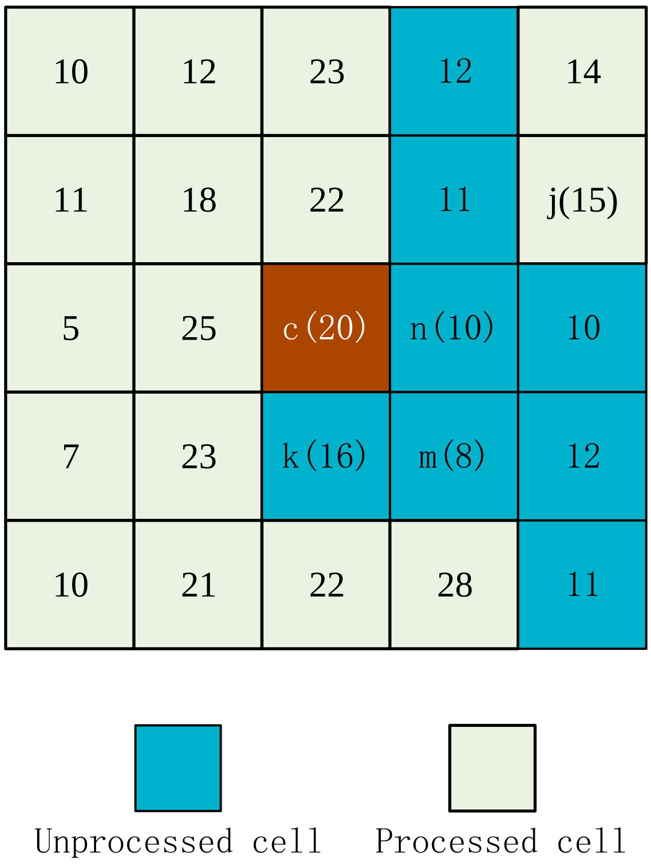

Figure 3. An example layout of the local neighborhood of a slope cell. The numbers are elevation values. Cell *c* is the focal cell. Cell *n*, *m*, and *k* are unprocessed and lower than cell *c*. Cell *j* is a processed neighbor of *n*.

not satisfy Statement (2). Because n is the neighbor of m and $S(n)$ is true, m satisfies Statement (3). $S(m)$ is set to be true, and m has a lower spill path than c. If m is a depression cell, m can be raised to the elevation of j so that it can have a spill path ($m{-}n{-}j$). Therefore, c cannot be the spill outlet of the depression where m is located because c is higher than j. Similarly, because $S(m)$ is true, k also satisfies Statement (3) and $S(k)$ is set to be true. In this case, because k can find a spill path ($k{-}m{-}n{-}j$) without being raised, k is a slope cell.

Our third improvement uses a temporary plain queue to store all the cells that need to be processed by PQ when a slope trace queue is processed. When a slope is traced, a slope trace queue is used in the region-growing processes. The trace queue may contain more than one seed cell before it is used for region growing. This typically happens after a depression is filled and those cells that border the depression are altogether pushed into the trace queue. From one seed cell, a slope can be traced and some cells on the border of the slope are considered to be potential spill cells. From another seed cell, another slope can be traced. If the two slopes are adjacent to each other, most of the potential spill cells do not need to be processed by PQ because their neighbors are all processed. This is illustrated in Figure 4. In Figure 4(a), some cells within the slope (S and T) or on the border of two slopes (K) are determined to be potential spill cells. Figure 4(b) is an example illustrating this case. Initially, the bottom left two cells with elevations of 1 and 10 are seed cells in the trace queue. Two slopes are traced from the two seed cells. When the first slope (right slope) is traced, one potential spill cell (with elevation of 12) is determined to need to be processed by PQ after checking Statement (3). However, the neighbors of the cell are all slope cells and it does not need to be

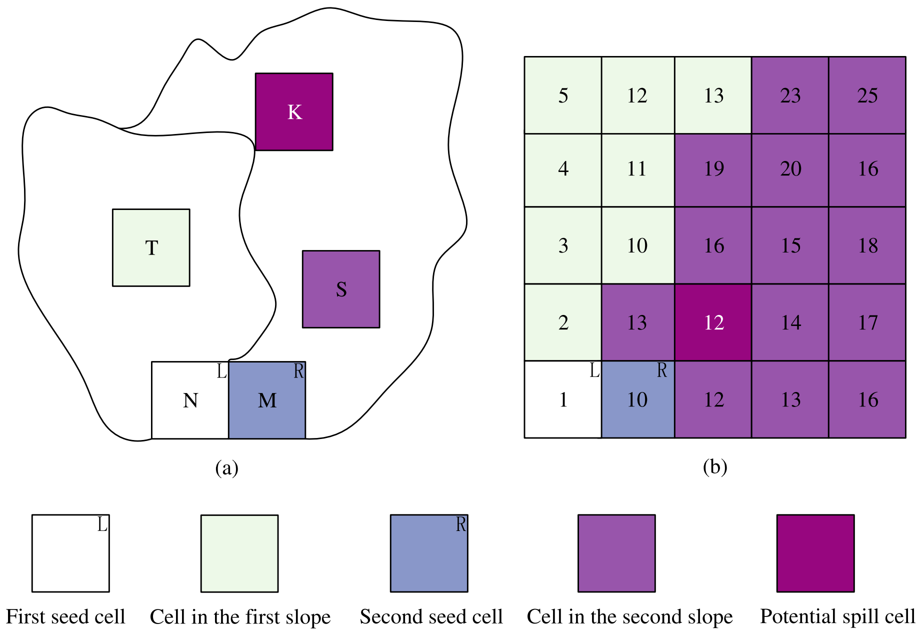

Figure 4. Spatial distribution of potential spill cells. (a) A schematic distribution of potential spill cells in the Zhou variant. (b) An example DEM illustrating the distribution in potential spill cells to be processed by PQ after checking Statement (3).

processed by PQ. Only after the two slopes have been traced can we determine that the cells do not need to be processed by PQ. Therefore, our third improvement uses a temporary plain queue to store all the cells that may need to be processed by PQ. After all the seed cells are used for tracing, the cells in the temporary plain queue are checked one by one to see whether the neighbors of these cells are all processed cells. If not, they are pushed into PQ. Otherwise, they are not pushed into PQ. This improvement further reduces the number of cells processed by PQ, which incurs a price for managing a plain queue that may contains a large number of cells. To keep a trade-off between the costs for managing a plain queue and reducing the number of cells to be processed by PQ, we introduce an index threshold. If an unprocessed lower neighbor n of cell c with a neighbor index of i (i ranges from 0 to 7) fails to satisfy Statement (3), and i is less than the index threshold, c is pushed into the temporary queue for further check. Otherwise, c is immediately pushed into PQ. We find out that an index threshold value of 2 works best for most of our test DEMs.

The flowchart of our proposed variant for tracing slope cells and identifying cells to be processed by PQ is shown in Figure 5. Note that the Zhou variant (Figure 2) keeps visiting the unprocessed neighbors of c after c is pushed into PQ. In contrast, we stop visiting the unprocessed neighbors of c once c is pushed into PQ or the temporary plain queue (PSQ in Figure 5). Since the neighbors of c will be visited after c is popped off PQ later on, stop visiting the unprocessed neighbors of c will still be able to fill all depressions correctly while reducing the unnecessary re-visiting of the unprocessed neighbors of c.

Our last improvement aims to reduce the time for initially populating the PQ with border cells of a DEM. The source code of the Zhou variant shows that it pushes border cells into PQ by looping over each cell. If a cell is not a NODATA cell, it visits each neighbor of the cell. If there exists one neighbor that is a NODATA cell, the cell is a border cell and pushed into PQ. For a typical DEM, the majority of cells are not NODATA cells and all of the eight neighbors of those cells are checked before they are found out to be not border cells. Our last improvement checks the cell in the opposite way while looping over each cell of a DEM. If a cell is a NODATA cell and there exists one neighbor

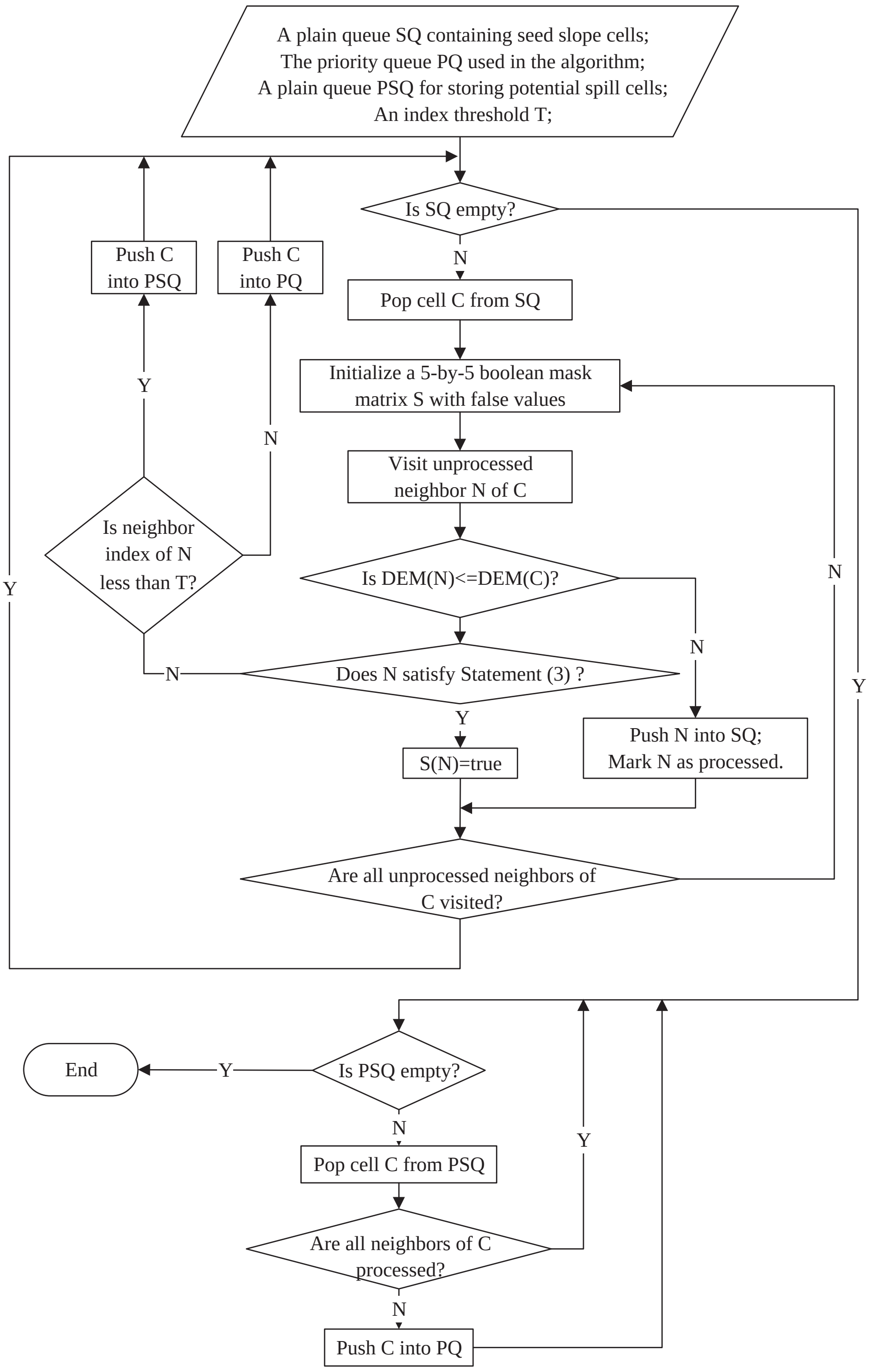
A plain queue SQ containing seed slope cells;
The priority queue PQ used in the algorithm;
A plain queue PSQ for storing potential spill cells;
An index threshold T;
Is SQ empty?
N
Pop cell C from SQ
Initialize a 5-by-5 boolean mask matrix S with false values
Visit unprocessed neighbor N of C
Is DEM(N)<=DEM(C)?
N
Does N satisfy Statement (3) ?
Y
S(N)=true
Push N into SQ; Mark N as processed.
Are all unprocessed neighbors of C visited?
N
Y
Is neighbor index of N less than T?
Y
N
N
Push C into PSQ
Push C into PQ
Y
Is PSQ empty?
Y
End
N
Pop cell C from PSQ
Are all neighbors of C processed?
Y
N
Push C into PQ

Figure 5. The flowchart of our proposed variant for tracing slope cells and identifying cells to be processed by a PQ.

of the cell that is not a NODATA cell, the neighbor cell is a border cell and pushed into PQ. Any cell that is not NODATA cell on the rectangle border of the DEM is a border cell and pushed into PQ. As pointed out in the above, because the majority of cells in a typical DEM are not NODATA cells, this reduces running time for populating the PQ with border cells. The details of this improvement can be found in the source code.

3.2. A worked example

A worked example of our proposed variant is presented in Figure 6. A plain queue SQ is used for tracing slopes. The neighbors are visited in clockwise order and the neighbor to the right is visited first. Initially, all border cells are pushed into PQ. D7 is popped off PQ and D6 is found to be a seed cell for slope tracing (Figure 6(b)). Slope cell D5 is pushed into SQ. D6 is found to be potential spill cell because its neighbor C5 fails to satisfy Statement (3). The visiting of the unprocessed neighbors of D6 is stopped (Figure 6(c)). In Figure 6(d), D4 is pushed into SQ. D5 is pushed into PQ because C4 does not satisfy Statement (3). In Figure 6(e), all unprocessed lower neighbors of D4 satisfy Statement (3) and it is not pushed into PQ. The next unprocessed neighbor of D7 is processed. All of its unprocessed lower neighbors satisfy Statement (3) and it is not pushed into PQ. SQ is now

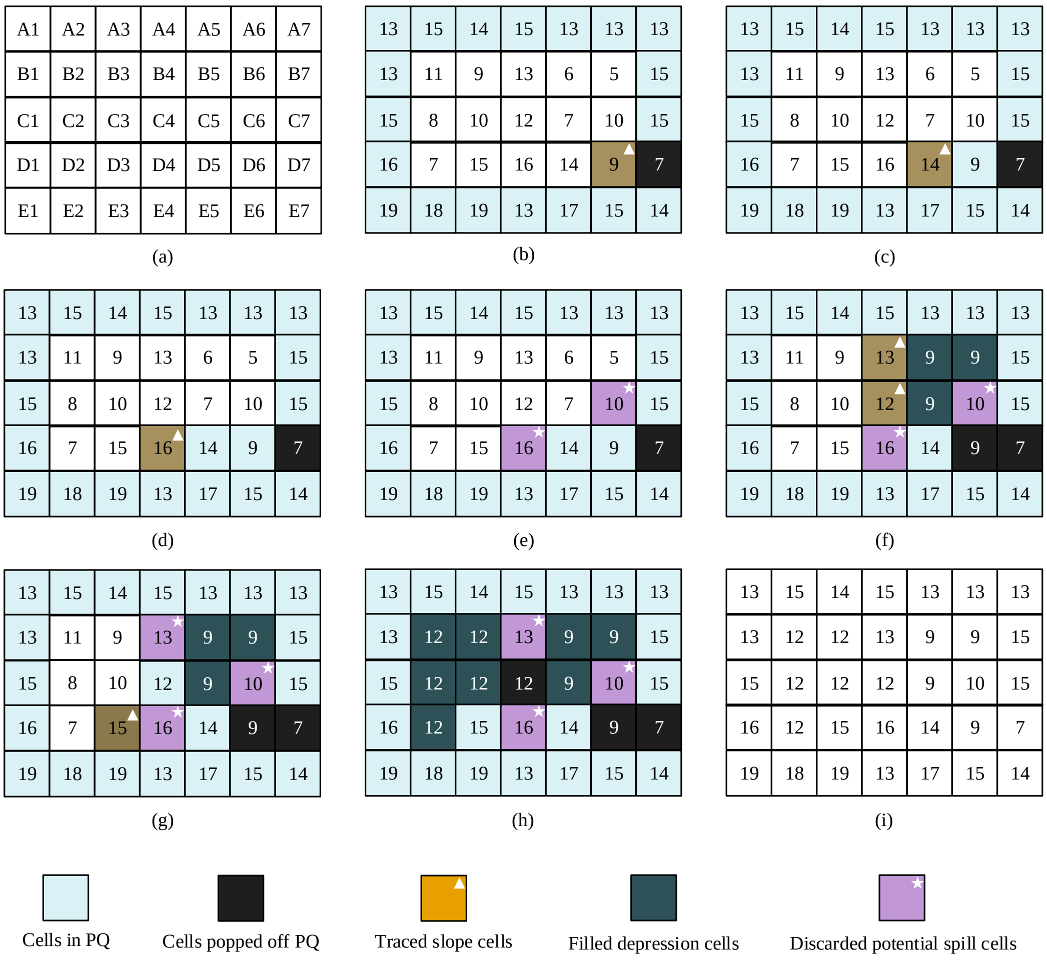

Figure 6. A worked example of our proposed variant. (a) A sample DEM with labeled cells; (b) a seed slope cell is identified; (c) slope cell D5 is traced and D6 is pushed into PQ; (d) slope cell D4 is traced and D5 is pushed into PQ; (e) D4 is not pushed into PQ; a seed slope cell C6 is identified and is not pushed into PQ. (f) A depression is filled; C4 and B4 become seed cells. (g) C4 is pushed into PQ; B4 is not pushed into PQ. (h) D3 is pushed into PQ; C4 is popped off PQ and another depression is filled, (i) the depression-filled DEM.

empty and D6 is popped off PQ. The depression composed of C5, B5, and B6 is filled and C4 and B4 are pushed into SQ (Figure 6(f)). In Figure 6(g), D3 is traced from the seed cell of C4. C4 is pushed into PQ because its neighbor C3 does not satisfy Statement (3). B4 is not pushed into PQ because all its unprocessed lower neighbors satisfy Statement (3). D3 is pushed into PQ because its neighbor D2 does not satisfy Statement (3). In Figure 6(h), C4 is popped off PQ and the depression to the left is filled. The depression-filled DEM is shown in Figure 6(i). In this example, our proposed third improvement does not affect the number of cells processed by PQ. In comparison, the Zhou variant needs to push 7 cells into PQ, whereas our proposed variant only pushes 4 cells into PQ.

3.3. Analysis of our proposed variant

Our proposed variant introduces a series of improvements over the Zhou variant. As with the Zhou variant, our proposed variant has a time complexity of $O(N \log N)$ for floating-point data, where N is the number of cells processed by PQ. The number of cells processed by PQ in our proposed variant is less than one third of that in the Zhou variant in our experiments. The increase in memory requirement is marginal. Our proposed variant requires a 5-by-5 mask matrix. In addition, it uses a temporary plain queue to store the cells that may be pushed into PQ. The queue is empties when the cells are pushed into PQ.

4. Experimental results

In this section, we compare the performances of the Zhou variant and our proposed variant. Our proposed variant is implemented in C++. As in Barnes, Lehman, and Mulla (2014) and Zhou, Sun, and Fu (2016), we use the C++ standard template library's implementation of the PQ. Doing so helps ensure the correctness and accessibility of our implementation, in addition to simplifying the programing. The source code is made available for downloading at GitHub (https://github.com/zhouguiyun-uestc/Depression-Filling) The same test dataset as used in Zhou, Sun, and Fu (2016), the LiDAR-based DEMS of 30 counties in the state of Minnesota, USA, are used for the comparison of the two variants. The average number of cells in the DEMs is around 3.96×10^8 cells. The dimensions and NODATA percentage of all DEMs can be found in Zhou, Sun, and Fu (2016). The experiments are conducted on a 64-bit Windows 7 with an Intel Core i7-6700k 4.0 GHz processor and 16GB RAM. Both the source codes of the Zhou variant and our proposed variant are compiled with the same optimization options. The index threshold in our proposed variant is set to be 2 in all experiments. The depression-filled DEMs of each county by both variants are identical.

Figure 7 plots the running times of the Zhou variant and our proposed variant for the tested DEMs. The speed-up ratio of algorithm A over B is defined as the difference in running times of them divided by the running time of B. The speed-up ratios of our proposed variant over the Zhou variant range from 31% for the county of Cleanwater to 52% for the county of Fillmore, with an average of 45%. It should be noted that many factors, including operating systems, compilation options, CPU frequency, CPU cache sizes, and the tested DEMs, affect the speed-up ratios. For example, the source code is also compiled with GCC 4.8.3 with O3 optimization on CentOS 6.5 with an Intel Xeon E5-2403 1.80 GHz processor. The speed-up ratios are 39% and 44% for the DEMs of Cleanwater and Fillmore, respectively. On both Windows and Linux, our proposed variant reduces the running time by a large margin.

Figure 8 plots the ratios between the number of cells processed by PQ and the number of depressions by the two variants. In addition, the two-pass implementation in Zhou, Sun, and Fu (2016) pushes less cells into PQ than the Zhou variant. Therefore, we also plot the ratios by the two-pass implementation. On average, the number of cells processed by PQ per depression is 22 for the Zhou variant, 18 for the two-pass implementation, and 7 for our proposed variant. This shows that our proposed variant is very efficient in reducing the number of cells that need to be processed by PQ.

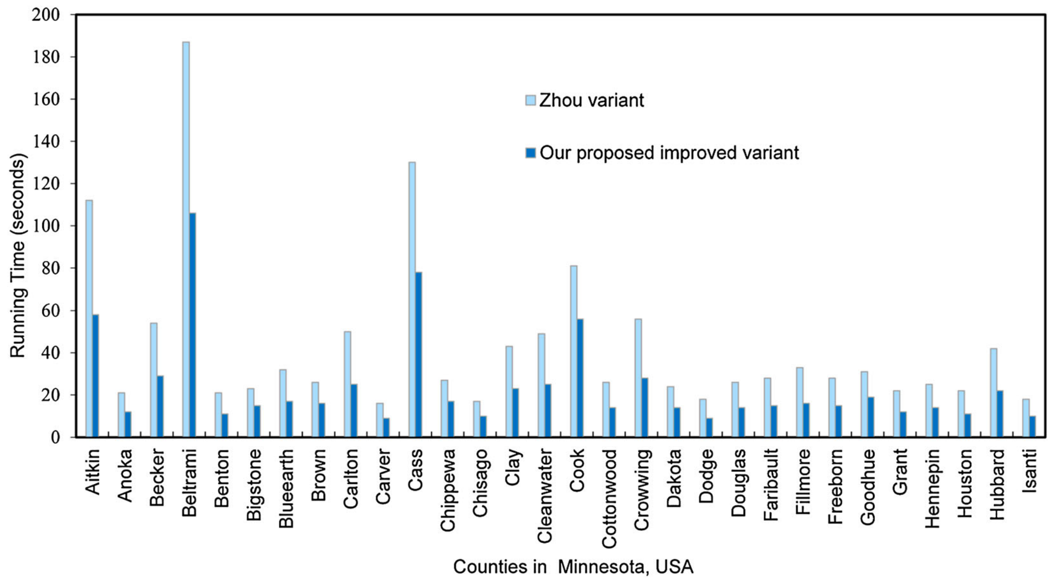

Figure 7. Running time of the Zhou variant and our proposed variant for the 3 m DEM of 30 counties in Minnesota, USA.

In addition to the experiments conducted on real DEM datasets, we also run our proposed variant on randomly generated synthetic DEMs to check the correctness of our proposed variant. In our experiments, 100,000 DEMs are generated with Perlin noise (Perlin 1985) to emulate large depressions in real DEMs. Our proposed variant and the Zhou variant generate the same depression-filled DEMs for each synthetic DEM. For completeness, the code for generating DEM with Perlin noises is also included in our GitHub repository.

5. Conclusion

The Zhou variant is the fastest existing sequential variant of the Priority-Flood algorithm for filling depressions in floating-point DEMs in the literature. In this study, we propose a new variant based

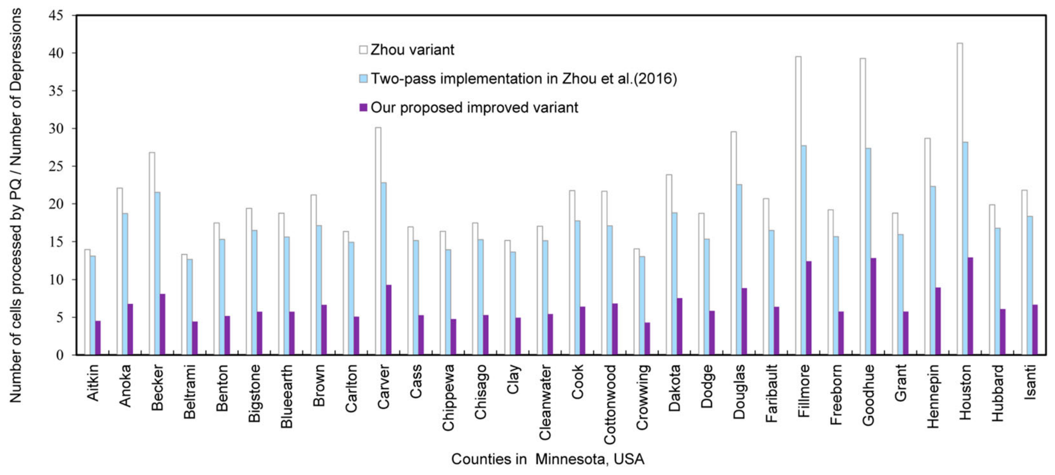

Figure 8. Running time (seconds) of the Zhou variant and our proposed variant for the 3 m DEM of 30 counties in Minnesota, USA.

on the Zhou variant. The proposed variant introduces a series of four improvements over the Zhou variant. It greatly reduces the number of cells that need to be processed by the PQ. The proposed variant is evaluated based on statistics from 30 experiments. On average, our proposed variant reduces the number of cells processed by the PQ by around 70% in comparison with the Zhou variant. The speed-up ratios of our proposed variant over the Zhou variant range from 31% to 52%, with an average of 45%. Our proposed variant can be used to fill depressions in DEMs in much less time. It can also be used as the sequential algorithm in the parallel implementation of the Priority-Flood algorithm to further reduce the running time for processing large DEMs that cannot be dealt with easily with single computers. The source code is available for download at GitHub (https://github.com/zhouguiyun-uestc/Depression-Filling)

Acknowledgements

The authors thank the anonymous referees for their constructive criticism and comments.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was supported by the National Natural Science Foundation of China [grant number 41671427]; the Open Fund of the State Key Laboratory of Soil Erosion and Dryland Farming on the Loess Plateau [grant number A314021402-1710]; the Fundamental Research Funds for the Central Universities [grant number ZYGX2016J148].

References

Arge, Lars, Jeffrey S. Chase, Patrick Halpin, Laura Toma, Jeffrey S. Vitter, Dean Urban, and Rajiv Wickremesinghe. 2003. “Efficient Flow Computation on Massive Grid Terrain Datasets.” *GeoInformatica* 7 (4): 283–313. doi:10.1023/a:1025526421410.

Asai, Masataro, and Alex Fukunaga. 2017. “Tie-breaking Strategies for Cost-optimal Best First Search.” *Journal of Artificial Intelligence Research (JAIR)* 58: 67–121. doi:10.1613/jair.5249.

Bai, Rui, Tiejian Li, Yuefei Huang, Jiaye Li, and Guangqian Wang. 2015. “An Efficient and Comprehensive Method for Drainage Network Extraction from DEM with Billions of Pixels Using a Size-balanced Binary Search Tree.” *Geomorphology* 238: 56–67. doi:10.1016/j.geomorph.2015.02.028.

Barnes, Richard. 2016. “Parallel Priority-Flood Depression Filling for Trillion Cell Digital Elevation Models on Desktops or Clusters.” *Computers & Geosciences* 96: 56–68. doi:10.1016/j.cageo.2016.07.001.

Barnes, Richard, Clarence Lehman, and David Mulla. 2014. “Priority-Flood: An Optimal Depression-filling and Watershed-labeling Algorithm for Digital Elevation Models.” *Computers & Geosciences* 62: 117–127. doi:10.1016/j.cageo.2013.04.024.

Fu, Suhua, Baoyuan Liu, Heping Liu, and Li Xu. 2011. “The Effect of Slope on Interrill Erosion at Short Slopes.” *Catena* 84 (1–2): 29–34. doi:10.1016/j.catena.2010.08.013.

Gong, Jianya, and Jibo Xie. 2009. “Extraction of Drainage Networks from Large Terrain Datasets Using High Throughput Computing.” *Computers & Geosciences* 35 (2): 337–346. doi:10.1016/j.cageo.2008.09.002.

Jenson, S. K., and J. O. Domingue. 1988. “Extracting Topographic Structure from Digital Elevation Data for Geographic Information System Analysis.” *Photogrammetric Engineering and Remote Sensing* 54 (11): 1593–1600.

Lindsay, John B. 2016. “Efficient Hybrid Breaching-filling Sink Removal Methods for Flow Path Enforcement in Digital Elevation Models.” *Hydrological Processes* 30 (6): 846–857. doi:10.1002/hyp.10648.

Lindsay, John B., and Irena F. Creed. 2006. “Distinguishing Actual and Artefact Depressions in Digital Elevation Data.” *Computers & Geosciences* 32 (8): 1192–1204. doi:10.1016/j.cageo.2005.11.002.

Luengo Hendriks, Cris L. 2010. “Revisiting Priority Queues for Image Analysis.” *Pattern Recognition* 43 (9): 3003–3012. doi:10.1016/j.patcog.2010.04.002.

Nobre, A. D., L. A. Cuartas, M. Hodnett, C. D. Rennó, G. Rodrigues, A. Silveira, M. Waterloo, and S. Saleska. 2011. “Height Above the Nearest Drainage – a Hydrologically Relevant new Terrain Model.” *Journal of Hydrology* 404 (1–2): 13–29. doi:10.1016/j.jhydrol.2011.03.051.

Perlin, Ken. 1985. "An Image Synthesizer." *ACM SIGGRAPH Computer Graphics* 19 (3): 287–296. doi:10.1145/325165.325247.

Planchon, Olivier, and Frédéric Darboux. 2002. "A Fast, Simple and Versatile Algorithm to Fill the Depressions of Digital Elevation Models." *Catena* 46 (2–3): 159–176. doi:10.1016/S0341-8162(01)00164-3.

Qin, Cheng-Zhi, and Lijun Zhan. 2012. "Parallelizing Flow-accumulation Calculations on Graphics Processing Units – from Iterative DEM Preprocessing Algorithm to Recursive Multiple-flow-direction Algorithm." *Computers & Geosciences* 43: 7–16. doi:10.1016/j.cageo.2012.02.022.

Rueda, Antonio, José M. Noguera, and Carmen Martínez-Cruz. 2013. "A Flooding Algorithm for Extracting Drainage Networks from Unprocessed Digital Elevation Models." *Computers & Geosciences* 59: 116–123. doi:10.1016/j.cageo.2013.06.001.

Soille, Pierre. 2004. "Optimal Removal of Spurious Pits in Grid Digital Elevation Models." *Water Resources Research* 40 (12): 1. doi:10.1029/2004wr003060.

Wallis, C., R. Wallace, D. G. Tarboton, D. W. Watson, K. A. T. Schreuders, and T. K. Tesfa. 2009. "Hydrologic Terrain Processing Using Parallel Computing." Paper presented at the 18th World IMACS Congress and MODSIM09 International Congress on Modelling and Simulation, Modelling and Simulation Society of Australia and New Zealand and International Association for Mathematics and Computers in Simulation, 2540–2545.

Wang, L., and H. Liu. 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:10.1080/13658810500433453.

Zhou, Guiyun, Xiaoli Liu, Suhua Fu, and Zhongxuan Sun. 2017. "Parallel Identification and Filling of Depressions in Raster Digital Elevation Models." *International Journal of Geographical Information Science*, 1–18. doi:10.1080/13658816.2016.1262954.

Zhou, Guiyun, Zhongxuan Sun, and Suhua Fu. 2016. "An Efficient Variant of the Priority-flood Algorithm for Filling Depressions in Raster Digital Elevation Models." *Computers & Geosciences* 90: 87–96. doi:10.1016/j.cageo.2016.02.021.

Zhu, Qing, Yixiang Tian, and Jie Zhao. 2006. "An Efficient Depression Processing Algorithm for Hydrologic Analysis." *Computers & Geosciences* 32 (5): 615–623. doi:10.1016/j.cageo.2005.09.001.