

Computing water flow through complex landscapes – Part 3: Fill-Spill-Merge: Flow routing in depression hierarchies

Richard Barnes^{1,2,3}, Kerry L. Callaghan^{4,5}, and Andrew D. Wickert^{4,5}

¹Energy & Resources Group (ERG), University of California, Berkeley, USA

²Electrical Engineering & Computer Science, University of California, Berkeley, USA

³Berkeley Institute for Data Science (BIDS), University of California, Berkeley, USA

⁴Department of Earth & Environmental Sciences, University of Minnesota, Minneapolis, USA

⁵Saint Anthony Falls Laboratory, University of Minnesota, Minneapolis, USA

Correspondence: Richard Barnes (richard.barnes@berkeley.edu)

1 **Abstract.** Depressions—inwardly-draining regions—are common to many landscapes. When there is sufficient moisture, de-
2 pressions take the form of lakes and wetlands; otherwise, they may be dry. Hydrological flow models used in geomorphology,
3 hydrology, planetary science, soil and water conservation, and other fields often eliminate depressions through filling or breach-
4 ing; however, this can produce unrealistic results. Models that retain depressions, on the other hand, are often undesirably ex-
5 pensive to run. In previous work we began to address this by developing a depression hierarchy data structure to capture the full
6 topographic complexity of depressions in a region. Here, we extend this work by presenting a Fill-Spill-Merge algorithm that
7 utilizes our depression hierarchy data structure to rapidly process and distribute runoff. Runoff fills depressions, which then
8 overflow and spill into their neighbors. If both a depression and its neighbor fill, they merge. We provide a detailed explanation
9 of the algorithm as well as results from two sample study areas. In these case studies, the algorithm runs 90–2,600× faster
10 (with a 2,000–63,000× reduction in compute time) than the commonly-used Jacobi iteration and produces a more accurate
11 output. Complete, well-commented, open-source code with 97% test coverage is available on Github and Zenodo.

12 1 Introduction

13 Depressions (see Lindsay (2015) for a typology) are inwardly-draining regions of a DEM that lack any outlet to an ocean or
14 other designated base elevation. Depressions occur naturally, and can be formed by glacial erosion and/or deposition (Brecken-
15 ridge and Johnson, 2009), compressional and/or extensional tectonics (Reheis, 1999; Hilley and Strecker, 2005), and cratering
16 (Cabrol and Grin, 1999). They often host lakes and wetlands by retaining water locally. Depressions may themselves contain
17 depressions. Such regions confound algorithms for geomorphological and terrain analysis, as well as those for hydrological
18 modeling, because many such algorithms simply route water down topographic slope following the local gradient: depressions
19 neither fill with water, nor drain.

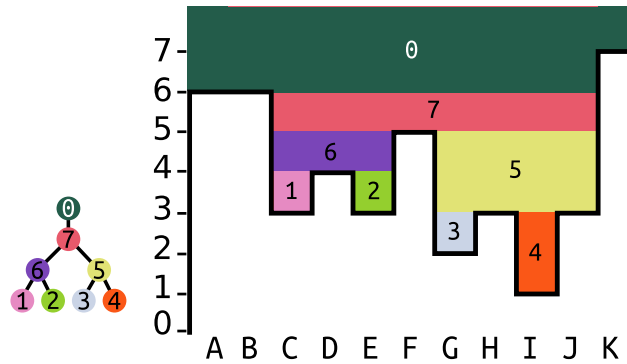
20 Many hydrological models deal with the complexity of depressions by removing them. This can be done either by filling the
21 depressions with earth so that they form a flat region of landscape (e.g. Jenson and Domingue (1988); Martz and Jong (1988));
22 breaching (Martz and Garbrecht, 1998) or carving them (Soille et al., 2003) so that water flows from their lowest point through

23 the carved channel and onward to downstream regions; or some combination of these (Lindsay and Creed, 2005b; Schwanghart
24 and Scherler, 2017; Soille, 2004; Lindsay, 2016). This approach is justified for situations in which spatiotemporal aspects of
25 the analysis allow depressions to be ignored or for cases in which all depressions can be considered to be data errors (Lindsay
26 and Creed, 2005a). Historically, many DEMs were constructed from sparse data, and small data errors produced depressions,
27 especially in flat areas (O’Callaghan and Mark, 1984). Such an assumption is no longer justified, as improved and increasingly
28 high-resolution data have become available (Li et al., 2011). Even coarse-resolution data are capable of resolving real-world
29 depressions (e.g. Riddick et al., 2018; Wickert, 2016). With this in mind, new approaches are beginning to be examined,
30 particularly in post-glacial landscapes where depressions have a significant impact on local hydrology (e.g., Lai and Anders,
31 2018) and therefore cannot be ignored during modeling.

32 FlowFill (Callaghan and Wickert, 2019) began to combat this problem by routing water across landscapes in a way that
33 conserved water volume, creating flow-routing surfaces that could still contain real depressions. Under reasonable runoff con-
34 ditions, their results show landscapes that still contain depressions and disrupted flow routes. The FlowFill method iteratively
35 routes water from higher to lower terrain. As depressions fill, they pose an extreme challenge to such a method: since water
36 seeks a level surface, the surface of a filled depression must eventually become flat and any fluid flowing onto the surface
37 diffuses across it. Even for moderately-sized surfaces it can take many iterations for a solver to reach steady state; we provide
38 a theoretical analysis of this in Section 4.1. Runtimes for FlowFill ranged from seconds to days: large datasets quickly became
39 unwieldy. Of those examples tested by Callaghan and Wickert (2019), the slowest was a dataset of 4,176,000 cells which took
40 approximately 33 hours for FlowFill to process. In contrast, the Fill-Spill-Merge algorithm presented here fills a similarly-sized
41 dataset in 8.7 s.

42 Other authors have considered the problems of extracting nested depression hierarchies and dynamically routing water
43 through them. However, these previous approaches are either slow, inexact, or both; additionally, most previous efforts were
44 not accompanied by source code, limiting their utility. Barnes et al. (2020) provide a more thorough literature review which
45 we briefly recap here. A hierarchical segmentation by Beucher (1994) did not produce a data structure on which flow could
46 be routed. Salembier and Pardas (1994) generated a hierarchical segmentation by repeatedly simplifying source images; hy-
47 drologically speaking, this can lead to unacceptable degradation of terrain information. Arnold (2010) developed an algorithm
48 similar to the one here, but without source code; the algorithm also generates looping topologies that require correction. Wu
49 et al. (2015) and Wu and Lane (2016) constructed depression hierarchies by first smoothing a DEM and then extracting vector
50 contour lines from it. Wu et al. (2018) build on this approach by discretizing the DEM into a number of horizontal slices. Both
51 approaches sacrifice exactness and the latter requires $O(N^2)$ time. Cordonnier et al. (2018) use planar graph minimum span-
52 ning trees to construct a hierarchy of depressions, but do not produce a data structure water can be routed on. In contrast, the
53 Fill-Spill-Merge algorithm relies on a well-defined data structure (Barnes et al., 2020); has complete, well-commented source
54 code with extensive correctness tests (Barnes and Callaghan, 2019, 2020); has strong efficiency guarantees (§4.1) which are
55 realized on actual and simulated datasets (§4.2); and provides exact answers.

61 To achieve this, we developed a data structure—the *depression hierarchy*—which represents the topologic and geographic
62 structure of depressions. In an accompanying paper, we provide details concerning how a depression hierarchy is constructed (Barnes



56

57 **Figure 1. A single subtree of a depression hierarchy and the depression it represents.** Depressions 1–4 are leaf depressions. Depression
 58 6 is a parent depression (also termed a meta-depression) that contains depressions 1 and 2. Water from the plateau on the left above cells *A*
 59 and *B* might *fill* Depression 1 (cell *C*), causing it to *spill* into Depression 2 (cell *E*). Only when both depressions are full do they *merge* and
 60 begin filling Depression 6 (cells *C*, *D*, and *E*). Modified from Barnes et al. (2020).

63 et al., 2020). In this paper, we explain how a depression hierarchy can be leveraged to accelerate hydrological models using a
 64 paradigm we call *Fill-Spill-Merge*.

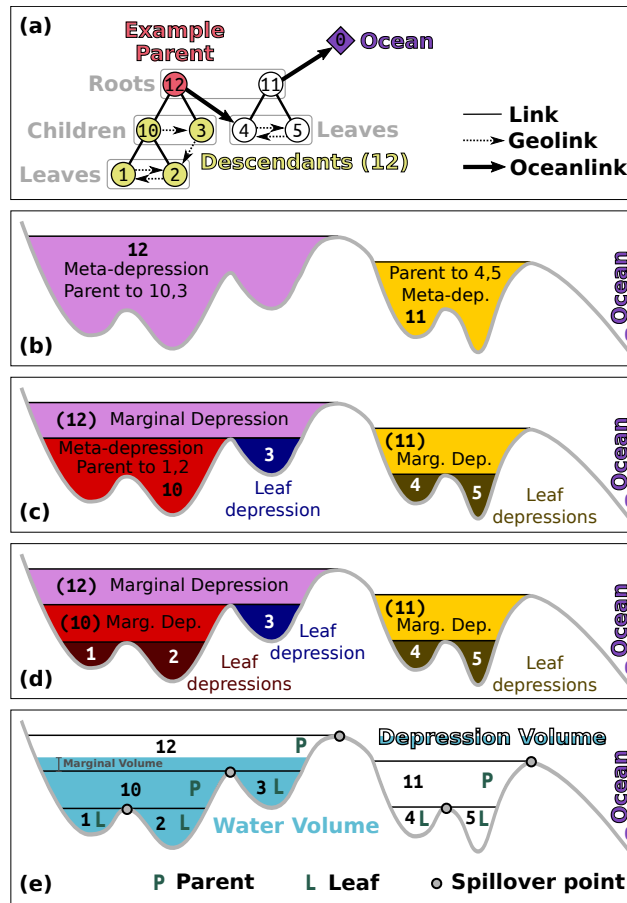
80 2 Using The Depression Hierarchy

81 Many of the techniques in this paper are based on binary tree data structures and their traversals. Although we define terms
 82 below, more complete explanations and visual examples can be found in the text for any introductory undergraduate course
 83 on data structures. We recommend Skiena (2008) and Sedgewick and Wayne (2011) as good references. In particular, a good
 84 understanding of recursion will be helpful.

85 2.1 Terminology

86 Depressions can themselves contain depressions, as shown in Figure 1. A depression hierarchy (DH) is a data structure repre-
 87 senting a forest of binary trees, as shown in Figure 2a, that represents the relationships between depressions (Figure 2a–d). Each
 88 node in the DH represents a depression. Nodes higher in the DH are depressions that themselves contain depressions; we term
 89 these *meta-depressions*. Although the depression hierarchy could be generalized to n-ary trees using multiple flow direction
 90 routing, the binary simplification is sufficient to cover most use cases. A node in the DH can have several classifications:

- 91 – **Parent:** A node, such as #10 and #12 in Figure 2a, that represents a meta-depression, and whose topological descendants
 92 therefore also form depressions.
- 93 – **Child:** A depression, such as both #10 and #1 in Figure 2a, that geographically and topologically exists within the
 94 meta-depression formed by its parent.



65

66 **Figure 2. Terminology for the depression hierarchy and water flow through it.** The depression hierarchy shown here is drawn from the
 67 left hand side of Figure 1 from the companion paper by Barnes et al. (2020). **(a)** Topology. A *parent* and its *descendants* are associated with
 68 depressions (b–d). Direct descendants are called *children*. *Leaves* are the terminal members of the depression hierarchy; they have no children
 69 and represent simple depressions (i.e., those that are not meta-depressions). Members of a single *binary tree* are joined in their hierarchy
 70 through *links*; directional links that represent water-spillover directions between geospatially adjacent depressions are called *geolinks*. Flow
 71 from one binary tree into another and towards the ocean follows the *oceanlinks*. Though only one binary tree is shown, the ocean may be the
 72 parent to an arbitrarily large *forest* of binary trees. **(b)** Parents in the hierarchy form *meta-depressions* — depressions that encompass other
 73 depressions. **(c)** These meta-depressions contain *leaf depressions* — depressions that themselves contain no depressions. These are associated
 74 with leaves in the depression hierarchy. Meta-depression 12 also contains another meta-depression, 10. The regions of Depressions 11 and
 75 12 that lie above their child depressions are termed “marginal depressions”. **(d)** Meta-depression 10 contains leaf depressions 1 and 2. **(e)**
 76 Using the depression hierarchy to simulate water flow. Water first fills *leaf depressions* before flooding into neighboring *depressions*. Once
 77 a depression and its neighbor are completely filled, their *parent* begins to flood. The *depression volume* is the full geometric volume of the
 78 depression. The *water volume*, naturally, is the volume of water within a given depression. The *marginal volume* is the volume of water
 79 partially filling the top-level meta-depression; appropriately spreading this water across the landscape is the topic of Section 3.3.

95 – **Leaf:** A depression, such as #1 and #2 in Figure 2a and Figure 2d, that has no children. The leaves of the binary trees
96 represent the smallest, most deeply-nested depressions. If a landscape were initially devoid of water, then water flowing
97 down slopes would begin to collect in some subset of these leaf depressions before it would begin to fill their parent
98 depressions.

99 – **Root:** A depression, such as #0, #11, and #12 in Figure 2, that has no parent. This term may also refer to any node that
100 is used as the starting point for a traversal that only considers the node and its descendants.

101 – **Descendant:** A child of a given parent, or the child of a child of that parent, and so on. In Figure 2a, #1, #2, #3, and #10
102 are all descendants of #12.

103 – **Sibling:** Every node has either no children (leaf nodes) or two children. Nodes which share a parent are siblings. In
104 Figure 2a, #1 and #2 are siblings, as are #4 and #5.

105 As depressions fill, their water surfaces eventually reach a *spill elevation* (Figure 2e) at which they overflow into neigh-
106 boring depressions. During this spilling, water flows from a depression into a geographically neighboring leaf depression,
107 topologically connected by a *geolink*. The spill elevations in Figure 1 are the highest points of each band of color.

108 Each node in the DH is associated with several properties:

109 – **Depression volume:** This is the *total* volume of water that the depression, including all of its descendants, can contain
110 before spilling over.

111 – **Water volume:** This is the *total* volume of water *actually being stored* in the depression. A parent depression will have
112 a non-zero water volume only if both of its children are completely full and the parent itself contains some additional
113 volume of water. In this case, the water volume will be the sum of the water volumes of the children and the additional
114 margin of water contained within the parent (i.e., the “marginal volume” indicated on Figure 2e). Parents whose children
115 are not both filled with water will have a water volume equal to zero. In this way, we can use this property to determine
116 which portions of the DH are fully or partially filled, and which are the highest water-containing nodes in any of the
117 binary trees.

118 – **Geolink:** When a depression spills, its water passes into the subtree rooted by its sibling. However, in a full model of
119 flow, the water would move downslope from the spill cell into whichever leaf depression of the sibling is geographically
120 proximal to the spill cell. *Geolinks* are pointers from depressions higher in the DH to the leaf depressions that receive
121 their water if they overflow. These are the dashed lines shown in Figure 2a. Geolinks are similar to the connections used
122 in a threaded binary tree (Fenner and Loizou, 1984).

123 – **Oceanlink:** Depressions high in the mountains may overflow down escarpments to depressions far below. In this case,
124 the depressions do not overflow into each other: the relationship is one-way. There can be multiple such escarpments, so
125 this can happen multiple times. In such cases, each group of depressions forms a proper binary tree. However, the root

126 of one of the trees has an *oceanlink* to a leaf node of the downstream binary tree. In Figure 2, both #11 and #12 are the
127 root nodes of a set of nested depressions. #12 has an oceanlink (heavy arrow) to #4, one of the leaf depressions of #11.
128 #11 itself has an oceanlink to the ocean. In many of the algorithms discussed below, oceanlinked nodes are processed
129 similarly to children.

130 Within the algorithm, oceanlinks and geolinks are used for different purposes: an oceanlink tells us that the depression in
131 question has grafted onto the leaf node of another tree of the depression hierarchy, locating a route for overflowing water to
132 eventually reach the ocean. The depression to which it is oceanlinked is considered its parent, but it is not the child of that
133 depression because water flows only one way along an oceanlink. In Figure 2a, depression #4 can be considered the parent
134 of #12, but #12 is not the child of #4. This is because #12 is not physically contained within #4, but #12 will send all of its
135 overflowing water to #4, as shown in Figure 2b–e. #4 will not contain the total water volume contained within #12, unlike other
136 parents. Geolinks route water within geographically adjacent depressions contained in the same meta-depression.

137 2.2 Traversals

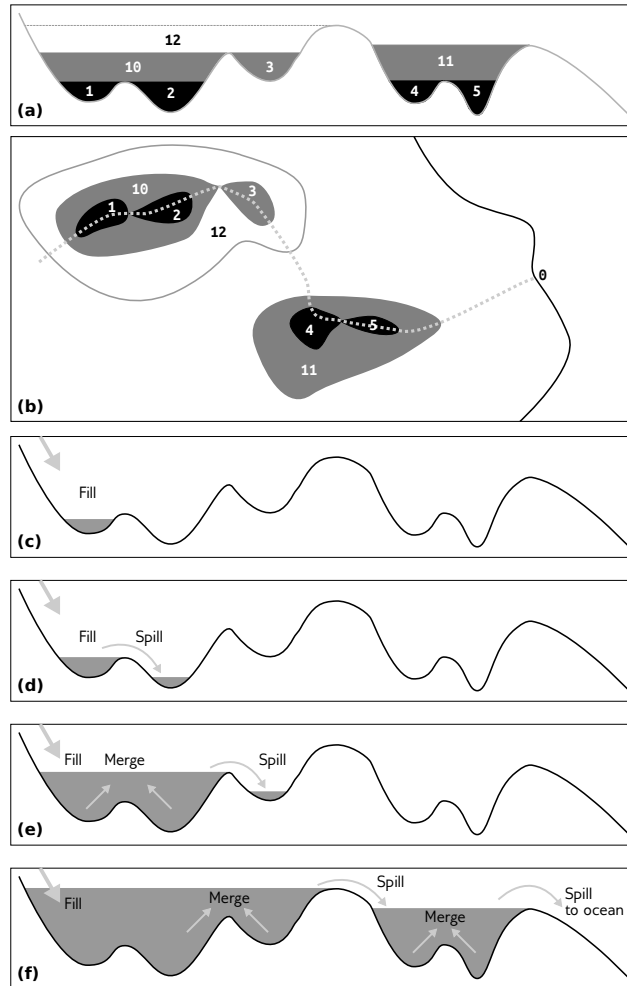
138 With these linkages in place, we can consider various ways of traversing the trees. Given a binary tree T with left and right
139 children $T.L$ and $T.R$, a breadth-first traversal considers both $T.L$ and $T.R$ before considering any of $T.L.L$, $T.L.R$, $T.R.L$,
140 or $T.R.R$. A depth-first traversal, on the other hand, will consider $T.L$ and all of its descendants before considering $T.R$ or any
141 of its descendants. The tree traversals we perform in this paper are all depth-first.

142 Depth-first traversals are most naturally expressed via recursion and come in three types: in-order, pre-order, and post-order.
143 Let a recursive traversal function be called $r(\cdot)$ and the processing we perform on a particular node in the tree $p(\cdot)$, then the
144 traversals are given by:

- 145 – in-order: $r(T.L)$ then $p(T)$ then $r(T.R)$
- 146 – pre-order: $p(T)$ then $r(T.L)$ then $r(T.R)$
- 147 – post-order: $r(T.L)$ then $r(T.R)$ then $p(T)$

148 3 The Algorithm

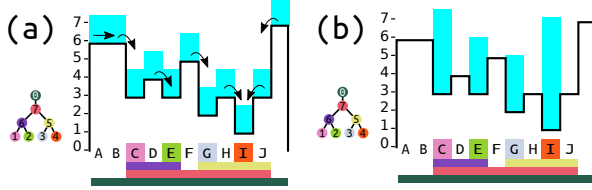
195 The Fill-Spill-Merge algorithm consists of several steps, outlined here, depicted in Figures 3 and 4, and shown in flowchart
196 form in Figure 5. This paper is also accompanied by complete, well-commented source code; the reader may find it helpful to
197 download this code and refer to it as an additional reference. First (§3.1), surface water needs to move downhill, either to the
198 ocean (i.e., a designated sink region or the map edge) or to collect in pit cells – the deepest points within leaf depressions. Note
199 that the landscape may already have standing water at this stage. This operation takes place across all the cells of the DEM.
200 Second (§3.2), water is redistributed across the depression hierarchy such that any depressions that have filled sufficiently spill
201 over into neighboring depressions and, if both depressions are full, flood their parent to merge into a single, larger body of



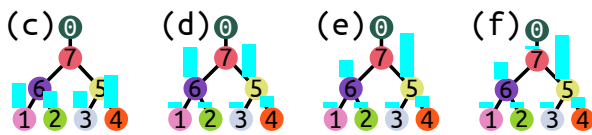
149

150 **Figure 3. Fill-Spill-Merge process.** Water moves through topographic depressions by filling them, spilling over sills, and merging to form
 151 meta-depressions. (a) Topographic cross section with labeled leaf depressions and their parents, following the left-hand side of the de-
 152 pression hierarchy in Figure 2. “0” represents the ocean; other numbers represent leaves and parents that together form depressions and
 153 meta-depressions. (b) Map showing this depression structure; the cross-section in (a) follows the dotted gray line. (c) A water source to the
 154 left begins to fill Depression 1. (d) Continued water input causes Depression 1 to overflow and spill into Depression 2. (e) Depression 2 fills,
 155 causing Depressions 1 and 2 to fill their parent (10) and merge to form a metadepression. This metadepression overflows into Depression 3.
 156 (f) Depression 3 fills and merges with Meta-Depression 10 (1 and 2 being implied members based on their position in the hierarchy) to flood
 157 their parent, 12. After Meta-Depression 12 overflows, it enters Depression 4, which then fills and spills into Depression 5. After Depression
 158 5 floods, its waters join with those from Depression 4 to fill Meta-Depression 11, which then spills to the ocean. Figures 4 and 5 describe the
 159 algorithm in more specific detail.

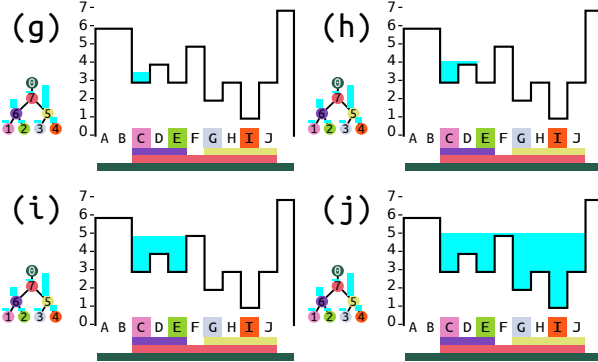
Move water downhill to pits



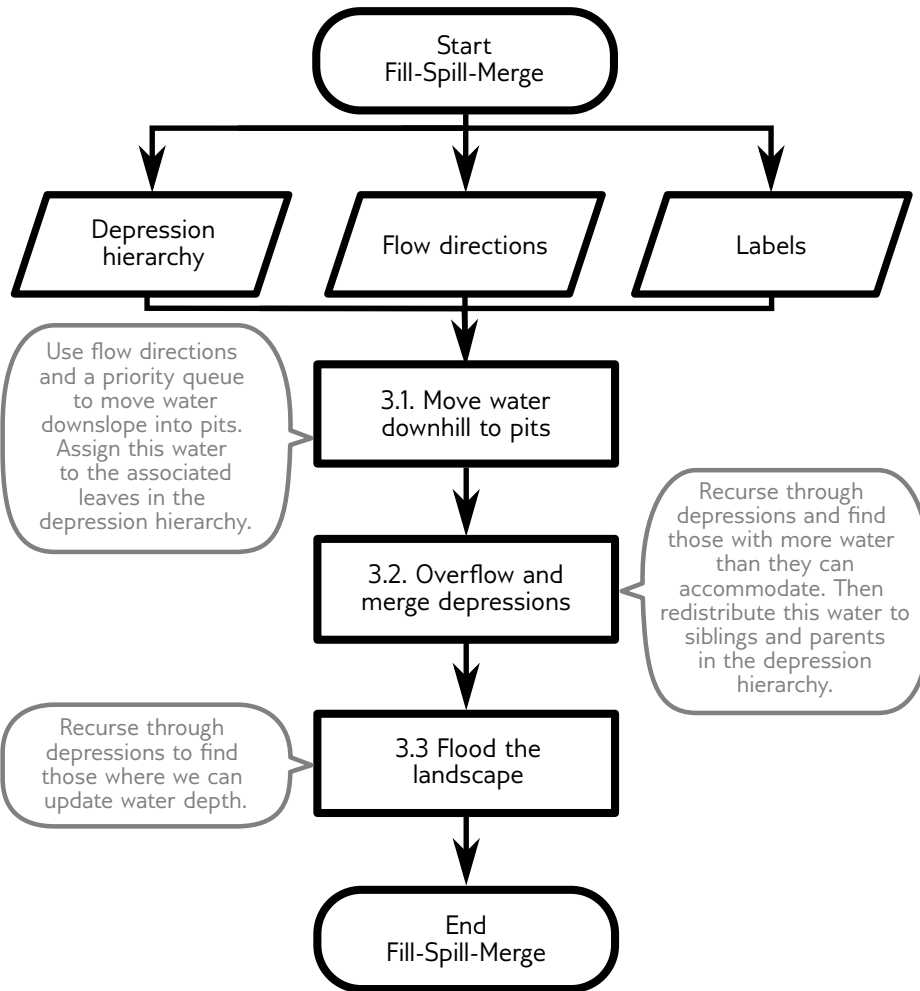
Overflow and merge depressions



Flood the landscape



160 **Figure 4. Visual Overview of the Algorithm.** Black outlines repre-
 161 sent the elevations of the cells. Blue areas are the heights of water in
 162 each cell or depression within the depression hierarchy. Capital let-
 163 ters label cells, and numbers on colored dots label depressions. Col-
 164 ors at the base of each panel match the colored dots and indicate to
 165 which depression each cell belongs. The algorithm consists of three
 166 major stages (Figure 5). From its initial distribution (a), water is moved
 167 downhill following flow directions in the steepest downslope direction
 168 from each cell, as indicated by the arrows. Water continues to move
 169 downslope until it reaches the pit cells (b, §3.1). Water is then moved
 170 within the depression hierarchy (c–f, §3.2). (c) shows the initial distri-
 171 bution of water within the depression hierarchy, based on how much
 172 water was in the pit cell of each depression. Water in depressions with
 173 insufficient volume overflow first into their sibling depressions and
 174 then – if the sibling depression becomes filled – passes to their par-
 175 ents. All of the leaf depressions in (c) are completely filled, so no sib-
 176 ling depressions can accommodate more water. Therefore, depressions
 177 1 and 2 pass their overflowing water up to their parent, depression 6,
 178 and depressions 3 and 4 pass their overflowing water up to their par-
 179 ent, depression 5. (d) Depression 6 is now overflowing, but its sibling,
 180 depression 5, is not full, so depression 6 passes as much of its over-
 181 flowing water as it can to depression 5. (e) Once depression 5 is full,
 182 some overflowing water still remains, so this is passed to the parent,
 183 depression 7. (f) In this case, depression 7 is able to accommodate the
 184 remainder of the water. Had depression 7 also overflowed, the leftover
 185 water would have overflowed into the ocean and been disregarded. Dep-
 186 ressions to be flooded are then identified and flooded (§3.3). Since
 187 depression 7 contains water, we know that all of its descendants must
 188 be completely full. Therefore, we can flood these all at the same time,
 189 on the level of depression 7. Any one of the pit cells within depression
 190 7 is arbitrarily selected as the starting point (g). More cells are added
 191 until all of the water has been accommodated. (h–j) are a visual rep-
 192 resentation of this process, although the algorithm would first locate
 193 affected cells C–J, and then calculate the final height of water in all of
 194 these cells in a single step.



205

206 **Figure 5. Flowchart showing the main steps taken by the algorithm.** These steps are described in more detail in §3.1 to §3.3.

202 water within a meta-depression. This operation is done without explicitly considering the cells of the DEM, which makes it
 203 very fast. Third and finally (§3.3), the water within the depression hierarchy is translated into an extent and depth of flooding
 204 across the topographic surface (DEM).

207 Computing a depression hierarchy (Barnes et al., 2020) is a necessary precursor to running Fill-Spill-Merge. The specific
 208 outputs from the depression hierarchy algorithm that are used in the Fill-Spill-Merge algorithm are:

- 209 – *DH*: the depression hierarchy itself.
- 210 – *Flowdirs*: a matrix of flow directions, indicating which of a cell’s neighbors receives its flow. Because Priority-Flood
 211 (Barnes et al., 2014) is used to generate the depression hierarchy, flat areas are automatically resolved.
- 212 – *Labels*: a matrix indicating the leaf depression to which each cell belongs.

213 By routing water according to the DH, we significantly accelerate the compute speed and ensure that the full network of
214 depressions is a topologically correct directed tree. Each of the following subsections details one of the numbered steps along
215 the central path of the flowchart shown in Figure 5.

216 **3.1 Move Water Downhill to Pits**

217 We route water in a similar way to standard flow-accumulation algorithms (Mark, 1988; Wallis et al., 2009; Barnes, 2017), but
218 for completeness summarize our approach here. Flow directions for each cell have already been identified by the depression
219 hierarchy algorithm. Each cell calculates how many of its neighbors flow into it. We call this value the cell's dependency count,
220 as it describes the number of immediate upstream cells whose flow accumulation must be resolved before flow accumulation
221 at the given cell can be computed. Local maxima in the DEM are identified as those cells that receive no flow from any
222 neighbor. These local maxima are placed in a queue. Cells are then popped (i.e., noted while being removed) from this queue.
223 The cells determine how much flow they generate locally (perhaps referring to a matrix of rainfall values, but also including
224 existing stores of standing water) and add this to their flow accumulation value. They then add their flow accumulation to
225 their downstream neighbor's and set their own flow accumulation value to zero. The neighbor's dependency count is then
226 decremented. If the neighbor's dependency count has reached zero during this step, it is added to the end of the queue. This
227 process of accumulating flow, passing it downstream, decrementing the dependency count, and adding cells to the queue
228 continues until the queue is empty, at which point every cell on the map has been visited and any water has been moved
229 downslope. Braun and Willett (2013) present an alternative formulation based on a depth-first traversal, but Barnes (2019)
230 demonstrates that a breadth-first ordering, such as that presented here, is better suited to parallelism.

231 When the accumulated flow reaches the pit cell of a depression, the downhill-directed flow routing stops because there is no
232 downhill neighbor to receive the flow. At this point, all of the flow-accumulated water in the pit cell is moved into the pit cell's
233 associated leaf depression in the DH. That is, the water is moved out of the geographic space and into the topologic space. This
234 then enables mass-conserving depression flooding via rapid Fill-Spill-Merge calculations, as detailed below.

235 **3.2 Overflow and Merge Depressions**

236 At this point, the Fill-Spill-Merge algorithm has routed all of the surface water into either the ocean or into the leaf nodes of the
237 DH. The next step is to redistribute this water through the DH to nodes with enough volume to contain the water, and to send
238 any excess water to the ocean. This set of operations can be performed entirely in the depression hierarchy without reference
239 to the digital elevation model.

240 Intuitively, the process of filling, spilling, and merging can be visualized as occurring from leaf nodes to their parents
241 (Figure 3). When a leaf depression initially contains more water than it can hold, the water will be redistributed by spilling
242 over into the neighboring depression. If this neighboring depression is already full, then the excess water must pass to the
243 parent of both the depression and its neighbor. This process continues recursively until either the supplied water is exhausted
244 or this water reaches the ultimate parent, the ocean. In this latter case, all excess water is dropped from the model and the ocean
245 is unaffected.

246 To effect the intuition developed above, we need a well-defined way to visit all of the nodes in the depression hierarchy. A
247 post-order traversal allows us to visit both of a node's children and their descendants before calculating any quantities on the
248 node itself. The result is that leaves get processed before their parents. However, a single traversal is insufficient: we need one
249 traversal (the "outer" traversal) to identify nodes that have excess water and another traversal (the "inner traversal") to distribute
250 this water. The outer traversal may launch the inner traversal many times as it works its way up hierarchy. Pseudocode showing
251 these travels is available in §6.1 and §6.2.

252 To efficiently redistribute water, the Fill-Spill-Merge algorithm performs nested depth-first traversals of the DH. The outer
253 traversal (§6.1) is post-order and considers each meta-depression in turn, from the most deeply nested to the least. For each
254 meta-depression, an inner traversal (§6.2) handles its overflows by moving water to its sibling (starting by filling the sibling's
255 descendants) and, if there's any left, passing it to the depression's parent. In this way, the outer traversal maintains an invariant
256 (a property which is true before and after each call a function): any meta-depression it has processed does not contain an
257 overflow. Put another way, the outer traversal finds problems and the inner traversal fixes them.

258 The outer traversal of the DH (which is, after all, a forest of binary trees) begins with the ocean. For each depression,
259 the algorithm first recurses into its oceanlinks, if any, and then into the left and then right child. In the post-order portion of
260 the traversal (which starts from the leaves and moves back up through the depression hierarchy), the algorithm identifies any
261 depressions containing more water than they can accommodate. This process continues until the recursion returns to the ocean,
262 at which point any additional water is assumed to be added to the ocean without impacting sea level, though this total discharge
263 to the sea is recorded within the "ocean" depression.

264 When an overfilled depression is located by the outer traversal above, its water needs to be redistributed to neighbouring
265 depressions. If we call the overfilled depression D , then the water can be redistributed by starting a second, inner post-order
266 traversal at D . This inner traversal carries Excess Water from one depression to another until it has found a home for all
267 of it. When we pass water into a depression, it can go to one of three places: the depression itself, its sibling, or its parent.
268 Distributing the water to any of these places may itself cause an overflow. Therefore, the inner (pre-order) traversal comprises
269 the following steps:

- 270 1. Call the depression that we are currently considering B . This may be the depression we originally considered, depression
271 D , or it may be some other depression reached during the steps detailed below. If B is overflowing, we add the overflow
272 to the Excess Water the inner traversal is carrying. If B has spare capacity we add water from the Excess to B until either
273 it fills or all of the Excess Water the inner traversal is carrying is used.
- 274 2. At this point, the inner traversal can terminate if: (i) there is no water left, (ii) B is the parent of D , or (iii) B was reached
275 via an oceanlink.
- 276 3. Otherwise, if B has a sibling and the sibling's water volume is less than its depression volume, then start from Step 1
277 with the new B set as the depression pointed to by the current B 's geolink.
- 278 4. Otherwise, if B has no sibling or the sibling's water volume is equal to its depression volume, then start from Step 1 with
279 the new B set as the parent of the current B or, if B has no parent, then use the depression to which B oceanlinks.

280 The next step of the outer traversal, which begins one level in the DH closer to the ocean, identifies a less nested metade-
281 pression for which the inner traversal might need to be run. If this step were not supplied with information about prior water
282 redistribution, it could cause the inner traversal to cover the same nodes repeatedly, which would be computationally wasteful.
283 To prevent this, the inner traversal returns the ID of the final node in which it placed water: this node is the only node in the
284 traversal with spare capacity so future traversals can begin there. Therefore, on subsequent overflows, if such a cached value is
285 available, then the recursion skips directly to that node. This ensures that all the calls to this part of the algorithm take no more
286 than $O(N)$ time collectively.

287 The following examples uses the geometry from Figure 2 to describe a set of inner traversals, starting with an overflowing
288 Depression #12. Step numbers mirror those above; numbers in parentheses indicate the number of recursions – that is, the
289 number of times that the inner-traversal algorithm has returned to Step 1:

290 1 Depression #12 fills and overflows.

291 2 Depression #12's water overflows into Depression #4, which is not full, following its geolink.

292 1(r1) Depression #4 acts as Depression #12's parent via an oceanlink. The inner traversal terminates.

293 At this point, the outer traversal moves one level closer to the ocean, and the inner traversal repeats, this time starting at
294 Depression #4.

295 1 Depression #4 fills and overflows.

296 2 Depression #4's water overflows into its sibling, Depression #5, which is not full and is a leaf depression. If Depression
297 #5 had descendants, water overflowing from Depression #4 would have followed a geolink to one of these.

298 1(r1) Depression #5s fills and overflows.

299 2(r1) Depression #4 is full.

300 3(r1) Depression #5 overflows into its parent, Depression #11.

301 1(r2) Depression #11 overflows into the ocean; the inner traversal terminates.

302 Now the outer traversal moves yet another level closer to the ocean, and the new inner traversal starts at Depression #11.

303 1 Depression #11 fills and overflows.

304 2 Depression #11 has no sibling.

305 3 Depression #11 overflows into its parent, the ocean; all remaining excess water is absorbed into an infinite sink.

306 1(r1) The now-selected node is the ocean; the inner traversal terminates.

307 At this point, the outer traversal moves one level closer to the ocean, and arrives at the ocean. The outer traversal also terminates.

308 3.3 Flood the landscape

309 After water moves through the DH (Section 3.2, above), each node in the DH exists in one of the three following states:

- 310 1. **Empty:** The depression's water volume is equal to zero. In this case, nothing needs to be done. The depression's descen-
311 dants might contain water, but the water never propagates to this level of the DH.
- 312 2. **Full:** The depression's water volume is equal to the volume of the depression itself. In this case, the depression is entirely
313 full. If the depression's parent contains water, then the calculation of water depth is dealt with at a higher stage in the
314 DH. If the depression's parent is empty, or if the depression's parent is the ocean, then the calculation is performed at
315 this level as described below.
- 316 3. **Partially filled:** The depression's water volume is less than its depression volume. In this case, the depth of water across
317 the depression and all its descendants' cells must be calculated at this level so that the depression fills to an appropriate
318 level. This is described below and indicated as the *marginal volume* on Figure 2e.

319 The next step is to distribute this water across the DEM, appropriately flooding geographic depressions.

320 Given the three states described above, the algorithm locates the highest-level nodes which contain water. It does so via a
321 post-order traversal. Each time the traversal reaches a leaf, the algorithm notes its label and pit cell. After identifying each of
322 these, the algorithm reverses direction, moving from child to parent so long as the parent node contains water. Call the highest
323 water-bearing node within a tree L .

324 In many cases, the water volume contained within the depression will be less than the total depression volume; therefore,
325 we must calculate what the water level in the depression will be. To do this, we pick an arbitrary pit cell within L and its
326 descendants, and then use this as a seed from which to start building a priority queue which will traverse the cells of the
327 depression. The priority queue returns cells ordered from lowest to highest elevation. At each step through the priority queue,
328 the algorithm checks whether the cells visited so far collectively have enough volume to hold the water. If so, the algorithm
329 exits, having successfully defined the flooded area. If not, it continues to use the priority queue to traverse the depression cell
330 by cell. The filling procedure is shown in pseudocode in §6.3.

331 To expand this brief conceptual discussion into a more formal set of steps, let us begin by calling the active cell – that is,
332 the one that is currently being considered by the algorithm – c_p . This cell is initially the arbitrary pit mentioned above, and is
333 added to the priority queue. The algorithm marks c_p , which stands for “cell of current highest priority”, as *visited*; all other
334 cells remain unvisited. The algorithm then follows these steps:

- 335 1. Pop c_p from the priority queue, call it c , and use its elevation to calculate the volume of water that can be accommodated
336 in the set of cells processed so far (Equation 3, below). If this volume is enough to accommodate the volume of water
337 available, exit the loop and compute the final water level (Equation 6, below). Otherwise, proceed to Step 2.
- 338 2. Add c (which was popped in Step 1) to a plain queue, which records all of the cells scanned so far; these cells will later
339 be inundated.

- 340 3. Add the cells neighboring c that are not marked as *visited* to the priority queue if they belong to one of the descendant
341 depressions of the one being filled. Each of these neighboring cells is then marked as *visited*.
- 342 4. Choose the lowest-elevation cell in the priority queue and label it as the new c_p and return to Step 1. If the priority queue
343 is empty, then all cells in the same meta-depression as c_p or its descendants have been visited and we are now guaranteed
344 to have sufficient depression volume to hold all of the water.

345 Step 1 in this approach requires an efficient way to determine the volume of a depression below any given elevation. If we
346 call this elevation z_o and the depression below the outlet contains N cells with elevations $\{z_1, z_2, z_3, z_4, \dots\}$ and unit cell area,
347 the volume of water that the depression can accommodate simply equals the sum of the depth of water in each of its cells:

$$348 (z_o - z_1) + (z_o - z_2) + (z_o - z_3) + (z_o - z_4) + \dots = N z_o - z_1 - z_2 - z_3 - z_4 - \dots \quad (1)$$

$$349 = N z_o - \sum_{i=1}^N z_i \quad (2)$$

350 Now, consider cells $c_i = c_1, \dots, c_N$ in the plain queue; that is, those cells that have been visited and popped from the priority
351 queue. We can calculate the volume of water that can be accommodated in the depression below the elevation z_s of the last cell
352 c_N (the sill) as:

$$353 V_{dep, z_s} = z_s \sum_{i=1}^N a_i - \sum_{i=1}^N z_i a_i \quad (3)$$

354 where z_i is the elevation of cell c_i and a_i is the area of cell c_i . Thus, if we keep running sums while traversing the depression,
355 it is possible to directly calculate the volume of water the depression can hold at each point in the traversal.

356 Once V_{dep, z_s} is greater than or equal to the volume of water in the depression, V_w , the plain queue contains all the cells
357 to be flooded. At this point, the algorithm updates z_w , which is the water level within this depression. If $V_w = V_{dep, z_s}$, the
358 algorithm sets $z_w = z_N$. If instead $V_w < V_{dep, z_s}$, the available volume in the depression is greater than the water volume, and
359 the algorithm calculates z_w in the depression as follows:

$$360 V_w = z_w \sum_{i=1}^N a_i - \sum_{i=1}^N z_i a_i \quad (4)$$

$$361 z_w \sum_{i=1}^N a_i = V_w + \sum_{i=1}^N z_i a_i \quad (5)$$

$$362 z_w = \left(\sum_{i=1}^N a_i \right)^{-1} \left(V_w + \sum_{i=1}^N z_i a_i \right) \quad (6)$$

363 We call Equation 6 the Lake-Level Equation (LLE). If all cells have a unit area, this simplifies to:

$$364 z_w = \frac{1}{N} \left(V_w + \sum_{i=1}^N z_i \right) \quad (7)$$

365 The conditional usage of the LLE described above is purely for computational efficiency: if $V_w = V_{\text{dep}, z_s}$, its solution is that
366 $z_w = z_N$.

367 After solving for the water-surface elevation, the algorithm pops each cell in the plain queue ($c_i = c_1, \dots, c_N$), corresponding
368 to the flooded region, and sets its water elevation to the computed z_w . This is the final step of the Fill-Spill-Merge algorithm. At
369 this point, it outputs a file representing the topography plus water thickness across the domain (i.e., topography with depressions
370 filled or partially filled with water).

371 Because Fill-Spill-Merge routes water cell-by-cell to the pit cells of depressions and manages an array of water depths, it
372 can be adapted for use with groundwater models, such as that described by Fan et al. (2013).

373 4 Algorithm performance

374 4.1 Theory

375 Here we use computational complexity as a means of contrasting the expected run-time of our algorithm against previous
376 algorithms such as FlowFill (Callaghan and Wickert, 2019). To do so, we describe a simple iterative solver similar to FlowFill
377 whose goal is to determine an appropriate water level for a depression. The solver operates on a one-dimensional domain of
378 cells bounded by high cliffs on either side in which each cell may have a column of water. At each step, if the solver finds a
379 discontinuity in water levels between two cells, it responds by averaging the heights of the cells' water columns. (The solver
380 we describe is known as Jacobi's method.) The challenge we present to this solver is a direct analogue of routing flow along a
381 stretch of river with negligible gradient and is very similar to routing flow across the surface of a lake or ocean.

382 For our analysis, we imagine that the system is initialized with a high column of water on the left and no water anywhere
383 else. We call the cell with the water Cell 1. We call the cells to its right 2, 3, 4, and so on. During the solver's first step, Cell 1
384 is initialized. On its second step, Cell 1 averages its height with Cell 2. On the third step, Cell 2 averages with Cell 3 and Cell
385 1 then averages with Cell 2. On the fourth step, Cell 3 averages to 4, 2 averages to 3, and 1 averages with 2. Thus, the number
386 of cells affected at each step are: 1, 2, 3, 4, and so on. Since there must be *at least* as many steps as there are cells, we can say
387 that there are N steps. The total time, t_{compute} , is then

$$388 \quad t_{\text{compute}} = \sum_{i=1}^N i = \frac{N(N+1)}{2} \quad (8)$$

389 Thus, for any model (Callaghan and Wickert, 2019; Fan et al., 2013) that uses a scheme similar to our simple solver, the time
390 required to solve the model is in $O(N^2)$.

391 In contrast, the new algorithm runs in $O(N \log N)$ time in the worst case. Moving water downhill (Section 3.1) is a flow-
392 accumulation algorithm. This is known to take $O(N)$ time (Mark, 1988) and efficient variants exist for performing flow
393 accumulation in parallel on large datasets (Barnes, 2017) and on GPUs (Barnes, 2019), though for simplicity we do not use
394 these techniques here. Moving water within the depression hierarchy (Section 3.2) requires a depth-first post-order traversal of
395 the entire hierarchy. This type of traversal is a foundational algorithm in computer science and takes $O(N)$ time. Each node

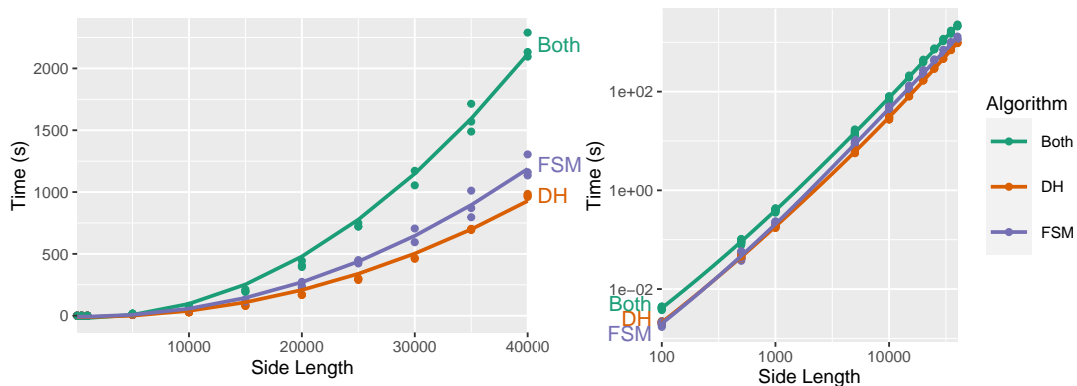


Figure 6. Performance on synthetic data. The left-hand plot shows the data on linear axes and the right-hand plot on log-log axes. The number of cells in each dataset is the square of the side length. The lines show $N \log N$ fits to each algorithm’s time ($R^2 \approx 0.99$ for each). “DH” shows the performance of the Depression Hierarchy algorithm while “FSM” shows that of the Fill-Spill-Merge algorithm; “Both” shows the addition of these two values.

396 in this traversal has the potential to overflow, which also results in a depth-first traversal, thereby requiring up to $O(N)$ time.
397 However, by using a jump table that persists between calls to the overflow function, we ensure that it is able to identify the
398 target of the overflow in amortized constant time; that is, the function is able to skip over fully-filled depressions. Finally, the
399 algorithm floods the digital elevation model from the pit cells up. This requires a depth-first post-order traversal, which calls
400 a flooding function (Section 3.3) on select subtrees of the DH. The depth-first traversal takes $O(N)$ time, as described above.
401 The priority queue used for flooding nominally takes $O(N \log N)$ time in the worst case for floating-point data and $O(N)$
402 time in the worst case for integer data (Barnes et al., 2014). However, with specialized data structures the time can be reduced
403 to $O(N)$ for both floating-point and integer data (Barnes et al., 2014). Most real datasets consist of many small depressions
404 whose cell counts $N_{\text{cells-in-dep}}$ are much smaller than the total number of cells in the digital elevation model. Therefore, the
405 actual time is for this step is $O(N_{\text{dep}} N_{\text{cells-in-dep}})$, where N_{dep} is the total number of depressions and $N_{\text{dep}} N_{\text{cells-in-dep}}$ can
406 be much less than N . Because the worst-case time complexity of any operation is $O(N)$, this bounds the time of the algorithm
407 as a whole. However, to reduce the potential for bugs, we use the C++ standard library’s $O(N \log N)$ priority queue in our
408 implementation, at the cost of reduced performance.

409 To put this in more concrete terms, consider a long stretch of low-gradient river. Such a feature poses a lower bound on the
410 time of our simple solver. North America’s Red River of the North runs for 885 km with a gradient that is often on the order of
411 0.03 m km^{-1} . On a 90 m grid of floating-point data, the river would be 9,833 cells long. Our simple (Jacobi) solver would then
412 take an estimated 97 million time units to reach a solution, whereas the new solver that we describe in this paper would take
413 9,833 time units, a $10,000\times$ speed-up. Our empirical results, below, support both the theory and this expected value.

Dataset	Dimensions	Cells	FSM Time [s]	Total Time [s]
Madagascar	2000×1000	$2.0 \cdot 10^6$	0.1	0.4
U.S. Great Basin	1920×2400	$4.6 \cdot 10^6$	0.2	8.7
Australia	5640×4200	$2.3 \cdot 10^7$	9.1	15.6
Africa	9480×9000	$8.5 \cdot 10^7$	65.3	118.0
N&S America	18720×17400	$3.2 \cdot 10^8$	53.2	231.6
Minnesota 30m topobathy	34742×23831	$8.2 \cdot 10^8$	307.8	792.6

416 **Table 1. Datasets used, their dimensions, and algorithm wall-times.** Tests were performed on the Comet cluster run by XSEDE (see
417 main text for full specifications). Times for Fill-Spill-Merge (“FSM Time”) alone and this time plus the depression hierarchy construction
418 time (“Total Time”) are shown. Topographic data for Madagascar, the U.S. Great Basin, Australia, Africa, and North & South America,
419 were clipped from the global GEBCO_08 30-arcsecond global combined topographic and bathymetric elevation data set (GEBCO, 2010).
420 The Minnesota 30m topobathy data is the merged result of two data sources. The topography is resampled from the Minnesota Geospatial
421 Information Office’s 1m LiDAR Elevation Dataset (MNGEO - Minnesota Geospatial Information Office, 2019). Bathymetric data were
422 provided by the Minnesota Department of Natural Resources (MNDNR - Minnesota Department of Natural Resources, 2014). Richard
423 Lively of the Minnesota Geological Survey merged and combined these data sets.

414 4.2 Computational Performance

424 We have implemented the algorithm described above in C++17 using the Geospatial Data Abstraction Library (GDAL) (GDAL
425 Development Team, 2016) to read and write data. There are 924 lines of code of which 50% are or contain comments. The
426 code can be acquired from <https://github.com/r-barnes/Barnes2020-FillSpillMerge> and Zenodo (Barnes and Callaghan, 2020).
427 The code contains extensive unit and end-to-end tests, which leverage both deterministic and random testing; the code passes
428 a total of 214,990 test assertions and achieve 97% test coverage. The missed lines flag emergency situations which can only
429 arise if there is a logic error, so they (in theory) cannot be reached.

430 Tests were run on the Comet machine of the Extreme Science and Engineering Discovery Environment (XSEDE) (Towns
431 et al., 2014). Each node of the machine has 2.5 GHz Intel Xeon E5-2680v3 processors with 24 cores per node and 128 GB of
432 DDR4 DRAM. Code was compiled using GNU g++ 7.2.0 with full optimizations enabled.

433 We ran two sets of scaling tests, one on actual data and one on synthetic data. On actual data, our scaling tests cover datasets
434 spanning three orders of magnitude in terms of their number of cells, as shown in Table 1. The R package GuessCompX Agenis-
435 Nevers et al. (2019) shows that an $O(N \log N)$ scaling relationship gives the best fit to the data, which agrees with the theory.

436 To more precisely demonstrate performance, we run Fill-Spill-Merge on synthetic landscapes of various sizes generated
437 using RichDEM’s Perlin noise random terrain generator (Barnes, 2018). Multiple landscapes are generated and timed at each
438 size to smooth timing variation due to both the data and fluctuations in the testing environment. This results in Figure 6, which
439 again shows that the performance data gives a good fit to an $N \log N$ function.

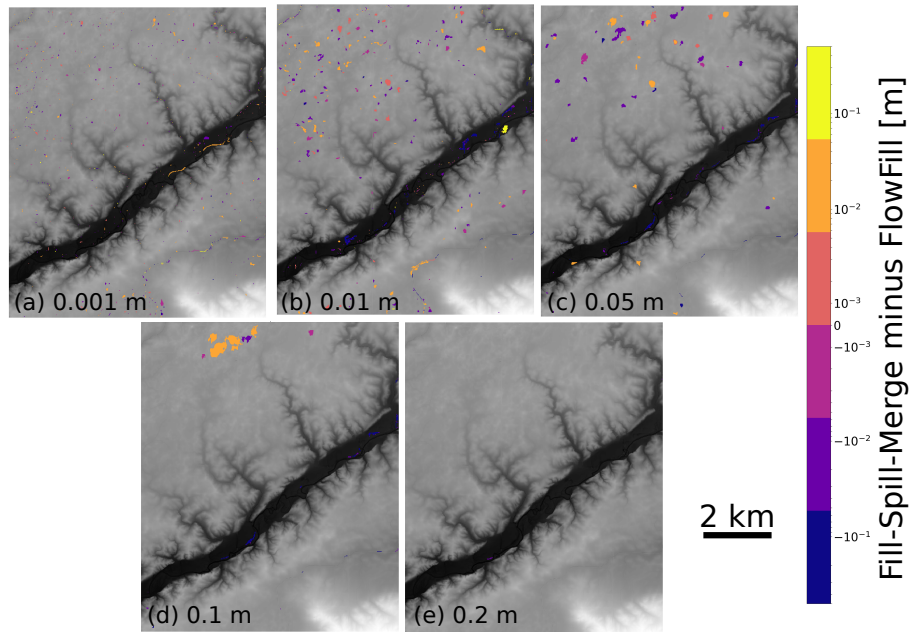
440 4.3 Model intercomparison

441 Given a depression hierarchy data structure, Fill-Spill-Merge provides an efficient method to route water across any surface
442 while taking depressions into account. Furthermore, Fill-Spill-Merge can be used to assess which depressions are most impor-
443 tant in day-to-day or seasonal changes to the hydrologic system. For example, small depressions will become flooded and spill
444 over even with relatively small amounts of water reaching them, while larger depressions may not be completely filled. These
445 depressions impact the hydrologic connectivity of the landscape (Callaghan and Wickert, 2019). If standing water is retained
446 between invocations of Fill-Spill-Merge, and new water added at each invocation, the algorithm can be used to simulate the
447 movement of water across landscapes; we will explore this further in future work.

460 We have compared Fill-Spill-Merge with a prior algorithm, FlowFill, at the same two sites used by Callaghan and Wickert
461 (2019): a reach of the Sangamon River in Illinois (Figure 7) and the Río Toro basin in Argentina (Figure 8). Like Fill-Spill-
462 Merge, FlowFill can be used to route water across a landscape while preserving real depressions, but the latter algorithm is
463 significantly slower (Table 2). The two selected study sites provide very different landscapes for testing the performance of the
464 algorithm. The Sangamon River site is located at 39.97°N, 88.72°W, in Illinois, USA. It is a low-relief, post-glacial landscape
465 containing many closed depressions, which may impact hydrologic connectivity as a function of runoff (Lai and Anders,
466 2018). It furthermore contains a grid of roads and associated embankments whose elevations are significant when compared to
467 regional relief and impact water flow paths and storage. Callaghan and Wickert (2019) resampled the 2.5 ft (0.76 m) resolution
468 LiDAR DEM (Illinois Geospatial Data Clearinghouse, 2020) to 15 m resolution for analysis and manually removed several
469 road bridges using GRASS GIS (Neteler et al., 2012) to prevent artificial pooling behind these; here we use the same modified
470 DEM to enable a direct comparison between the algorithms. The Río Toro site is located mainly in Salta Province, Argentina,
471 around 24.5°S, 65.8°W. This site exhibits more rugged fluvially sculpted topography (Hilley and Strecker, 2005). Callaghan
472 and Wickert (2019) resampled the 12-m TanDEM-X DEM of this region (Krieger et al., 2013; Rizzoli et al., 2017) to 120 m
473 resolution. Here we use this same resampled DEM for comparison. The runoff depths used at each of the two study sites were
474 selected to show a range of water levels present in the depressions. The depths shown were therefore scaled based on the
475 amount of water required to completely fill depressions in the landscape.

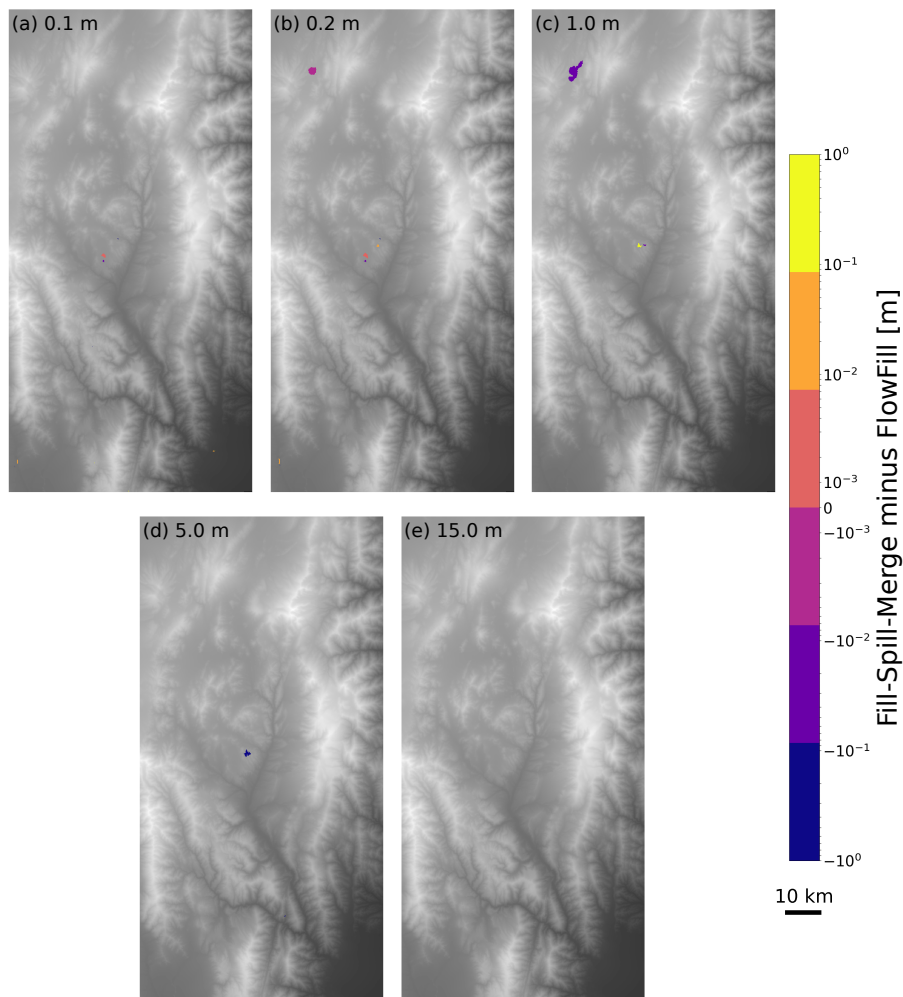
476 As shown in Table 2, wall-times using Fill-Spill-Merge ranged from 0.227–0.243 s for the Sangamon River site and 0.300–
477 0.319 s for the Río Toro site. This compares with times ranging from 20–643 s and 31–155 s, respectively, for FlowFill. These
478 times for both sites correspond to a 86–2,645× reduction in wall-time using FSM. Since FlowFill was run with 24 processors,
479 this translates to a 2,064–63,480× reduction in compute time. Considering that each of these example DEMs is quite small
480 relative to modern full-resolution LiDAR-derived elevation data sets or continental-scale 30-meter DEMs (Table 1), this speed-
481 up and its associated $O(N \log N)$ scaling provides a significant advantage for topographic analysis and solving associated
482 problems in hydrology and geomorphology.

492 Although both FlowFill and Fill-Spill-Merge route water downslope, flooding depressions based on the quantity of available
493 water, our FSM results differ in some ways from those of FlowFill (Callaghan and Wickert, 2019). In both Figures 7 and 8,
494 Fill-Spill-Merge flooded some depressions more deeply than FlowFill did and flooded some depressions with less water. At



448

449 **Figure 7.** The difference between results of Fill-Spill-Merge and FlowFill at the Sangamon River site. The values for panels (a) to (e)
 450 indicate the depth of uniform runoff applied across the landscape for both algorithms. For example, in (a), each cell across the domain starts
 451 with 0.001 m of surface water. Orange to yellow colors indicate locations where Fill-Spill-Merge had more water, and purple to blue colors
 452 indicate locations where FlowFill had more water. Differences of less than 3 mm have been masked out. Differences are generally small,
 453 and are likely a result of the iterative nature of the FlowFill algorithm which causes it to asymptotically approach the correct values. In
 454 some locations, Fill-Spill-Merge retains slightly more water in depressions that FlowFill does. This could be due to water which has not yet
 455 finished moving downslope and into these depressions in the FlowFill algorithm. In other locations, FlowFill has retained more water. One
 456 possible reason for this is that some depressions have a narrow outlet, through which Fill-Spill-Merge is able to move all water as appropriate
 457 but the cell-by-cell movement of water with FlowFill can produce transient dams that reroute additional water towards these subcatchments.
 458 This DEM was prepared by Lai and Anders (2018) and Callaghan and Wickert (2019) from LiDAR topographic data provided by the Illinois
 459 State Geological Survey (Illinois Geospatial Data Clearinghouse, 2020).



483

484 **Figure 8.** The difference between results of Fill-Spill-Merge and FlowFill at the Río Toro site. The values for panels (a) to (e) indicate the
 485 depth of uniform runoff applied across the landscape for both algorithms. For example, in (a), each cell across the domain starts with 0.1
 486 m of surface water. Orange to yellow colors indicate locations where Fill-Spill-Merge had more water, and purple to blue colors indicate
 487 locations where FlowFill had more water. Differences of less than 3 mm have been masked out. In panel (e), 15 m of water was enough to fill
 488 all depressions with both algorithms, so there are no differences between the two. The most significant difference is seen in panel (c), where
 489 FlowFill retained additional water in a large depression. This is likely due to transient damming of its narrow inlet in FlowFill’s cell-by-cell
 490 method of moving water, which may have prevented the full volume of water from leaving the depression. This DEM was generated with
 491 data acquired from the TanDEM-X mission (Krieger et al., 2013; Rizzoli et al., 2017).

Runoff depth [m]	Sangamon			Río Toro		
	FlowFill [s]	FSM [s]	Speed-up [x]	FlowFill [s]	FSM [s]	Speed-Up [x]
15	642.65	0.243	2645	154.70	0.317	488
10	626.59	0.241	2600	124.37	0.309	402
5	570.02	0.241	2365	93.56	0.300	312
1	472.33	0.241	1960	53.09	0.316	168
0.2	508.87	0.235	2165	38.30	0.316	121
0.1	464.15	0.230	2018	35.75	0.301	119
0.05	418.71	0.243	1723	33.62	0.316	106
0.01	200.81	0.227	885	32.06	0.315	102
0.001	20.12	0.235	86	30.99	0.319	97

508 **Table 2. Time comparison of Fill-Spill-Merge vs FlowFill.** Wall-times are in seconds comparing FlowFill (Callaghan and Wickert, 2019)
509 parallelized across 24 cores versus Fill-Spill-Merge on a single core; “Speed-Up” is a multiplicative factor. Using FlowFill, wall-times
510 increased with the depth of applied runoff and on flatter landscapes. Using FSM, wall-time is independent of depth of applied runoff and
511 ruggedness of landscape, but increases for larger domains. FSM’s wall-times were 86–2,645 times faster than FlowFill for these examples;
512 compute times were 2,064–63,480 times faster.

495 both study sites, the differences between the two algorithms are minimized at the extreme high and extreme low starting runoff
496 values. For the highest runoff values, this is because both algorithms successfully fill all depressions in the landscape, so that
497 no differences are possible. For the lowest runoff values, both algorithms simulate only a small amount of water filling any
498 depression, so that that significant differences between the two algorithms are not possible. The biggest differences are therefore
499 seen for moderate starting runoff values, when depressions contain substantial water volumes, but are still only partially filled.
500 One possible cause for these discrepancies is FlowFill’s asymptotic approach to an equilibrium water level, which may prevent
501 small volumes of water from reaching the depression to which they belong. On the other hand, depressions with a narrow
502 outlet could be especially prone to being overfilled by FlowFill because its cell-by-cell algorithm could dynamically dam
503 this outlet, routing additional water into the depression. Both of these possibilities are further linked to the fact that FlowFill
504 dynamically evolves a land-plus-water flow-routing surface, whereas Fill-Spill-Merge routes flow just over the land surface.
505 These differences make FlowFill more useful for understanding temporal changes in surface water distribution, while Fill-
506 Spill-Merge provides a more accurate snapshot of surface hydrology under equilibrium conditions.

513 5 Conclusions

514 Here we leverage the depression hierarchy data structure (Barnes et al., 2020) to route flow through surface depressions in
515 a realistic, yet efficient, manner. In comparison to previous approaches, such as Jacobi iteration, the new algorithm runs in
516 log-linear time in the input size and is accompanied by extensively commented source code. This computationally efficient al-

517 gorithm may help us to better understand hydrologic connectivity and water storage across the land surface, and is an important
518 step forwards in recognising the importance of depressions as real-world features in digital elevation models.

519 *Code availability.* Complete, well-commented source code, an associated makefile, and correctness tests are available from [https://github.](https://github.com/r-barnes/Barnes2020-FillSpillMerge)
520 [com/r-barnes/Barnes2020-FillSpillMerge](https://github.com/r-barnes/Barnes2020-FillSpillMerge) and Zenodo (Barnes and Callaghan, 2020).

521 *Author contributions.* KC and AW conceived the problem. RB conceived the algorithm and developed initial implementations. KC and RB
522 completed, debugged and tested the algorithm. All authors contributed to the preparation of the manuscript.

523 *Competing interests.* The authors declare that they have no conflict of interest.

524 *Acknowledgements.* RB was supported by the Department of Energy's Computational Science Graduate Fellowship (Grant No. DE-FG02-
525 97ER25308) and, through the Berkeley Institute for Data Science's PhD Fellowship, by the Gordon and Betty Moore Foundation (Grant
526 GBMF3834) and by the Alfred P. Sloan Foundation (Grant 2013-10-27).

527 KLC was supported by the National Science Foundation under grant no. EAR-1903606, the University of Minnesota Department of Earth
528 Sciences Junior F Hayden Fellowship, the University of Minnesota Department of Earth Sciences H.E. Wright Footsteps Award, and start-up
529 funds awarded to AW by the University of Minnesota.

530 Empirical tests and results were performed on XSEDE's Comet supercomputer (Towns et al., 2014), which is supported by the National
531 Science Foundation (Grant No. ACI-1053575). Portability and debugging tests were performed on the Mesabi machine at the Minnesota
532 Supercomputing Institute (MSI) at the University of Minnesota (<http://www.msi.umn.edu>).

533 The Deutsches Zentrum für Luft- und Raumfahrt (DLR) provided 12 m TanDEM-X DEM coverage of the Río Toro catchment via proposal
534 DEM_GEOL1915 awarded to Taylor Schildgen, Andrew Wickert, Stefanie Tofelde, and Mitch D'Arcy. Jingtao Lai and Alison Anders
535 provided a copy of their Sangamon River DEM.

536 This collaboration resulted from a serendipitous meeting at the Community Surface Dynamics Modeling System (CSDMS) annual meet-
537 ing, which RB had attended on a CSDMS travel grant.

538 **6 Pseudocode**539 **6.1 MoveWaterInDepHier**

```

540 1: function MoveWaterInDepHier(root, DH, JumpTable)
541 2: Let root be the id of the depression we're currently con-
542 sidering
543 3: Let DH be a Depression Hierarchy
544 4: Let JumpTable be a hash table mapping DH labels to DH
545 labels
546 5:
547 6: ▷ For "children" of leaves
548 7: if root=NOVALUE then return
549 8:
550 9: ▷ The traversal
551 10: for each ocean-linked child c of root do
552 11:   Call MoveWaterInDepHier(c, DH, JumpTable)
553 12: end for
554 13: Call MoveWaterInDepHier(c.left_child, DH, JumpTable)
555 14: Call MoveWaterInDepHier(c.right_child, DH, JumpTable)
556
557 15:
558 16: if root=OCEAN then return
559 17:
560 18: if root has children and both their depression volumes
561 equal their water volumes and root's water volume is zero
562 then
563 19:   root.water_vol += root.left_child.water_vol
564 20:   root.water_vol += root.right_child.water_vol
565 21: end if
566 22:
567 23: if root.water_vol>root.dep_vol then
568 24:   Call OverflowInto(root, root.parent, DH, JumpTable, 0)
569 25: end if

```

570 **6.2 OverflowInto**

```

571 1: function OverflowInto(root, StopNode, DH, JumpTable,
572 ExtraWater)
573 -----
574 2: Let root be the id of the depression we're currently con-
575 sidering
576 3: Let StopNode be the id of the depression that ends the
577 traversal. It is the parent of the depression that first called
578 this function.
579 4: Let DH be a Depression Hierarchy
580 5: Let JumpTable be a hash table mapping DH labels to DH
581 labels
582 6: Let ExtraWater be the water that needs to be distributed
583 in DH
584 7:
585 8: ▷ If depression is too full, get its excess so we can find a
586 home for it
587 9: if root.water_vol>root.dep_vol then
588 10:   ExtraWater += root.water_vol - root.dep_vol
589 11:   root.water_vol = root.dep_vol
590 12: end if
591 13:
592 14: if root=StopNode or root=OCEAN then
593 15:   root.water_vol += ExtraWater
594 16: return root
595 17: end if
596 18:
597 19: ▷ 1st place to stash water: in this depression
598 20: if root.water_vol<root.dep_vol then
599 21:   Let C=root.dep_vol - root.water_vol
600 22:   if ExtraWater< C then
601 23:     root.water_vol = root.water_vol+ExtraWater
602 24:     ExtraWater = 0
603 25:   else
604 26:     root.water_vol = root.dep_vol

```

```

604 27:   ExtraWater -= C
605 28:   end if
606 29: end if
607 30:
608 31: if ExtraWater=0 then
609 32:   return root
610 33: end if
611 34:
612 35: if root∈JumpTable then
613 36:   return JumpTable(root) = OverflowInto(JumpTable(root),
614     StopNode, DH, JumpTable, ExtraWater)
615 37: end if
616 38:
617 39: ▷ 2nd place to stash water: in the depression's sibling
618 40: if root.sib≠NOVALUE then
619 41:   if root.sib.water_vol<root.sib.dep_vol then
620 42:     return JumpTable(root) = OverflowInto(root.geolink,
621       StopNode, DH, JumpTable, ExtraWater)
622 43:   else if root.sib.water_vol>root.sib.dep_vol then
623 44:     e=root.sib.water_vol-root.sib.dep_vol
624 45:     ExtraWater += e
625 46:     root.sib.water_vol = root.sib.dep_vol
626 47:   end if
627 48: end if
628 49:
629 50: ▷ 3rd place to stash water: in the depression's parent
630 51: if root.parent.water_vol=0 and root is not oceanlinked to
631   root.parent then
632 52:   root.parent.water_vol += root.water_vol
633 53:   if root.sib≠NOVALUE then
634 54:     root.parent.water_vol += root.sib.water_vol
635 55:   end if
636 56: end if
637 57: return JumpTable(root) = OverflowInto(root.parent, StopN-
638   ode, DH, JumpTable, ExtraWater)

```

639 6.3 FillDepressions

```

640 1: function FillDepressions(PitCell, OutCell, DepLabels, Wa-
641   terVol, dem, labels, wtd)


---


642 2: Let PitCell be the cell to start filling from
643 3: Let OutCell be the outlet/spill cell
644 4: Let DepLabels be the labels contained within the metade-
645   pression we are trying to fill
646 5: Let WaterVol be the amount of water that needs to be
647   spread throughout the depression
648 6: Let dem be the topography.
649 7: Let labels be the labels from the Depression Hierarchy
650 8: Let wtd be the depth of water in each cell.
651 9: Let visited be a hash set of cell ids
652 10: Let PQ be a priority queue sorted by increasing elevation
653 11: Let affected be a plain queue
654 12: Let  $T_e$  be the total elevation; initially 0
655 13:
656 14: if WaterVol=0 then return
657 15:
658 16: Place PitCell into PQ and mark it visited
659 17: while PQ is not empty do
660 18:   Let c=pop(PQ)
661 19:   Let  $V = |affected| \cdot c.elev - T_e$ 
662 20:
663 21:   if WaterVol< V then
664 22:      $W_L = (WaterVol + T_e)/|affected|$ 
665 23:     Set wtd for all cells in affected to  $W_L$ 
666 24:     return
667 25:   end if
668 26:
669 27:   if c ≠ OutCell then
670 28:     Place c into affected
671 29:      $T_e += c.elev$ 
672 30:   end if

```



```
673 31:  Add all of  $c$ 's neighbours that belong to depressions in
674       $DepLabels$  and are not the outlet cell to  $PQ$  and mark
675      them visited
676 32:  if  $PQ$  is empty then
677 33:      Add  $OutCell$  to  $PQ$  and mark it visited
678 34:  end if
679 35:  end while
```

680 **References**

- 681 Agenis-Nevers, M., Bokde, N. D., Yaseen, Z. M., and Shende, M.: GuessCompX: An empirical complexity estimation in R,
682 arXiv:1911.01420v1, 2019.
- 683 Arnold, N.: A new approach for dealing with depressions in digital elevation models when calculating flow accumulation values,
684 *Progress in Physical Geography: Earth and Environment*, 34, 781–809, <https://doi.org/10.1177/0309133310384542>, <https://doi.org/10.1177/0309133310384542>, 2010.
- 685 Barnes, R.: Parallel non-divergent flow accumulation for trillion cell digital elevation models on desktops or clusters, *Environmental Modelling & Software*, 92, 202–212, <https://doi.org/10.1016/j.envsoft.2017.02.022>, 2017.
- 686 Barnes, R.: r-barnes/richtem: Zenodo DOI Release, Software, <https://doi.org/10.5281/zenodo.1295618>, <https://doi.org/10.5281/zenodo.1295618>, 2018.
- 687 Barnes, R.: Accelerating a fluvial incision and landscape evolution model with parallelism, *Geomorphology*, 330, 28–39,
688 <https://doi.org/10.1016/j.geomorph.2019.01.002>, 2019.
- 689 Barnes, R. and Callaghan, K.: Depression Hierarchy Source Code, <https://doi.org/10.5281/zenodo.3238558>, 2019.
- 690 Barnes, R. and Callaghan, K.: Fill-Spill-Merge Source Code, <https://doi.org/10.5281/zenodo.3755142>, 2020.
- 691 Barnes, R., Lehman, C., and Mulla, D.: Priority-flood: An optimal depression-filling and watershed-labeling algorithm for digital elevation
692 models, *Computers & Geosciences*, 62, 117 – 127, <https://doi.org/10.1016/j.cageo.2013.04.024>, 2014.
- 693 Barnes, R., Callaghan, K., and Wickert, A.: Computing water flow through complex landscapes, Part 2: Finding hierarchies in depressions
694 and morphological segmentations, *Earth Surface Dynamics*, <https://doi.org/10.5194/esurf-2019-34>, 2020.
- 695 Beucher, S.: Watershed, Hierarchical Segmentation and Waterfall Algorithm, in: *Mathematical Morphology and Its Applications to
696 Image Processing*, edited by Viergever, M. A., Serra, J., and Soille, P., vol. 2, pp. 69–76, Springer Netherlands, Dordrecht,
697 https://doi.org/10.1007/978-94-011-1040-2_10, 1994.
- 698 Braun, J. and Willett, S. D.: A very efficient $O(n)$, implicit and parallel method to solve the stream power equation governing fluvial incision
699 and landscape evolution, *Geomorphology*, 180-181, 170–179, <https://doi.org/10.1016/j.geomorph.2012.10.008>, 2013.
- 700 Breckenridge, A. and Johnson, T. C.: Paleohydrology of the upper Laurentian Great Lakes from the late glacial to early Holocene, *Quaternary
701 Research*, 71, 397–408, <https://doi.org/10.1016/j.yqres.2009.01.003>, 2009.
- 702 Cabrol, N. A. and Grin, E. A.: Distribution, classification, and ages of Martian impact crater lakes, *Icarus*, 142, 160–172, 1999.
- 703 Callaghan, K. L. and Wickert, A. D.: Computing water flow through complex landscapes – Part 1: Incorporating depressions in flow routing
704 using FlowFill, *Earth Surface Dynamics*, 7, 737–753, <https://doi.org/10.5194/esurf-7-737-2019>, 2019.
- 705 Cordonnier, G., Bovy, B., and Braun, J.: A Versatile, Linear Complexity Algorithm for Flow Routing in Topographies with Depressions,
706 *Earth Surface Dynamics Discussions*, 2018, 1–18, <https://doi.org/10.5194/esurf-2018-81>, 2018.
- 707 Fan, Y., Li, H., and Miguez-Macho, G.: Global Patterns of Groundwater Table Depth, *Science*, 339, 940–943,
708 <https://doi.org/10.1126/science.1229881>, 2013.
- 709 Fenner, T. I. and Loizou, G.: Loop-free Algorithms for Traversing Binary Trees, *BIT*, 24, 33–44, <https://doi.org/10.1007/BF01934513>, 1984.
- 710 GDAL Development Team: GDAL - Geospatial Data Abstraction Library, Open Source Geospatial Foundation, available at <http://www.gdal.org>, 2016.
- 711 GEBCO: General Bathymetric Chart of the Oceans (GEBCO), GEBCO_08 grid, version 20100927, <http://www.gebco.net>, 2010.
- 712

716 Hilley, G. E. and Strecker, M. R.: Processes of oscillatory basin filling and excavation in a tectonically active orogen: Quebrada del Toro
717 Basin, NW Argentina, GSA Bulletin, 117, 887–901, <https://doi.org/10.1130/B25602.1>, 2005.

718 Illinois Geospatial Data Clearinghouse: Illinois Height Modernization (ILHMP), [https://clearinghouse.isgs.illinois.edu/data/elevation/
719 illinois-height-modernization-ilhmp-lidar-data](https://clearinghouse.isgs.illinois.edu/data/elevation/illinois-height-modernization-ilhmp-lidar-data), 2020.

720 Jenson, S. and Domingue, J.: Extracting Topographic Structure from Digital Elevation Data for Geographic Information System Analysis,
721 Photogrammetric Engineering and Remote Sensing, 54, 1–5, [https://doi.org/0099-1112/88/5411-1593\\$02.25/0](https://doi.org/0099-1112/88/5411-1593$02.25/0), 1988.

722 Krieger, G., Zink, M., Bachmann, M., Bräutigam, B., Schulze, D., Martone, M., Rizzoli, P., Steinbrecher, U., Antony, J. W., De Zan, F., et al.:
723 TanDEM-X: A radar interferometer with two formation-flying satellites, Acta Astronautica, 89, 83–98, 2013.

724 Lai, J. and Anders, A. M.: Modeled Postglacial Landscape Evolution at the Southern Margin of the Laurentide Ice Sheet: Hydrological
725 Connection of Uplands Controls the Pace and Style of Fluvial Network Expansion, Journal of Geophysical Research: Earth Surface, 123,
726 967–984, <https://doi.org/10.1029/2017JF004509>, 2018.

727 Li, S., MacMillan, R., Lobb, D. A., McConkey, B. G., Moulin, A., and Fraser, W. R.: Lidar DEM error analyses and topo-
728 graphic depression identification in a hummocky landscape in the prairie region of Canada, Geomorphology, 129, 263–275,
729 <https://doi.org/10.1016/j.geomorph.2011.02.020>, 2011.

730 Lindsay, J. and Creed, I.: Removal of artifact depressions from digital elevation models: towards a minimum impact approach, Hydrological
731 processes, 19, 3113–3126, <https://doi.org/10.1002/hyp.5835>, 2005a.

732 Lindsay, J. B.: Efficient hybrid breaching-filling sink removal methods for flow path enforcement in digital elevation models: Efficient Hybrid
733 Sink Removal Methods for Flow Path Enforcement, Hydrological Processes, <https://doi.org/10.1002/hyp.10648>, 2015.

734 Lindsay, J. B.: Efficient hybrid breaching-filling sink removal methods for flow path enforcement in digital elevation models: Efficient
735 Hybrid Sink Removal Methods for Flow Path Enforcement, Hydrological Processes, 30, 846–857, <https://doi.org/10.1002/hyp.10648>,
736 <http://doi.wiley.com/10.1002/hyp.10648>, 2016.

737 Lindsay, J. B. and Creed, I. F.: Removal of artifact depressions from digital elevation models: Towards a minimum impact approach, Hydro-
738 logical Processes, 19, 3113–3126, <https://doi.org/10.1002/hyp.5835>, 2005b.

739 Mark, D.: Modelling in Geomorphological Systems, chap. Network models in geomorphology, pp. 73–97, John Wiley & Sons, 1988.

740 Martz, L. W. and Garbrecht, J.: The treatment of flat areas and depressions in automated drainage analysis of raster digital elevation models,
741 Hydrological Processes, 12, 843–855, [https://doi.org/10.1002/\(SICI\)1099-1085\(199805\)12:6<843::AID-HYP658>3.0.CO;2-R](https://doi.org/10.1002/(SICI)1099-1085(199805)12:6<843::AID-HYP658>3.0.CO;2-R), 1998.

742 Martz, L. W. and Jong, E. d.: CATCH: A FORTRAN program for measuring catchment area from digital elevation models, Computers and
743 Geosciences, 14, 627–640, [https://doi.org/10.1016/0098-3004\(88\)90018-0](https://doi.org/10.1016/0098-3004(88)90018-0), 1988.

744 MNDNR - Minnesota Department of Natural Resources: Lake Bathymetric Outlines, Contours, Vegetation, and DEM, [https://gisdata.mn.
745 gov/dataset/water-lake-bathymetry](https://gisdata.mn.gov/dataset/water-lake-bathymetry), 2014.

746 MNGEO - Minnesota Geospatial Information Office: LiDAR Elevation Data for Minnesota, [http://www.mngeo.state.mn.us/chouse/elevation/
747 lidar.html](http://www.mngeo.state.mn.us/chouse/elevation/lidar.html), 2019.

748 Neteler, M., Bowman, M. H., Landa, M., and Metz, M.: GRASS GIS: A multi-purpose open source GIS, Environmental Modelling and
749 Software, 31, 124–130, <https://doi.org/10.1016/j.envsoft.2011.11.014>, <http://dx.doi.org/10.1016/j.envsoft.2011.11.014>, 2012.

750 O’Callaghan, J. and Mark, D.: The extraction of drainage networks from digital elevation data, Computer Vision, Graphics, and Image
751 Processing, 28, 323–344, [https://doi.org/10.1016/S0734-189X\(84\)80011-0](https://doi.org/10.1016/S0734-189X(84)80011-0), 1984.

752 Reheis, M.: Highest Pluvial-Lake Shorelines and Pleistocene Climate of the Western Great Basin, Quaternary Research, 52, 196–205,
753 <https://doi.org/10.1006/qres.1999.2064>, 1999.

754 Riddick, T., Brovkin, V., Hagemann, S., and Mikolajewicz, U.: Dynamic hydrological discharge modelling for coupled climate model
755 simulations of the last glacial cycle: the MPI-DynamicHD model version 3.0, *Geoscientific Model Development*, 11, 4291–4316,
756 <https://doi.org/10.5194/gmd-11-4291-2018>, 2018.

757 Rizzoli, P., Martone, M., Gonzalez, C., Wecklich, C., Tridon, D. B., Bräutigam, B., Bachmann, M., Schulze, D., Fritz, T., Huber, M., et al.:
758 Generation and performance assessment of the global TanDEM-X digital elevation model, *ISPRS Journal of Photogrammetry and Remote*
759 *Sensing*, 132, 119–139, 2017.

760 Salembier, P. and Pardas, M.: Hierarchical morphological segmentation for image sequence coding, *IEEE Transactions on Image Processing*,
761 3, 639–651, <https://doi.org/10.1109/83.334980>, 1994.

762 Schwanghart, W. and Scherler, D.: Bumps in river profiles: uncertainty assessment and smoothing using quantile regression techniques, *Earth*
763 *Surface Dynamics*, 5, 821–839, <https://doi.org/10.5194/esurf-5-821-2017>, 2017.

764 Sedgewick, R. and Wayne, K.: *Algorithms*, Addison-Wesley, 4 edn., 2011.

765 Skiena, S. S.: *The Algorithm Design Manual*, Springer, 2008.

766 Soille, P.: Optimal removal of spurious pits in grid digital elevation models, *Water Resources Research*, 40, 1–9,
767 <https://doi.org/10.1029/2004WR003060>, 2004.

768 Soille, P., Vogt, J., and Colombo, R.: Carving and adaptive drainage enforcement of grid digital elevation models, *Water Resources Research*,
769 39, 1366, <https://doi.org/10.1029/2002WR001879>, 2003.

770 Towns, J., Cockerill, T., Dahan, M., Foster, I., Gaither, K., Grimshaw, A., Hazlewood, V., Lathrop, S., Lifka, D., Peterson, G. D., et al.:
771 XSEDE: accelerating scientific discovery, *Computing in Science & Engineering*, 16, 62–74, 2014.

772 Wallis, C., Watson, D., Tarboton, D., and Wallace, R.: Parallel flow-direction and contributing area calculation for hydrology analysis in
773 digital elevation models, in: *Proceedings of the 2009 International Conference on Parallel and Distributed Processing Techniques and*
774 *Applications*, Las Vegas, Nevada, <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.158.2864>, 2009.

775 Wickert, A. D.: Reconstruction of North American drainage basins and river discharge since the Last Glacial Maximum, *Earth Surface*
776 *Dynamics*, 4, 831–869, <https://doi.org/10.5194/esurf-4-831-2016>, 2016.

777 Wu, Q. and Lane, C. R.: Delineation and quantification of wetland depressions in the Prairie Pothole Region of North Dakota, *Wetlands*, 36,
778 215–227, 2016.

779 Wu, Q., Liu, H., Wang, S., Yu, B., Beck, R., and Hinkel, K.: A localized contour tree method for deriving geometric and topological properties
780 of complex surface depressions based on high-resolution topographical data, *International Journal of Geographical Information Science*,
781 29, 2041–2060, <https://doi.org/10.1080/13658816.2015.1038719>, <https://doi.org/10.1080/13658816.2015.1038719>, 2015.

782 Wu, Q., Lane, C. R., Wang, L., Vanderhoof, M. K., Christensen, J. R., and Liu, H.: Efficient Delineation of Nested Depression Hierarchy in
783 Digital Elevation Models for Hydrological Analysis Using Level-Set Method, *Journal of the American Water Resources Association*, 0,
784 <https://doi.org/10.1111/1752-1688.12689>, 2018.