Earth Surf. Dynam. Discuss.,
https://doi.org/10.5194/esurf-2018-81-AC2, 2019


Earth Surface Dynamics Discussions
Open Access
EGU



# *Interactive comment on* "A Versatile, Linear Complexity Algorithm for Flow Routing in Topographies with Depressions" *by* Guillaume Cordonnier et al.

**Guillaume Cordonnier et al.**

gllme.cordonnier@gmail.com

Received and published: 8 March 2019

We first want to thank the referee for all these comments that will certainly help to greatly improve the paper. We will respond to all of them in the text, and do our best to clarify or correct all the points you mentioned.

Here is a more detailed answer to the specific comments.

———— Major issues: ————

3) The only major flaw with this paper is that the benchmarks seem ill-constructed. Figure 5a is nice, though I think it would be improved by showing Zhou (2017). (If



Printer-friendly version

Discussion paper

I recall correctly, RichDEM also has an implementation of this, so it should be easy to do.) I think it would also be improved by considering larger datasets, perhaps up to 10,000x10,000. I think Figure 5c needs some real work. Wei (2018) should be included here, no doubt. Figure 5a shows that it has completely different performance characteristics than Barnes (2014a), so this is critical to contextualizing your work. Surely modifying the algorithm to include surface offsets or flow directions isn't too difficult? Otherwise, perhaps Wei et al. or Barnes et al. have such an implementation? Please find a way to address this.

Answer: Maybe the referee means Zhou (2016)? Zhou (2017) is about a parallel version of the algorithm, and it would be irrelevant to add it in the benchmarks. On the other hand, Zhou (2016) could definitively be added, although all our benchmarks agree that Wei (2018) performs better, although with a similar pattern. We already have results of benchmarks for both Zhou (2016) and larger datasets (up to 16k x 16k), so it would not be difficult to add them in the paper if needed. Obviously, as pointed by the referee, all these implementations do not include flow direction. Modifying the code to include surface offset or flow direction would represent additional work (design/implementation/testing) that could be published separately, whereas in this paper we preferred to compare our method with state-of-the-art algorithms that have been already published and for which one can find an existing implementation. The suggested modification is, in our opinion, not yet state-of-the-art.

4) Similarly, as discussed below, the Wei, Barnes, and Zhou algorithms are all adapted to large datasets and, when I've looked at the source, written in a very literate style (minimal bit-hacking). This raises the question of whether versions of these algorithms optimized for use on smaller datasets would perform better in your benchmark. I think you've done a good job of showing that your algorithm ranks with the state of the art, but the comparison isn't strong enough to conclude that the design itself is superior. You may or may not consider this worth addressing.

Answer: We will elaborate on the details below. As stated before, we now have results

for larger grids so we can easily propose a fairer comparison. It should be noted that our goal is more to explore theoretical results (demonstrating the time linearity of our algorithm) than to propose a competitive, optimal implementation.

5) Please make the source code available online. It seems to only be available as a pull request at the moment and this is not suitable for publication. Ideally, the source code and associated tests would be available on Github and archived on Zenodo or Figshare.

Answer: We share the same concerns than the referee about code availability. Although a pull request on GitHub already refers to a git branch that is published online on this platform, we agree that making the code available in a separate repository may be better in order to reproduce the results presented in this paper. However, we are concerned by having to maintain duplicate codes in different repositories. Maintaining individual repositories for every algorithm that we want to include in the fastscapelib library is not a good solution either. This might actually have a negative impact on reproducibility and readability. Therefore, we propose to create a repository specifically for this paper; once the paper is accepted we will archive it (i.e., making it read-only) and redirect users to the fastscapelib library where further development and maintenance will happen.

————— Detailed answers: —————

Q: p. 3, Line 22: "zk>zn, and rcv(k)=n". Could it be that "zk>zn and not rcv(k)=n" or "not zk>zn and rcv(k)=n"? That is, are both conditions necessary, or does one imply the other? You should clarify this.

A: Both conditions are necessary: a node is always the receiver of all his donors, and is always strictly bellow them.

Q: p. 3, Line 25: Lindsay (2016) also defines a "flat-bottomed depression". In such a depression, no cell is a proper local minimum. Presumably such cells would be labeled

as singular nodes due to the manner in which you construct the rcv array, though you don't make this special case explicit in your paper. You should clarify this.

A: True, "flat bottomed depressions" are singular nodes in our terminology. This will be clarified.

Q: This is the inevitable result of using convergent/unidirectional flow routing.

A: p. 4, Pt 1: This is true for unicity, not for existence. The path ends at a base level node (or boundary node), which is in general not the case due to the existence of local minima.

Q: p. 4, Pt 3: I have difficulty following this point. A minimum can only be defined with respect to a set of possible alternatives and it is not clear at this point what the set of alternatives is. Since you haven't introduced it previously, I would appreciate a forward reference to the point in your paper where you do introduce it.

A: the minimum is chosen among all possible flow paths starting at a given point. This will be rephrased. Another issue with this equation is that this does not hold for nodes below water lever. We will rewrite this condition to properly take this point into account.

Q: p. 4, Line 8: "whether the singular node is a base level node". I feel like the term "base level node" is unintuitive, since many nodes could be at, or below, a given base level, but be interior to a landscape. The term "boundary node" seems more intuitive.

A: We have mixed feelings with this point: "boundary" is clearly more intuitive in the sense of boundary condition, but not in the geometrical meaning. Any boundary condition could easily be chosen inside the domain. Nevertheless, we will do the change.

Q: p. 4, Line 9: "that all have an elevation below a given water level". I find this definition counterintuitive. A basin includes regions outside of a depression (up to the peaks of mountains). We are now calling a depression the set of cells below a chosen waterlevel, but nothing about this paragraph constrains a water level to be contained with a Lindsay (2016) typology-style depression. It seems possible to set the spill as

a mountain peak and declare the entire DEM a depression. I wonder if there's a more intuitive term that can be used here? Or perhaps a constraint can be added?

A: Linking this definition with Lindsay terminology will certainly make it clearer.

Q: p. 4, Lines 19-30: I think these steps are performed iteratively, but don't see mention of this. Could you please clarify?

A: No, these steps are needed only once to find and resolve all local minima. But each of them include inner iterations over the domain. We will clarify this.

Q: p. 5, Line 3: "visiting the donors recursively". I followed the reference to the Appendix, but did not find it to be enlightening since it requires knowledge of the stack construction from Braun and Willet (2013). I think you are doing a depth-first traversal of all of the cells in a basin (those cells whose flow ultimately terminates at a given singular node). If so, stating this directly would make things clearer. The word recursion also implies a particular computational motifâĂŤfunctions calling themselvesâĂŤwhich isn't implied by the Appendix. If you are using recursion, the Appendix should be modified. However, this is a risky strategy for larger datasets since some operating systems impose relatively low limits on the depth of recursion.

A: We can copy Braun and Willet (2013) algorithm in the appendix if needed (which is indeed a depth first traversal). We will change recursion to traversal which is technically more correct.

Q: Algorithm 6: "if elevation of the pass of Links(link) < zpass then" finds the highest link between two basins. I think this is the opposite of what you want.

A: No, zpass is the lowest altitude found so far, so we update it only if the candidate is better.

Q: p. 8, Line 1: "rather unlikely". This approach to correctness seems a bit cavalier. I expect that a good algorithm works in all cases, not merely when it's convenient. The rest of this paragraph seems to imply that, in theory, this problem can be handled. But

I'm left with the feeling that your implementation may contain subtle bugs that you've thought are too unlikely to pay attention to! I would like assurance that you check for such exceptional situations and either handle them gracefully and correctly, or flag them for user intervention. It's also unclear why the assumption you state solves the problem and why it isn't possible to build adversarially nested structures.

A: There are several aspects here. First, in 8-connected meshes, the graph may no longer be planar but the crossing edges are in general in low proportion in the overall graph, hence the "rather unlikely" which, indeed, needs more rigorous explanations. By changing 8 to 16 in Mares (2002) algorithm, we validated experimentally that the mathematical condition for the O(n) complexity holds for any (tested) 8-connected graphs. We unfortunately do not have a formal proof for that, but this has no impact on the correctness of the algorithm, only on the complexity. This means that 1) our algorithm is proven to be O(n) for any planar input mesh (e.g. 4 connected or triangulation), and 2) it is not proven that there is no very small chance that for some very rare cases, the complexity might be O (n log n) for 8-connected graph. Once again, we never saw this case occurring.

Q: p. 8, 2.3.1: Are all of the depressions carved to make a continuous flow network? If so, the situation shown in Figure 3 doesn't seem as though it can happen: Flow will progress along the carved network before water ever pools. Similarly, if depression filling is applied, pooling won't occur. And: Figure 3c: The depiction of standing water shown here is at odds with my understanding of what the flow network looks like following carving or depression-filling. Or perhaps erosion fills in the channels connecting depressions, resulting in standing water?

A: Depression carving and filling are metaphors: our algorithm updates the flow directions, but does not change the elevation of the DEM. We name our variants "filling" and "carving" to match with previous work, but in our case, we compute the flow directions that would appear if the topography has been filled or carved. Pooling can be computed quite easily as a post process on the filling strategy by parsing the flow network

from top to bottom (which can obviously not work with carving), and we show it to give an idea of the water level, and hence of the location of the spills on fig 3, and latter to show the effect of the different strategies on erosion in fig 4.

Q: Algorithm 4: I struggled somewhat to follow the logic here. Why is a cost function used instead of a simple breadth-first ordering? Perhaps a figure showing what the different flow paths look like would be helpful. The algorithm also seems as though it will terminate at the depression's pit cell. However, the flow pattern to the outlet will then have half of the depression exhibiting convergent flow to the pit cell and half of the depression showing convergent flow to the outlet. This flow pattern is unrealistic if the entire depression is treated as having been filled with flow superimposed on the resultant flat surface.

A: There are several issues with a filling led by a breadth first traversal, the first one being that it depends on the order of that parse. Local methods generally suffer from straight patterns resulting from the 4 or 8 connectivity, which is why we propose a better approximation with a cost function associated to a simple Euclidean distance. This is not a major contribution of our work, and any other solution could be explored here. There is indeed a mistake in the notation. It should be: for all neighbors n_nb such as z_n_in >= z_n_nb.

Q: p. 11: If only the receiver array is modified, how is the Equation From Page 9 implemented when water flows uphill? Campforts and Grovers (2015, "Keeping the Edge") show that the Stream Power Law as it's used in Fastscape doesn't preserve points or other topography discontinuities well. The channels carved here seem as though they are representative of the sorts of terrain this critique applies to. Some discussion of modeling philosophy might be useful.

A: We will add a reference to Campforts and Grovers (2015) to discuss on the limitation of this strategy. We simply follow the fastscape implementation, which numerically holds even for uphill flows, although this has no physical meaning. Lake deposition

or any other processes could be applied in this case but are outside the scope of this work.

Q: Figure 5: What I find most interesting about this figure is that Wei 2018 is competitive with your results in (a), but gets dropped in (c). Why, in (c), do you choose to compare your work against a slower algorithm? And: Why does Figure 5c uses Barnes (2014a)? I can't find an explanation anywhere. It seems like comparing against Barnes (2014a) is a poor choice. Barnes (2014a) suggests a path for accelerating Priority-Flood by reducing the number of cells passed through the Priority-Queue. Barnes (2014a), Zhou (2016), and Wei (2018) all present progressively better algorithms for doing this. It's better to think of all of these algorithms as being O(N + M log M), where M is the number of cells which pass through the priority-queue. M decreases dramatically between Barnes (2014a) and Wei (2018), as shown in Wei (2018, Figure 8). By the time Wei (2018), the O(N) term might even dominate the O(M log M) term, meaning that Wei (2018) is a linear-time algorithm! Indeed, your Figure 5a suggests this. In any case, Figure 5c seems to overstate your results since it's not a proper performance comparison.

A: This point is very interesting. One major difference between our work and previous work is that we apply our algorithm on landscape evolution models, rather than as a filling processing step on real, acquired data. As such, we work on synthetic data where the number of local minima can be arbitrary important. One of our finding here is that there is a critical proportion of local minima where the original implementation (Barnes 2014) outperforms any subsequent work. An intuition of the reason of this effect can be given by changing the reviewer notation to N: the number of nodes outside of depressions and M: the number of nodes inside the depressions. We try to give a better analysis in the paper, but a very simplified intuition could be that Barnes (2014)'s complexity "looks like" O(M + N log N), while subsequent work "looks like" O (N + M log M). There are obviously some constraints on M and N that makes the analysis more complex, but the simplicity of Barnes algorithm makes it O(N) part ex-

tremely efficient, which is shown on fig 5.a. This raises two important questions: are our results globally better than more recent work, and in which cases our algorithm (and recent work), performs better than Barnes (2014) ; a study that seems to lack in Zhou or Wei's papers. This is the point of figure 5. b and c, that shows that in a typical fastscape model, our algorithm is in condition to perform better (in term of number of local minima), than Barnes'. The first question (whether our algorithm performs better than previous work, being Zhou 2017 or Wei 2018), is partially answered in fig 5.a. We choose to add more focus on this behavior analysis than a detailed performance comparison, because it gives the reader more hints on when to choose a method or another, and a performance evaluation, as noted by the referee, largely depends on a variety of external aspects such as optimization or adaptation of the algorithm to the computer architecture.

Q: p. 12 and 13: Given this description, I don't see how your benchmarks will be meaningful. Using the Wei (2018) algorithm in conjunction with the Barnes (2014b) algorithm for the benchmarks of Figure 5a makes a certain sense, but why not simply modify the Wei (2018) algorithm using the PF+e strategy of Barnes (2014a)? Wouldn't this comparison be more fair?

A: I double checked the code, we do not include the flat resolution algorithm on the benchmark. We add, however, a post process connectivity computation (receivers, donors and stack), which is, however, an important output of our algorithm. This is performed only for the purpose of comparing performance, because the connectivity computation on flat surfaces does not cancel local minima.

Q: p. 13, Line 8: "we reuse the implementations available in the RichDEM" This seems reasonable, but it should be done with caveats. As you note on p. 2, this set of algorithms has "been optimized so that they can deliver acceptable performance when used with large datasets." It seems as though your algorithm uses significantly more space than the B, Z, and W algorithms you cite in your paper. The Barnes (2014a) algorithms work in-place using only the priority-queue and a boolean array as extra

memory; that is, it requires about O(sqrt(N)) additional space. In comparison, your code has a Receivers array, Donors array (which takes O(8N) space), and Union-Find array, leading to a O(10N) space requirement. Trading space for time is a well-known optimization technique. This raises the question of how the B,Z,W algorithms would perform had they been optimized for small datasets. It also raises the question of how your algorithm would perform on larger datasets. The average size of the test dataset used by Wei (2018), for instance, was 2*10ËĘ8 cells - two orders of magnitude larger than the 1500x1500 grid you use and three orders of magnitude larger than the Figure 5a benchmark. I don't think additional tests are necessary, but it should be clarified somewhere that the algorithms you're comparing against are being used for dataset sizes they weren't designed for.

A: We do not agree with this point. The donor & receiver data structures are still needed for many landscape evolution models using upwind numerical schemes), independently of the algorithm used, and the lack of optimization in this regard (donors array could be a N sized array) is more in favor of B, Z, W algorithms than ours. The union find and graph data structure are needed only for the depression graph, which size is generally much smaller than the grid size. Furthermore, we use grid size in the order of (or slightly larger than) what is generally used in landscape evolution modeling. We tried to benchmark larger data, up to 16k x 16k, which shows consistent results, but with more noise due to cache misses and other OS/material related issues that makes the results less readable.

Q: p. 14, Lines 10-15: This is a really nice analysis. However, I'm not sure I agree with it entirely. This line: "Because those variants are more complex, k4 and k5 have higher values." Why is the cost of using a priority-queue higher for the Zhou and Wei algorithms? I would think that k2=k5. I agree that k4>k1. However, one of the reasons priority queues are expensive is because they have poor cache coherence. (Search online for "Latency numbers every computer programmer should know". See also, Luengo-Hendriks (2010)'s discussion of priority queue performance variation.) This

means I expect k1

A: Yes, there is numbering issue, k3 > k0 and k4>k1. Cache coherency is an inherent problem of these depression filling algorithms, because of the random location of the depression on the topography, and which is hard to include in the discussion. Furthermore, the ratio of cache size to the grid size depends on the hardware and might change drastically in the next few years.

We left some comments unanswered because they are easy to solve, we will look carefully at each of them in the final version. We thank again the referee for all these comments and discussions.

Best regards,

The authors.

---

Interactive comment on Earth Surf. Dynam. Discuss., https://doi.org/10.5194/esurf-2018-81, 2018.



Printer-friendly version

Discussion paper