Jun 27th, 3:40 PM - 5:00 PM

Creation of a data model to retrieve constrained watershed boundaries.

Scott Haag
Drexel University, smh362@drexel.edu

Ali Shokoufandeh
Drexel University, as79@drexel.edu

Austin Gentry
Drexel University, acg59@drexel.edu

9th International Congress on Environmental Modelling and Software
Fort Collins, Colorado, USA, Mazdak Arabi, Olaf David, Jack Carlson, Daniel P. Ames (Eds.)
https://scholarsarchive.byu.edu/iemssconference/2018/

Creation of a Data Model to Retrieve Constrained Watershed Boundaries

Scott M. Haag[a,b], **Austin Gentry**[b], **and Ali Shokoufandeh**[b]
[a]*Academy of Natural Sciences of Drexel University (smh362@drexel.edu),* [b]*Department of Computer Science, Drexel University (acg59@drexel.edu, as79@drexel.edu)*

Abstract: When using flow direction grids to describe impacts to fluvial systems (streams, rivers, lakes and oceans) two spatial scales of interest are commonly applied, the watershed or the local stream reach (14). In this paper, we consider an alternative approach we will refer to as the constrained watershed boundary (CWB), defined as a polygon containing all the flow direction grid cells with a surface flow distance less than a user prescribed threshold. The proposed algorithm builds upon the HSM algorithm proposed by Haag and Shokoufandeh in 2017 (8), and augments the data structure with a flow distance grid calculated directly from the original flow direction grid. The only parameter that controls the rapid retrieval and visualization of CWB is the user defined distance threshold(s). The proposed algorithm is a variant of the HSM algorithm and therefore it will retrieve watershed boundaries more efficiently than competing grid searching techniques. Empirical tests for the Delaware River Watershed Retrieval problem indicate a reduction from 35 million read operations to 45 thousand using the HSM approach Haag and Shokoufandeh 2017. We have implemented the CHSM algorithm based on a restful-API architecture for the Chesapeake Bay Watershed using the 30 m flow direction grid from the NHDPlus v2 (10). Our results show similar speed increases for the CHSM as the original HSM algorithm with reductions of query complexity of between 99.94 and 99.59% compared to existing watershed retrieval algorithms. This platform can be augmented to support any hydro enforced D8 flow direction grid such as Hydrosheds. The techniques we applied to create the HSM and CHSM algorithms can be applied to any contiguous surface extraction from a regular grid.

Keywords: Flow Direction Grid (FDG); Haag- Shokoufandeh March (HSM); Watershed Retrieval; Constrained Watershed, Hydrological Modelling

1 Introduction.

In this paper we describe a general model for retrieving a constrained watershed boundaries for any location (grid cell or pour point) within a flow direction grid (FDG). Watershed modelling is an important computational problem that allows researchers to relate upflow parameters to conditions within fluvial systems (streams, rivers, lakes, and oceans). Tesfa et al.(17) consider watersheds retrieval as the "basic modelling element" for numerous hydrological problems. Delineation and retrieval of watershed boundaries are also utilized in a number of disciplines including engineering, urban infrastructure (1), ecological (13; 4), and biological (4; 18) among many others.

There are two primary scales that have been traditionally used to relate upstream conditions (e.g., Land Cover, Point Sources, terrestrial conditions and others) to fluvial water quality variables within Geographic Information Systems. The most general one is at the catchment or watershed scale that delineates the continuous surface where precipitation will eventually funnel through a point. This is normally done by delineating watersheds (retrieving) from digital elevation models (DEMs) by converting them into flow direction grids (D8 (11), D-Infity (16)) and others. In a secondary spatial approach, the stream reach is used

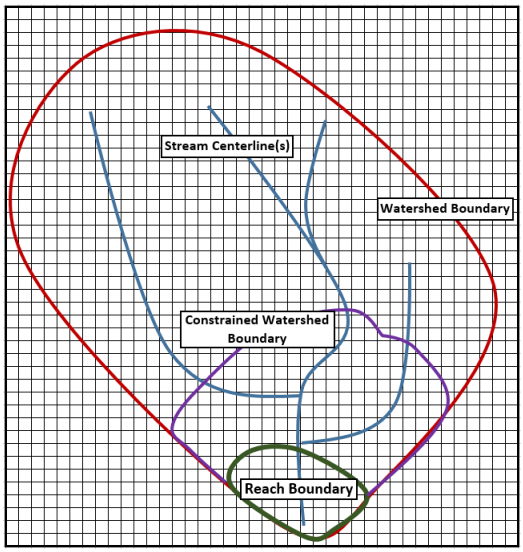

Figure 1: Example of the reach scale, watershed scale, and the constrained watershed boundaries.

to quantify a more local connection to fluvial impacts. Stream reaches and their associated local catchments are defined as segments of streams that share similar characteristics (soils, slope, vegetation), and the local catchments is defined as the polygonal area that delineates non-fluvial or non-channelized flow into the reach. The National Hydrography Data-set Version 2 uses the local stream reaches (catchments) and the entire watershed to help users compare in stream measurements to landscape parameters. The EPA stream catchment database provides precomputed attributes for 2.6 million at the stream reach and watershed scale (9). Our proposed method provides two advantages over these traditional spatial scales of analysis. First, it allows the user to quickly extract flow boundaries or constrained watershed boundaries using any flow distance from a user defined pour point. Second, it provides the ability to build subsets of the stream catchments, enabling water flow based segmentation similar to a GIS based buffer analysis using multiple buffer distances. Taken together, these allow users to efficiently and accurately extract a constrained watershed for any input location within a valid hydro-enforced $D8$ FDG (for an example see Figure 1).

The constrained watershed model described in this paper is a variation of the Haag Shokoufandeh Marching algorithm HSM. A general watershed retrieval algorithm that returns watershed boundaries in linear times, $O(n)$, in terms of number of vertices, n, found on the boundary of the watershed. The HSM (8) algorithm has been shown to be quadratically faster in comparison to existing watershed retrieval algorithms available in general purpose hydrologically tools (e.g. Gage Watershed from Taudem (16) or ESRI's watershed function (6)). Baker et al. (2) describes the general approach to retrieving a watershed boundary using a flow direction grid as 1) Building a Digital Elevation Model, 2) Filling sinks, 3) Creating a flow direction grid. Traditional watershed delineation algorithms (e.g. Gage Watershed from Taudem (16) or ESRI's watershed function (6)) traverse the flow direction grid identifying grid cells that flow through the submitted pour point. After all of the grid cells are identified, the output watershed boundary is retrieved by running a grid to vector operation resulting in a array of coordinates pairs (vector nodes) that represent the output polygon.

Haag and Shokoufandeh (7) proposed a new algorithm called Haag-Shokoufandeh March (HSM), which utilizes an additional data structure, known as the modified nested set (MNS), to efficiently retrieve watershed boundaries from flow direction grids. Most marching algorithms separate surfaces into inside and outside components or region of interest (15). Similarly, the HSM algorithm, incrementally constructs the watershed boundary by first finding a vertex that belongs to the boundary and advances the watershed construction by marching forward around the outside, never entering the interior of the watershed nor leaving the boundary. This technique has two major advantages over existing techniques that in turn will reduce the overall computational complexity of the watershed retrieval. First, the HSM algorithm has linear complexity, i.e., it executes at the same rate as the boundary length of the final watershed. In contrast, existing

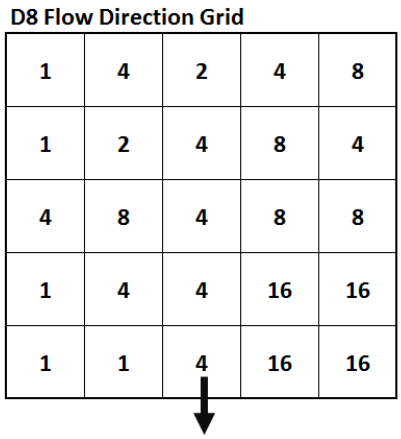

Figure 2: Example $D8$ flow direction grid (From Haag Shokoufandeh 2017 (8).

watershed delineation algorithms have a quadratic complexity, i.e, their execution time is proportional to the area of the watershed. The second advantage is that the marching algorithm extracts the watershed boundary as an array of coordinate pairs natively, and therefore it is not necessary to run a secondary raster to vector conversion.

2 Related Work.

Applying the constrained watershed variant of the HSM algorithm requires the input data-set in the form of a hydrologicaly corrected D8 flow direction grid. A corrected D8 FDG needs to have its sinks filled and stream center-lines burned through dam like structures (breached) to ensure a topological drainage surface where all water precipitating on land eventually flows to a large body of water (e.g., sea, ocean, and or lake). These processing conditions can cause issues in some physical environments such as karst or anthropomorphic underground drainage networks (e.g., drainage pipes, stormwater basins). Using a sink filled and burned-in hydrological channels is a common practice in the watershed retrieval and modelling domain (12). In earlier work, we described the similarities between graph data-structures used in the computational sciences and the D8 flow direction grid. Namely, D8 grids, when sink filled and hydro enforced, are specific types of graphs know as Directed Acyclic Graphs (DAGs). Further, D8 FDGs that do not allow divergent flows will result in combinatorial tree structures, where each grid cell location can be abstracted as a vertex and each value in the D8 grid will correspond to an edge in the resultant graph tree.

Before describing the variation of the HSM algorithm (hereafter the constrained HSM or CHSM) in detail, we discuss the common notation and structures used throughout the rest of the manuscript. These notations are the the same as the original notation used in the HSM description by Haag and Shokoufandeh 2017 (8). We use the natural numbers $\{0, 1, 2, 3\}$ for storing cardinal directions North, East, South, and West, respectively. We apply the CHSM algorithm to delineate watershed boundaries for any surface using a $D8$ flow direction grid (11) as input. Within a $D8$ structure every grid cell contains a value in $\{2^0, 2^1, ..., 2^7\}$ that denotes connectivity to one of its eight neighboring grid cells (11). Additionally, since each node in $v \in V$ corresponds to a regular grid cell, we can describe its location using index pair $\rho(v) = (x(v), y(v))$. Lastly, we added one new attribute for all v in G, the distance from the root as $t(d)$ and a constant scalar value σ which denotes the maximum distance that water can travel to be considered within the watershed.

The Modified Nested Set (MNS) structure, as described our early work (7), (8), is applied to the the D8 grid (Figure 2) resulting in a discovery and finish parameters for every grid-cell (Figure 3). The nested set algorithm was first described by (3) and was modified for the hydrological domain by (8) and (5). The discovery value is a unique integer label for each node v in the D8 grid, and it is found using a modified depth first search (DFS) traversal from the root of the tree G associated with D8 FDG. The finish value of a node v is the largest discovery value of any node above v during the DFS conducted by the MNS algorithm. Figure 3 represents the discovery (left panel) and finish times (right panel) for the vertices of the D8 grid in Figure 2, respectively. In this manuscript, we describe a special case of HSM algorithm that supports restrictions including the distance from root and one additional Boolean condition that will allow

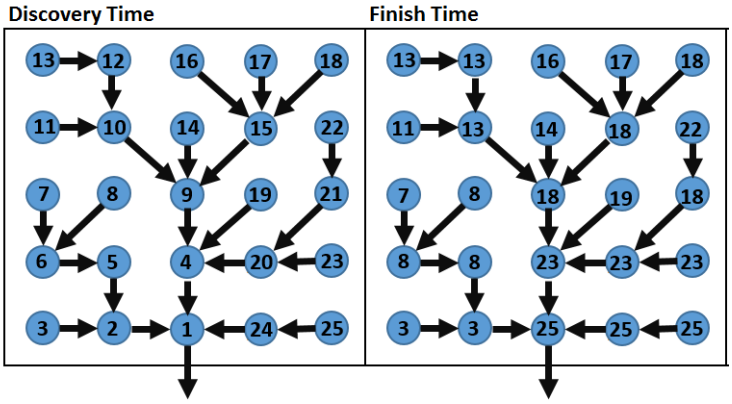

Figure 3: Example modified nested set labeling of flow direction grid cell (from Haag Shokoufandeh (8)), left panel shows unique discovery value, right panel shows finish time.

the retrieval of a constrained watershed. Figure 4 shows a third labelling value necessary to retrieve the constrained watershed boundary, the distance from the root. We will assume the existence of an additional labelling $t(v)$ for every node v. These labels are distance attributes that can be computed using a depth first traversal, where each step deeper into the graph (away from the root) adds 1 to the depth, and every step towards the root reduces the distance by one (to accurately measure distance we need to differentiate between orthogonal ($D8$ = 1, 4, 16 and, 64) and diagonal ($D8$ = 2, 8, 32, 128) moves, but for this paper we show only distance from the root in the DFS.

3 Method.

The HSM algorithm returns a watershed boundary by accepting as input a grid cell v^* and a $D8$ graph G where its vertices are labeled with discovery- and finish-time. Using this information, the HSM algorithm marches around the watershed identifying valid edges and returning to the original starting location v^*. For a complete description of this process referring to our original work (7). Intuitively, at each step of the march the algorithm will identify a valid march direction based on a Boolean inclusion condition, where a vertex is deemed as exclusively on the boundary of the watershed boundary if and only if its discovery value is between the range of the original pour points discovery and finish value. It does this by repeatedly calling the BoundaryPoint function which returns a Boolean value for the two grid cells in front of the current march location. Based on the 4 possible combinations of the BNP function calls and without a loss of generality assuming the previous march step was towards the north if both BNP are True then the march will move to the east, if both are false towards the west, if the right hand cell is False and the left hand cell

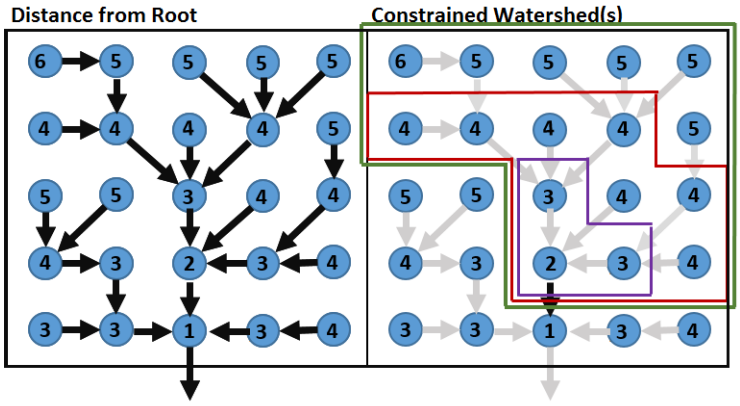

Figure 4: Left distance from root. right creation of three constrained watershed boundaries given Discovery Time of 2, and constrained distance value of (2 in purple, 3 in red, and 5 in green).

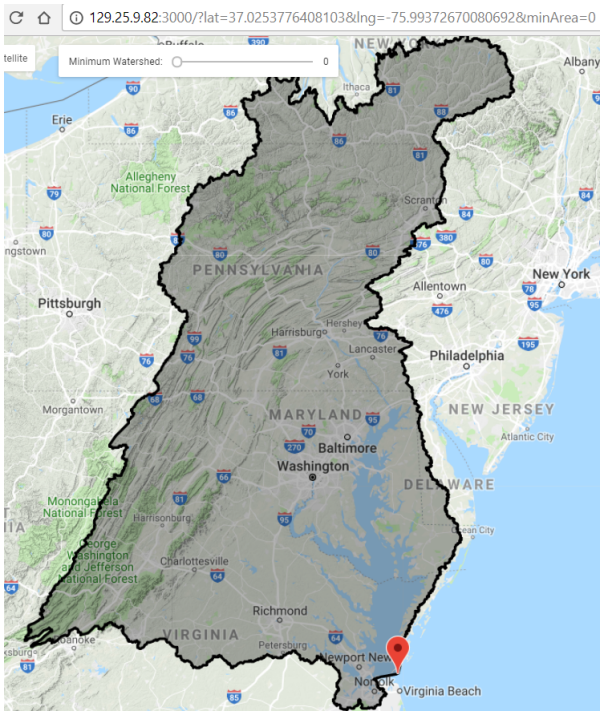

Figure 5: Overview map showing the endpoint for the HSM algorithm for the Chesapeake Bay Watershed (10) 30 m FDG. The background image is the Google Map API accessed in March of 2018.

is True then towards the north, and finally if the right hand cell is True and the left hand cell is False then the march can either move to the west or to the east, please see (8) for a complete description of these cases and corresponding graphics. We have shown that this is a necessary and sufficient condition that can be examined in constant time in every step of the algorithm by asking at most three Boolean questions of the MNS labelled grid. Figure 5 is example output from the endpoint of the HSM algorithm for the Chesapeake Bay Watershed (10) 30 m FDG. See Appendix for a overview of HSM Algorithm.

The incremental marching algorithm of Haag and Shokoufandeh (8) at each iteration extends the current watershed boundary of a pour point v^* by identifying the next candidate grid cell it. Specifically, it utilizes a Boolean function, BoundaryPoint, to examine the whether a given grid cell v^i is a within the watershed boundary or not. The function makes use of a structural property for any cell point with respect to discovery and finish-time of pour point v^*. Specifically, the necessary condition for point v^i being part of watershed of v^* is that its discovery time $d(v^i)$ must belong to the interval $[d(v^*), f(v^*)]$. that are within the watershed for a pour point v^*. It is this latter condition that BoundaryPoint will examine and depending on the outcome will return a value of True or False.

The modified version of BoundaryPoint function will enforce an additional constraint while examining the necessary condition for vertex v^i. Namely, we assume the algorithm will receive distance parameter σ and will verify that in addition to $d(v^i)$ belonging to the interval $[d(v^*), f(v^*)]$ the distance between v^* and v^i does not exceed σ. To verify this latter constraint we will assume access to an distance function $d(u, v) = \|u - v\|_2$ for every pair of cell points u and v. The Boolean function ConstrainedBoundaryPoint is realization of the above description.

Function ConstrainedBoundaryPoint(v^i, v^*, σ)

1: **Input:** v^i vertex cell $\in G$ and v^* starting vertex $\in G$:
2: **Output:** A Boolean indicating $v^i \in \Omega_{v^*}$:
3: **value = False**
4: **if** $d(v^i) \in [d(v^*), f(v^*)]$ and $(t(v^*) + \sigma) \leq t(v^i)$ **then**
5: **value = True**
6: **end if**
7: **return value**

To illustrate the utility and performance of our algorithm we have created a simple empirical experiment for the Chesapeake Bay watershed FDG provided as part of the National Hydrography Data-set V2 (10). We compared the computational complexity of the CHSM algorithm and that of custom ESRI analysis by measuring the number of grid cells required to return a common result for differing values of t (the distance from the root) keeping the root location constant (Latitude = 37.025378, Longitude = −75.993728 in World Geodetic System 1984 (WGS 84)) Figure 5 location map.

Constrained Distance (km)	1,160(max)	1,000	500	125	62.5
Traditional Grid Techniques Complexity	$1.97e^8$	$1.89e^8$	$8.89e^7$	$3.17e^7$	$8.90e^6$
CHSM Complexity	$1.65e^5$	$1.22e^5$	$7.45e^4$	$5.48e^4$	$2.21e^4$

Table 1. Results Comparing traditional processing techniques to the the CHSM algorithm for the 30 m FDG provided as part of the National Hydrography Dataset Version 2 (10) using differing distance constraints

5 Conclusions and Future Work.

Comparing upstream attributes to fluvial conditions at any location is typically done at one of the two scales; the watershed or the stream reach. In this manuscript we have proposed a new technique and its associated algorithm and data structure(s) to compute the constrained watershed for any location within a flow direction grid. Other variations of the constrained watershed problem could be used to extract various polygonal surfaces from flow directions grids (or any raster data-sets). For example, given a riparian zone defined as contiguous wetland, riparian zones, or open water, the constrained watershed algorithm could return those boundaries with respect to a starting location on the boundary of the riparian zone. In general, the marching algorithm can be used to return any contiguous grid surface that can be characterized through a first-order Boolean condition. A number of raster to vector problems could be reformulated in terms of marching algorithms (some are already similar to retrieval of contour lines from DEMs). The major advantage of this approach is the speed in which the algorithm computes the required output in comparison to existing area-based techniques. In a simple experiment for the Chesapeake bay watershed we compared the complexity of grid searching algorithms to the CHSM algorithm. The results were consistent with the HSM tests from (8) resulting between 99.94 and 99.59% reduction in complexity based on the user defined constrained distance value t. The CHSM algorithm can be run using a standard PC in less then 1 second for the largest test case in this example.

We note that in some circumstance the application of the CHSM algorithm will leave inclusions where the constrained distance is greater then the user defined threshold value t. This occurs when an area of longer travel distances is surrounded by areas of shorter distances on all sides. This occurred in a number of the test cases used to create table 1, but the total area of the inclusions was never more then 1% of the total watershed. This is similar to the issue we noted on sinks within the HSM manuscript, because the marching algorithm makes local decisions it is unaware of the pocket of longer travel time that is inside it's boundary. Depending on the use case this may or may not be an issue.

In future work, we will focus on efficient ways to return attributes for constrained watersheds given upstream ancillary data-sets. It is possible to sum upstream attributes for any watershed in $O(n)$ where n is the number of cells in the flow direction grid. This complexity is similar to that of preprocessing costs for the creation of the sink filled and breached DEM, and the MNS data-structure. Calculating attributes for the constrained watershed will require a more complex algorithm, and we have had preliminary investigation on using column or row summed values to quickly extract summary statistics during the HSM or CHSM algorithms (personal communication David Tarboton, 2017). More work is required to better understand how other statistics that rely on flow conditions, e.g., distance, or speed can be efficiently computed by a marching algorithm. For sparse data-sets such as the National Pollutant Discharge Elimination System (NPDES) a better solution would be to spatially join the point location with the MNS discovery attribute, this would allow all points sources to be retrieved from a watershed in $2\log_2 n$, where n denotes the number of NPDES points, if the NPDES points where stored in a sorted array by the grid cell discovery values.

REFERENCES

[1] Al-Sabhan, W., Mulligan, M., and Blackburn, G. A. A real-time hydrological model for flood prediction using GIS and the WWW. *Computers, Environment and Urban Systems 27*, 1 (Jan. 2003), 9–32.

[2] Baker, M. E., Weller, D. E., and Jordan, T. E. Comparison of automated watershed delineations: Effects on land cover areas, percentages, and relationships to nutrient discharge. *Photogrammeetric Engineering & Remote Sensing 72*, 2 (2006-02), 1696–1709.

[3] Celko, J. *Joe Celko's Trees and Hierarchies in SQL for Smarties*. Elsevier, 2012.

[4] Daniel, E. B., Camp, J. V., Eugene J. LeBoeuf, Jessica R. Penrod, James P. Dobbins, and Mark D. Abkowitz. Watershed Modeling and its Applications: A State-of-the-Art Review. *The Open Hydrology Journal 5*, 5 (2011), 26–50.

[5] Demir, I., and Szczepanek, R. Optimization of river network representation data models for web-based systems: Optimization of network representation. 336–347.

[6] ESRI, E. S. R. I. Esri's hydrologic analysis tools.

[7] Haag, S., and Shokoufandeh, A. Development of a data model to facilitate rapid watershed delineation. *Environmental Modelling & Software (In Press)*, DOI:10.1016/j.envsoft.2017.06.009 (2017).

[8] Haag, S., and Shokoufandeh, A. A watershed delineation algorithm for 2d flow direction grids. *CoRR abs/1708.00354* (2017).

[9] Hill, R. A., Weber, M. H., Leibowitz, S. G., Olsen, A. R., and Thornbrugh, D. J. The Stream-Catchment (StreamCat) Dataset: A Database of Watershed Metrics for the Conterminous United States. *J Am Water Resour Assoc 52*, 1 (Feb. 2016), 120–128.

[10] Horizon Systems, EPA, o. NHD PLus version 2 user manual.

[11] Jenson, S. K., and Domingue, J. O. Extracting topographic structure from digital elevation data for geographic information system analysis. *Photogrammetric engineering and remote sensing 54*, 11 (1988), 1593–1600.

[12] Lindsay, J. B. Efficient hybrid breaching-filling sink removal methods for flow path enforcement in digital elevation models: Efficient hybrid sink removal methods for flow path enforcement. 846–857.

[13] Malvadkar, U., Scatena, F., and Leon, M. A Comparison of Connectivity Metrics on Watersheds and Implications for Water Management. *River Res. Applic. 31*, 2 (Feb. 2015), 256–267.

[14] Richards Carl, Haro Roger, Johnson Lucinda, and Host George. Catchment and reach-scale properties as indicators of macroinvertebrate species traits. 219–230.

[15] Sethian, J. A. *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge University Press. Google-Books-ID: ErpOoynE4dIC.

[16] Tarboton, D. G. Terrain analysis using digital elevation models (TAUDEM).

[17] Tesfa, T. K., Tarboton, D. G., Watson, D. W., Schreuders, K. A. T., Baker, M. E., and Wallace, R. M. Extraction of hydrological proximity measures from dems using parallel processing. *Environmental Modeling & Software 12* (2011-12), 1696–1709.

[18] Wang, L., Lyons, J., Kanehl, P., and Gatti, R. Influences of Watershed Land Use on Habitat Quality and Biotic Integrity in Wisconsin Streams. *Fisheries 22*, 6 (June 1997), 6–12.

Appendix: Original Haag Shokoufandeh March Algorithm

Function 1. LatticeMove (ℓ, μ)

Input: Lattice point $\ell \in \mathcal{L}$ and μ direction $\in$ {0,1,2,3,}. Here ℓ is uniquely identified as $\ell = \Lambda(x(\ell), y(\ell))$,
Output: new lattice cell ℓ in direction μ.

if $\mu == 0$ **then**
$\mathcal{W} = A(\ell)$
$\mathcal{E} = B(\ell)$
end if

if $\mu == 1$ **then**
$\mathcal{W} = B(\ell)$
$\mathcal{E} = \Theta(\ell)$
end if

if $\mu == 2$ **then**
$\mathcal{W} = \Theta(\ell)$
$\mathcal{E} = \Delta(\ell)$
end if

if $\mu == 3$ **then**
$\mathcal{W} = \Delta(\ell)$
$\mathcal{E} = A(\ell)$
end if
$\mathcal{W}$ = BoundaryPoint($\mathcal{W}$, v^*)
$\mathcal{E}$ = BoundaryPoint($\mathcal{E}$, v^*)

if $\mathcal{W}$ == T and $\mathcal{E}$ == F **then return** $\Lambda(x(\ell) + \delta x, y(\ell) + \delta y)$
end if $\mathcal{P} = (\mu + 1) \pmod 2$
$\mathcal{S} = \lceil \mu/2 \rceil$
$\delta x = (-1)^{(1+\mathcal{S})}|\mathcal{P}|$
$\delta y = (-1)^{(1+\mathcal{S})}|\mathcal{P} - 1|$
return $\Lambda(x(\ell) + \delta x, y(\ell) + \delta y)$

Function 2. BoundaryPoint(v^i, v^*)

Input: v^i vertex cell $\in G$ and v^* starting vertex $\in G$:
Output: A Boolean indicating $v^i \in \Omega_{v^*}$:
value = False
if $d(v^i) \in [d(v^*), f(v^*)]$ **then**
    value = True
end if return value

Algorithm 1. Watershed Marching Algorithm $(G, \mathcal{L}, v^*)$

Input: Graph $G(V, E)$ such that $\forall v \in V$ and Lattice $\mathcal{L}$ such that $\forall\ \ell \in \mathcal{L}$:
    - v^* has discovery, $d(v)$, and finish, $f(v)$, values computed by MNS algorithm.
    - Lattice point $\ell \in \mathcal{L}$ has unique grid coordinates pair $\Lambda(x(\ell), y(\ell))$,

Output: Watershed as a (cyclic) array of lattice points:
        $\Omega(v^*) = \langle \Lambda(x_1, y_1), \Lambda(x_2, y_2), \ldots, \Lambda(x_k, y_k), \Lambda(x_1, y_1), \Lambda(x_2, y_2) \rangle$.
$\mathcal{C} = \langle \beta(), \gamma(), \gamma(), \delta(), \delta(), \alpha(), \alpha(), \beta() \rangle$ ▷ Aux. array of possible lattice functions
$\mathcal{D} = \langle 1, 1, 2, 2, 3, 3, 0, 0 \rangle$ ▷ Aux. array of possible cardinal transition directions
$idx = \lg D8(v^*)$ ▷ Aux. var for the index in arrays $\mathcal{C}$ and $\mathcal{D}$
$l_{v^*} = \mathcal{C}[idx](\rho(v^*))$ ▷ Select the corner function ▷ Identify Starting Lattice location
$\Omega(v^*) = \langle l_{v^*} \rangle$ ▷ Store the lattice coordinate pair in the final watershed
$\mu \leftarrow \mathcal{D}[idx]$ ▷ Store march direction state
while $(\Omega[1] \neq \Omega[|\Omega| - 1]$ and $\Omega[2] \neq \Omega[|\Omega|])$ **do** ▷ Stop when the first edge is recrossed.
    $\ell, \mu \leftarrow$ **LatticeMove**$(\Omega[|\Omega|], \mu)$. ▷ apply march function based on direction
end while
$\Omega \leftarrow \Omega[1, ..., |\Omega| - 1]$ ▷ remove the duplicate vertex at the end of the polygon.
return Ω