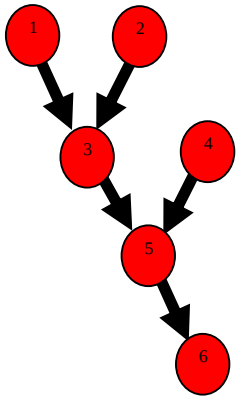
Graph Theory for Watershed Analysis

Nick van de Giesen (n.c.vandegiesen@tudelft.nl)

For a general introduction to Graph Theory, please have a look at Wikipedia (http://en.wikipedia.org/wiki/Graph_theory).

Two cases will be discussed:
1) Drainage patterns (determination of river networks and watershed areas)
2) Water use along a river network (WEAP)

1 Drainage patterns



If we divide an area over equal square elements, or pixels, and assign a height to each element, we can, for each pixel assign a lower neighboring pixel to which all excess water will flow. A very simple example of six pixels is given to the left. Assume each circle is a pixel and that all pixels fill (part of) a plane. Pixel 1&2 drain into pixel 3, which drains into 5, etc. Such a system of nodes and links (or edges) is called a graph. This particular type of graph is called a "tree" because ... (homework).

We can develop an adjacency matrix, *A*, in which we enter a one in each cell (*n*,*m*) where node *n* drains into cell *m*, and zeros where no connections exist. For our example, this would look like:

A

	1	2	3	4	5	6
1	0	0	1	0	0	0
2	0	0	1	0	0	0
3	0	0	0	0	1	0
4	0	0	0	0	1	0
5	0	0	0	0	0	1
6	0	0	0	0	0	0

One could say that the ones indicate which node *n* (row) is one step removed from node *m* (column). We could also make a matrix, *A'*, that shows a one wherever two nodes are connected through two steps, and *A''* for three steps:

A'

	1	2	3	4	5	6
1	0	0	0	0	1	0
2	0	0	0	0	1	0
3	0	0	0	0	0	1
4	0	0	0	0	0	1
5	0	0	0	0	0	0
6	0	0	0	0	0	0

A''

	1	2	3	4	5	6
1	0	0	0	0	0	1
2	0	0	0	0	0	1
3	0	0	0	0	0	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0
6	0	0	0	0	0	0

Through inspection, one can quickly check that $A'=A^2$ and $A''=A^3$. Remember that the product of two matrices, $A \cdot B$, is formed by entering in each cell (*n*,*m*), the inproduct of the row *n* of *A* with and column *m* of *B*:

$$(AB)_{ij} = \sum_{r=1}^{n} a_{ir} b_{rj} = a_{i1} b_{1j} + a_{i2} b_{2j} + \cdots a_{in} b_{nj}$$

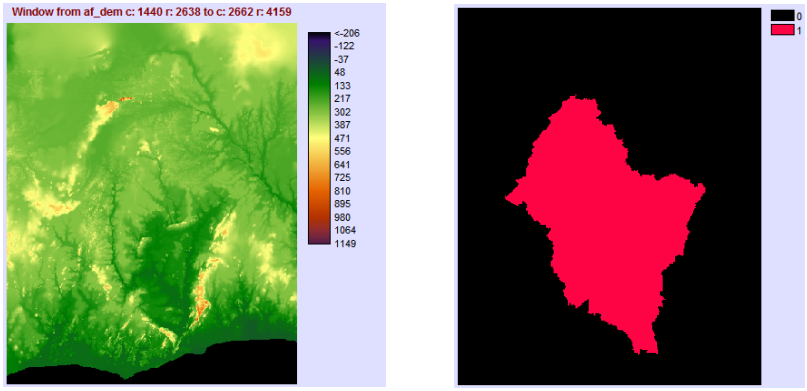
In general, $A \cdot B \neq B \cdot A$, so we distinguish pre-multiplication and post-multiplication. (http://en.wikipedia.org/wiki/Matrix_multiplication).

The sum of these matrices, $\Sigma=A+A^2+A^3$, would in each row show the "downstream" pixels and in the columns all "upstream" pixels. In the case of a watershed, we would also like to include a "zero-th" step, the unity matrix, I, because each pixel drains also itself. Clearly, node 3 in our example drains 1,2, and 3. If we add I, we obtain the "watershed" matrix, $\Sigma+I$:

Σ+I	1	2	3	4	5	6
1	1	0	1	0	1	1
2	0	1	1	0	1	1
3	0	0	1	0	1	1
4	0	0	0	1	1	1
5	0	0	0	0	1	1
6	0	0	0	0	0	1

So the water from node 1 flows through 1, 3,5 & 6. All nodes drain through node 6 or, in other words, form the watershed of 6. By summing the columns, we quickly find the watershed size (or drainage area) for each pixel: 1=>1, 2=>1, 3=>3, 4=>1, 5=>5, 6=>6. Interestingly, $I+A+A^2+...+A^n=(I-A)^{-1}$, which can easily be checked by pre-multiplying left and right with $(I-A)$. This means that we can derive the *complete structure of the watershed by just looking at the local drainage directions*.

Practically, GIS software uses the above algorithm to determine watershed areas for each pixel, whereby the height of each pixel is given by a Digital Elevation Model (such as GTOPO30 and SRTM, see http://eros.usgs.gov/products/elevation.html). By cutting off pixels with a watershed area below a certain threshold, one obtains a drainage or river network. Similarly, one can quickly outline for any chosen pixel what the watershed area is.

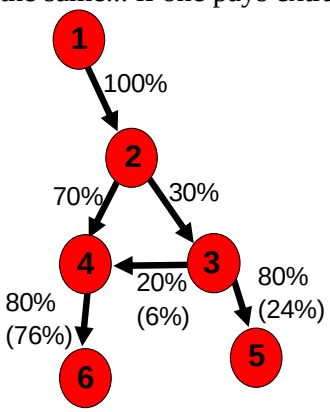


Sample application: Left GTOPO30 for the Volta Basin and environs, right watershed of the Akosombo dam.

2 Water use along a river network

In the example of a watershed, the same weight of one was given to each pixel, assuming all pixels had the same size. One could also imagine that different pixels produce different amounts of runoff and that one wants to know how these flows accumulate along the stream network. If, say, Node 1 produces 3.2 units of water, Node 2, 2.3 units, etc., one would simply multiply row 1 with 3.2, row 2 with 2.3 etc. This is the same as pre-multiplying the matrix $\Sigma+I$ with a row array that contains the water production of each cell. The sums of the columns would then tell us how much water would, cumulatively, be available at each node.

How does this work for a river network with extraction, like in WEAP? Pretty much the same... if one pays extra attention. If we look at the example to the left, we have again a node-link network. From Node 1, all available water flows to Node 2. At Node 2, the water is split, 70% continues to Node 4, 30% goes to Node 3. From a water balance point of view, it is helpful if the totals add up to 100% but that is not essential from a mathematical point of view. At Node 3, 80% of the available water is used by sending it to Node 5, and 20% return-flow flows back to Node 4. At Node 4, the water from Node 2 and Node 3 is collected and sent to Node 6. If water only enters the system through Node 1 (100%), it is easy to see that the numbers between brackets are the amounts flowing through those links. Not surprisingly, the sum of the water available at the two end Nodes (5&6) add up to 100%.



The adjacency matrix, *W*, now looks like:

W	1	2	3	4	5	6
1	0.00	1.00	0.00	0.00	0.00	0.00
2	0.00	0.00	0.30	0.70	0.00	0.00
3	0.00	0.00	0.00	0.20	0.80	0.00
4	0.00	0.00	0.00	0.00	0.00	1.00
5	0.00	0.00	0.00	0.00	0.00	0.00
6	0.00	0.00	0.00	0.00	0.00	0.00

Again, we can multiply W with itself and each time we do that, we obtain a matrix that shows how much flows from one Node *n* to another Node *m* in a number of steps that is equal to the power of the matrix. For example, W^3 is:

A^3	1	2	3	4	5	6
1	0.00	0.00	0.00	0.06	0.24	0.70
2	0.00	0.00	0.00	0.00	0.00	0.06
3	0.00	0.00	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.00	0.00
6	0.00	0.00	0.00	0.00	0.00	0.00

It can readily be verified that, indeed, 6% of the water available at Node 2 will be available at Node 6 after three steps, etc.

The maximum number of steps for the given graph/system is four. If we calculate the sum $\Omega = W + W^2 + W^3 + W^4$, we obtain:

Ω	1	2	3	4	5	6
1	0.00	1.00	0.30	0.76	0.24	0.76
2	0.00	0.00	0.30	0.76	0.24	0.76
3	0.00	0.00	0.00	0.20	0.80	0.20
4	0.00	0.00	0.00	0.00	0.00	1.00
5	0.00	0.00	0.00	0.00	0.00	0.00
6	0.00	0.00	0.00	0.00	0.00	0.00

Note that the unity matrix I has not been added because that assumes that there is a source of water of 100% at each node. The math still holds so instead of calculating the sum of the powers, we can also calculate $\Omega = (I-W)^{-1} - I$. (Check with, for example, Excel or MatLab).

WEAP assumes that any water available upstream that is not used at a node, is immediately available downstream. So there is no delay or hydraulic routing involved. In the watershed example, we summed the columns but that does not seem to be directly applicable here. Instead, we pre-multiply Ω with the row vector that contains the sources at each node. So if water enters the system only through Node 1 (say 200 units of water), we pre-multiply Ω with the vector (200,0,0,0,0,0) to obtain:

	1	2	3	4	5	6
Q	0	200	60	152	48	152

And the columns give indeed the amount of water available at each node.

As an exercise: check, by adding an extra input node or by simply adding a source term to an existing node, if more complicated networks still "work". What happens with water use? What happens if more water is used ata a node than is available? Could your correct for that?
Bonus question: Are there networks for which this does not work (or: for which network is $I-W$ singular?).

Resume:

1. Fill adjacency matrix W (with fractions if appropriate)
2. Calculate the water availability matrix $\Omega = (I-W)^{-1} - I$
3. Pre-multiply Ω with the row vector containing for each node the amount of water produced at that node
4. Sum the columns to obtain the amount of water available at each node