UFV

Computing the drainage network on huge grid terrains

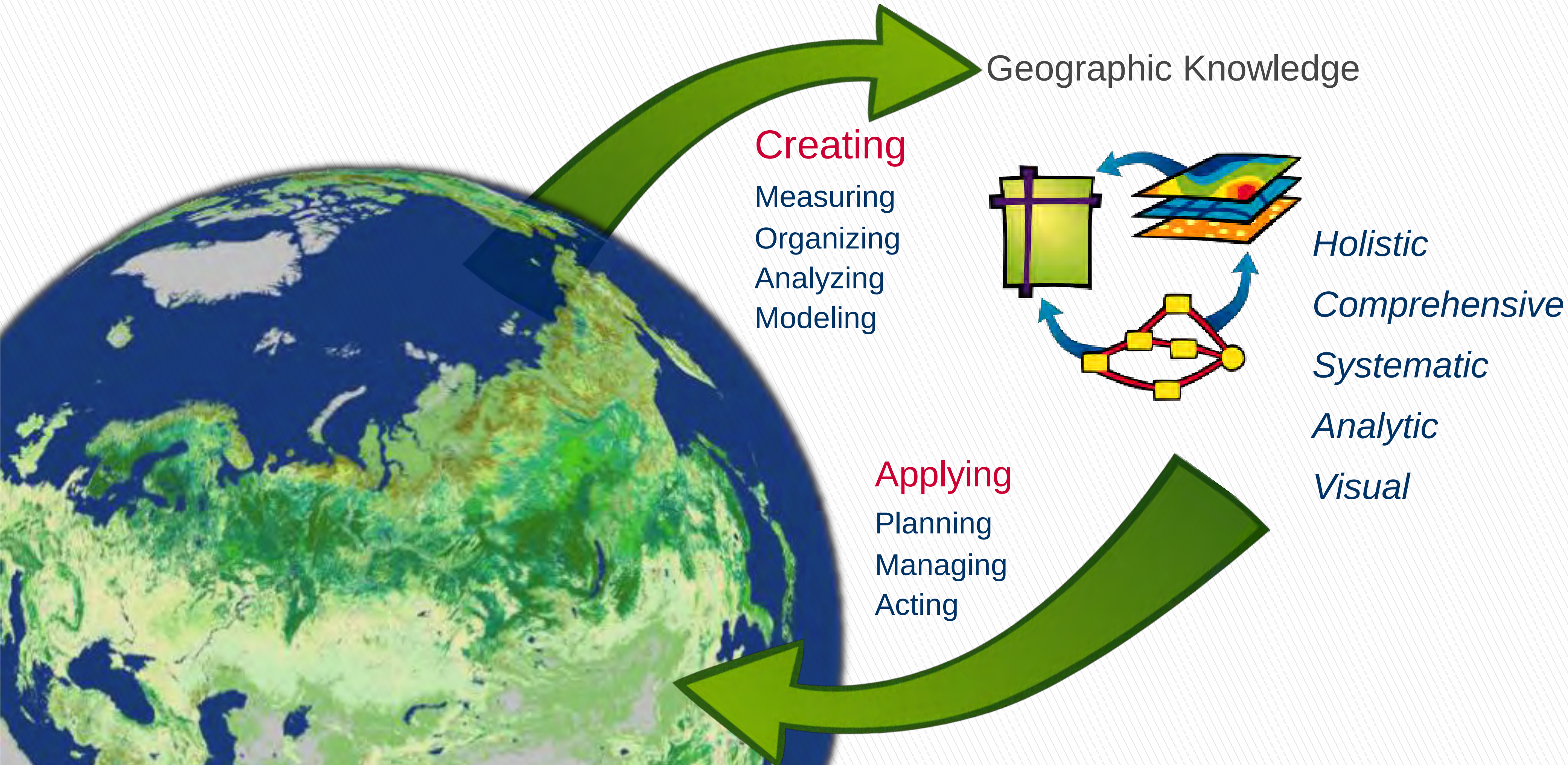
Thiago L. Gomes
Salles V. G. Magalhães
Marcus V. A. Andrade
W. Randolph Franklin
Guilherme C. Pena

Universidade Federal de Viçosa (UFV) – Brazil
Rensselaer Polytechnic Institute (RPI) – USA

Summary

- Introduction: GIS and Big Data
- Drainage network computation
- *EMFlow* algorithm description
- Implementation details
- Results

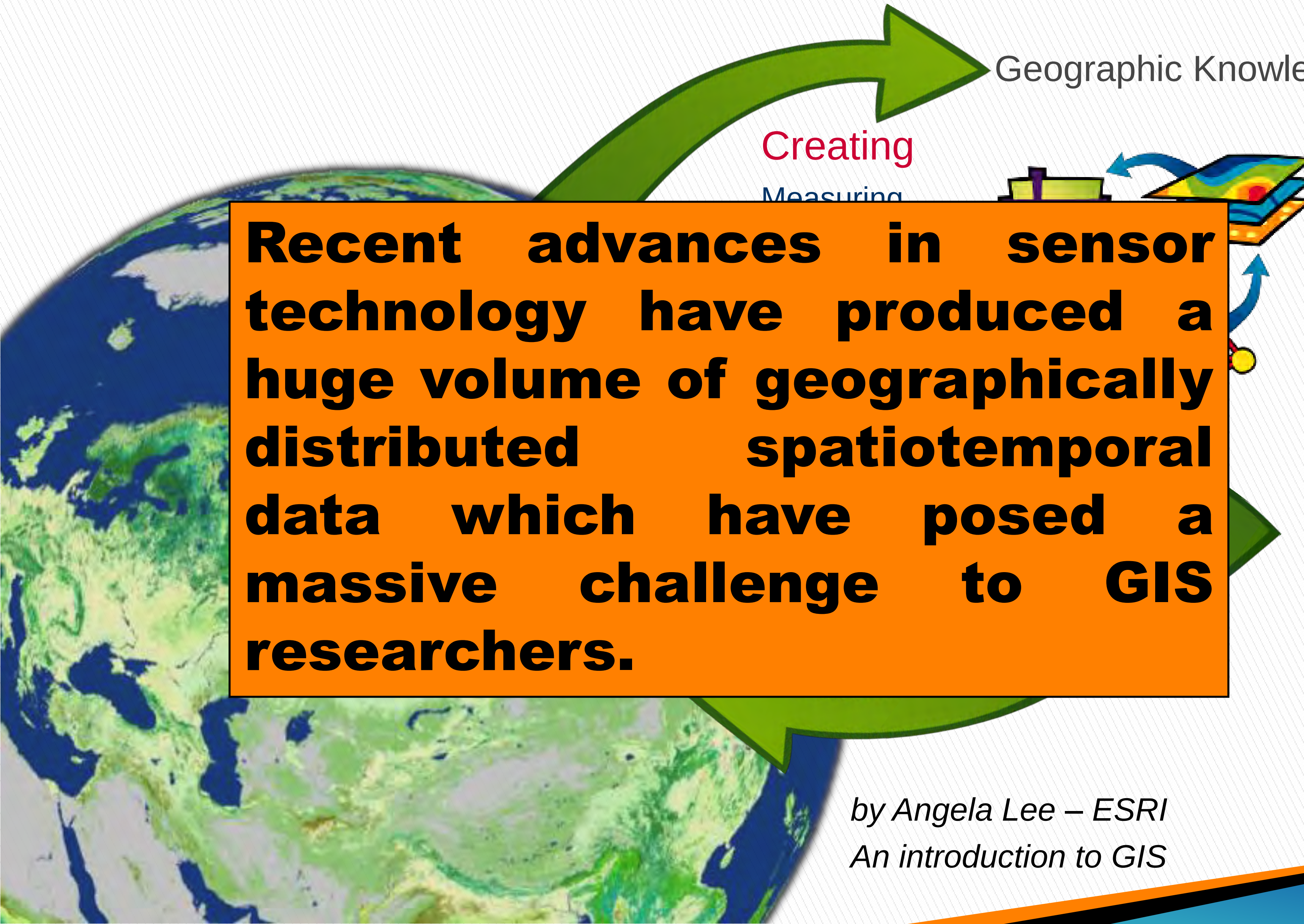
GIS: A Framework for Understanding and Managing Our Earth



by Angela Lee – ESRI

An introduction to GIS

GIS: A Framework for Understanding and Managing Our Earth



Recent advances in sensor technology have produced a huge volume of geographically distributed spatiotemporal data which have posed a massive challenge to GIS researchers.

Holistic
Comprehensive
Systematic
Analytic
Visual

by Angela Lee – ESRI
An introduction to GIS

External memory processing

- On most computers, such huge volume of data do not fit in internal memory and need to be processed externally (mainly in disks);

External memory processing

- On most computers, such huge volume of data do not fit in internal memory and need to be processed externally (mainly in disks);
- But, in this case, the algorithms designed for internal processing do not run well since the time to access data on disk is much higher than the internal access;

External memory processing

- On most computers, such huge volume of data do not fit in internal memory and need to be processed externally (mainly in disks);
- But, in this case, the algorithms designed for internal processing do not run well since the time to access data on disk is much higher than the internal access;
- Thus, the communication between the fast internal memory and the slow external memory is often the performance bottleneck;

External memory processing

- Thus, the algorithms must be designed focusing the optimization of I/O operations (data movements); not only the CPU processing;

External memory processing

- Thus, the algorithms must be designed focusing the optimization of I/O operations (data movements); not only the CPU processing;
- These algorithms are termed external memory (or I/O-efficient) algorithms;

External memory processing

- Thus, the algorithms must be designed focusing the optimization of I/O operations (data movements); not only the CPU processing;

- These algorithms are termed external memory (or I/O-efficient) algorithms;

- To show how the algorithm performance can be affected by the external memory access, suppose you want to print a huge matrix *M* with *n* x *n* cells stored in external memory;

External memory processing

- Consider these two methods:

__Alg. 1__

```
for (i=1; i <= n; i++)
  for (j=1; j <= n; j++)
    cout << M[i,j];
```

__Alg. 2__

```
for (j=1; j <= n; j++)
  for (i=1; i <= n; i++)
    cout << M[i,j];
```

External memory processing

- Consider these two methods:

Alg. 1

for (*i=1; i <= n; i++***)**
  for (*j=1; j <= n; j++***)**
    cout << *M*[i,j];

Alg. 2

for (*j=1; j <= n; j++***)**
  for (*i=1; i <= n; i++***)**
    cout << *M*[i,j];

- Based on CPU instructions, both algorithms are $\Theta(n^2)$;

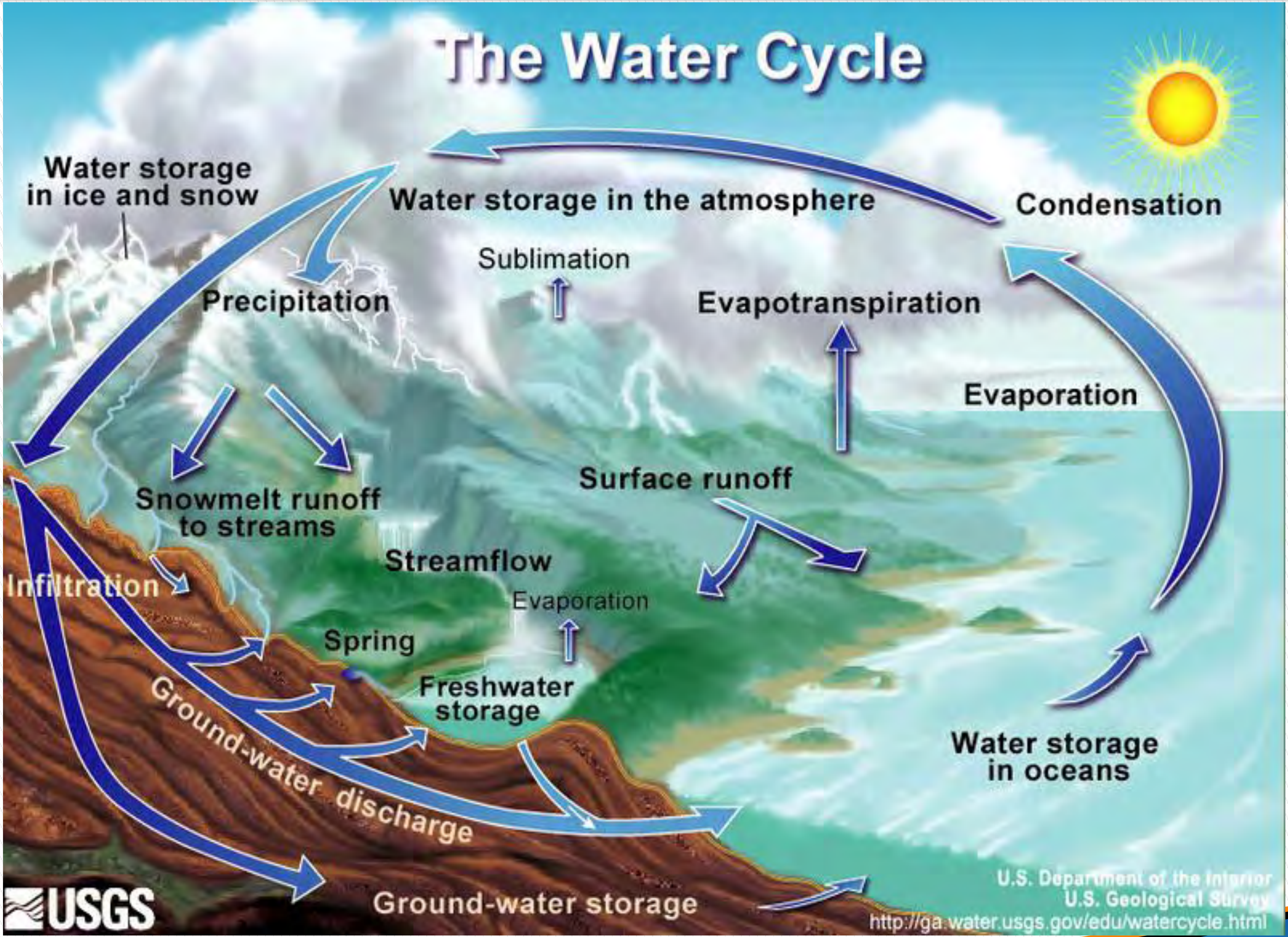
External memory processing

- But, considering I/O operations, if the block size *B* is smaller than the matrix row:
  - Algorithm 1 executes $\Theta(n^2/B)$ I/O operations
  - Algorithm 2 executes $\Theta(n^2)$ I/O operations

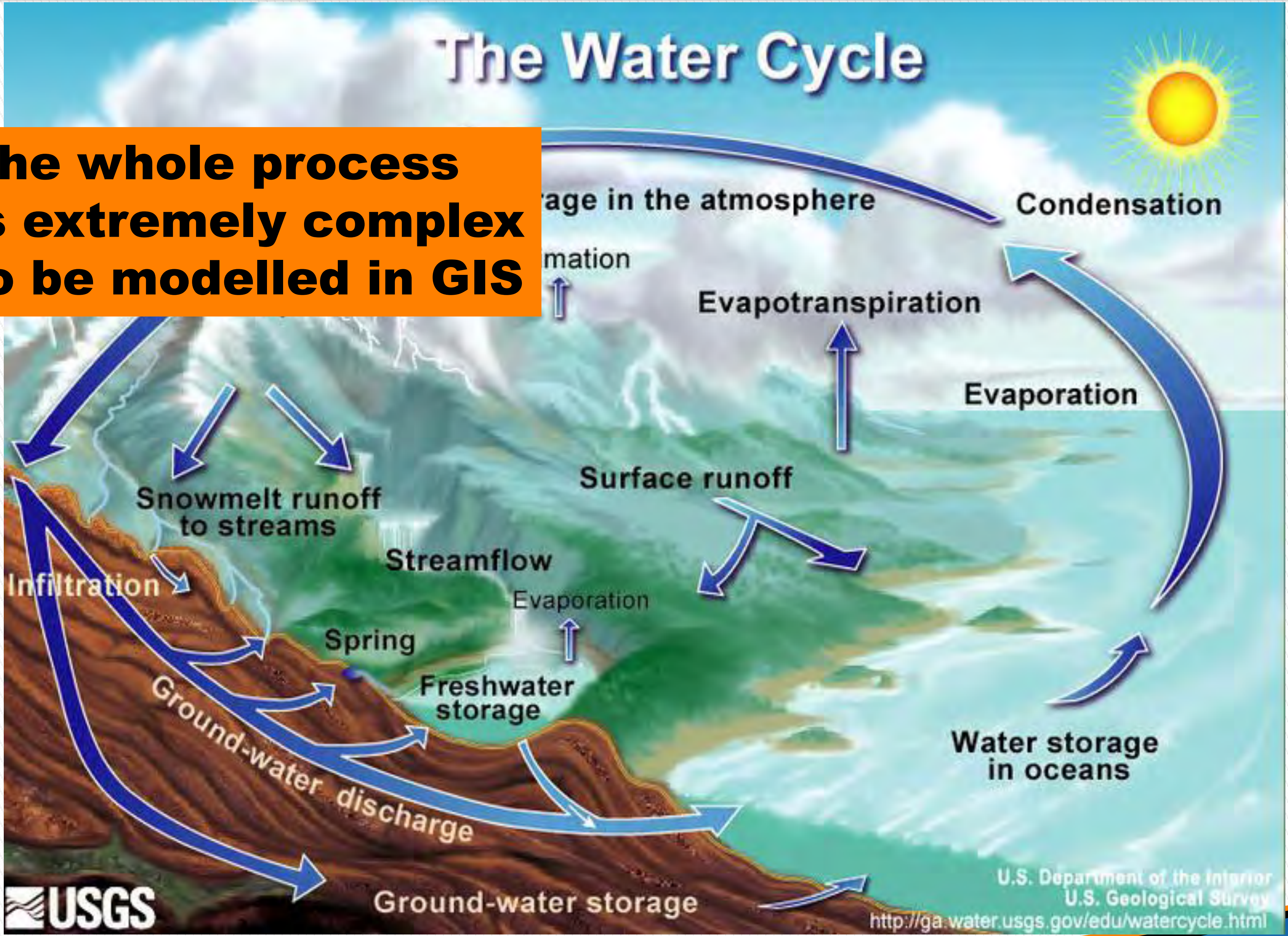
External memory processing

- But, considering I/O operations, if the block size *B* is smaller than the matrix row:
  - Algorithm 1 executes $\Theta(n^2/B)$ I/O operations
  - Algorithm 2 executes $\Theta(n^2)$ I/O operations
- In a machine where the cache memory can store 10000 cells and the time to read a block is 10 miliseconds (9 for seek and 1 for read), the time to read (and print) a matrix with 50000^2 cells is:
  - Algorithm 1 ≈ 4 minutes
  - Algorithm 2 ≈ 10 months

Hydrologic modelling

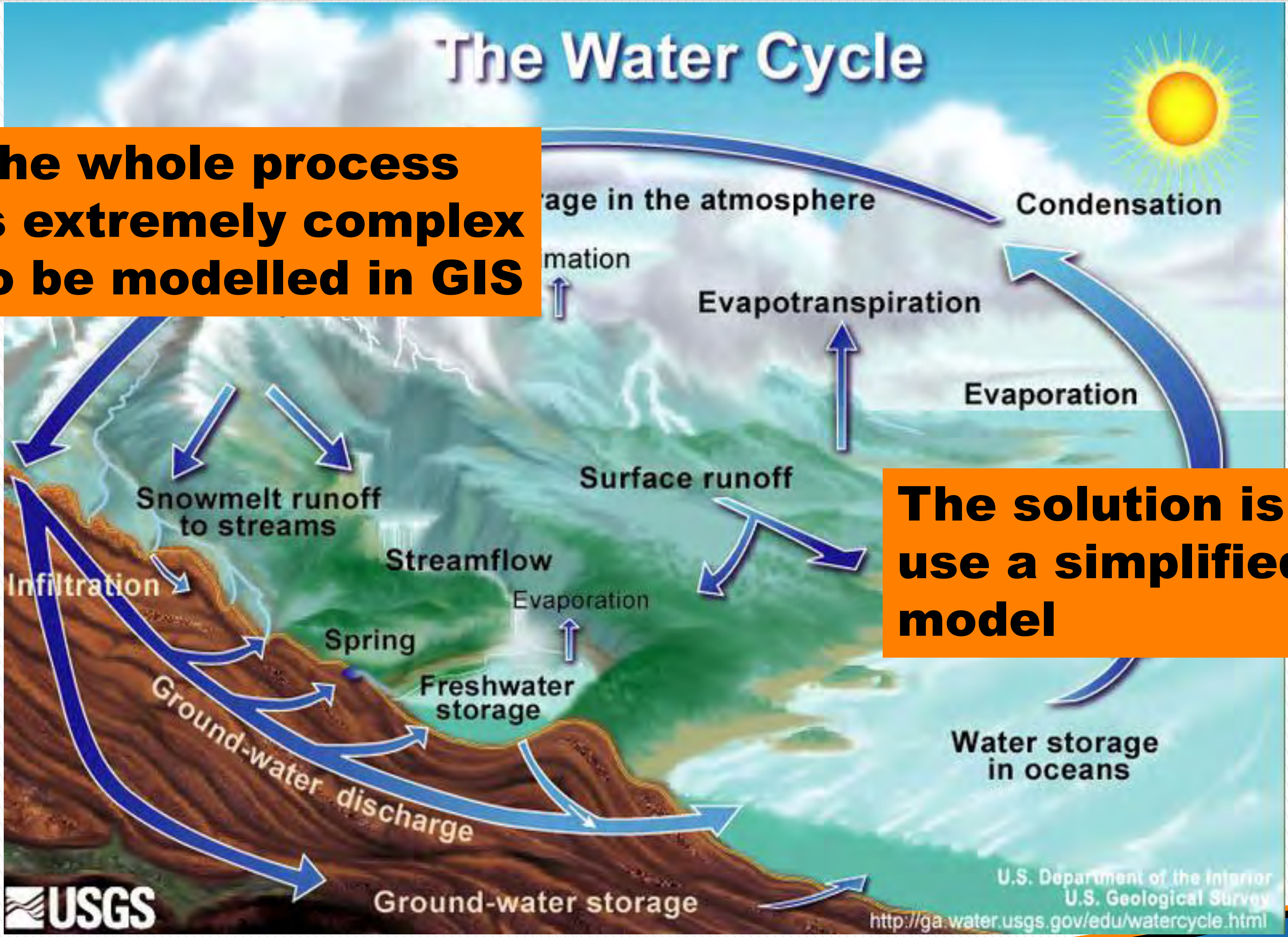


Hydrologic modelling



The whole process is extremely complex to be modelled in GIS

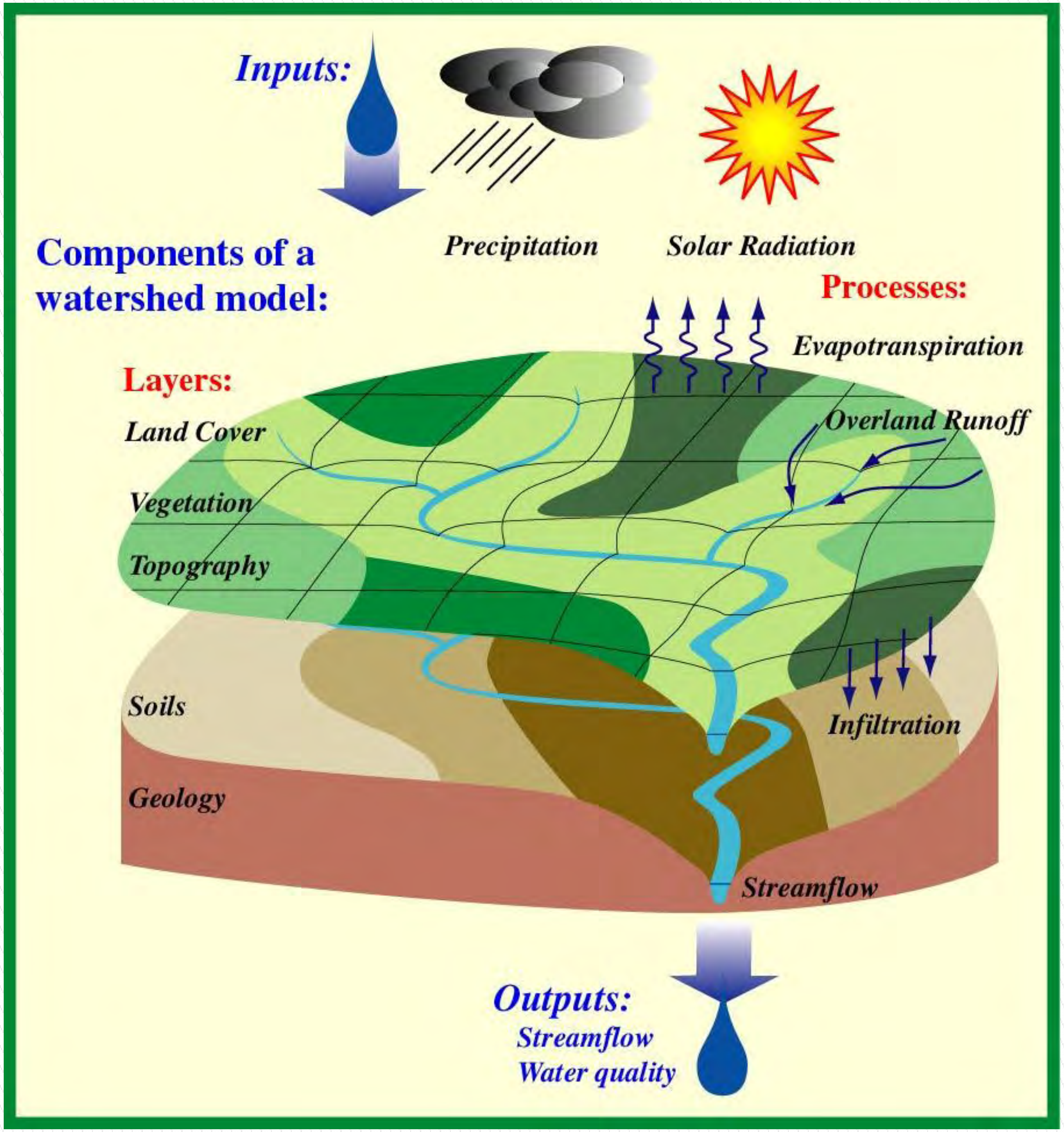
Hydrologic modelling



The whole process is extremely complex to be modelled in GIS

The solution is to use a simplified model

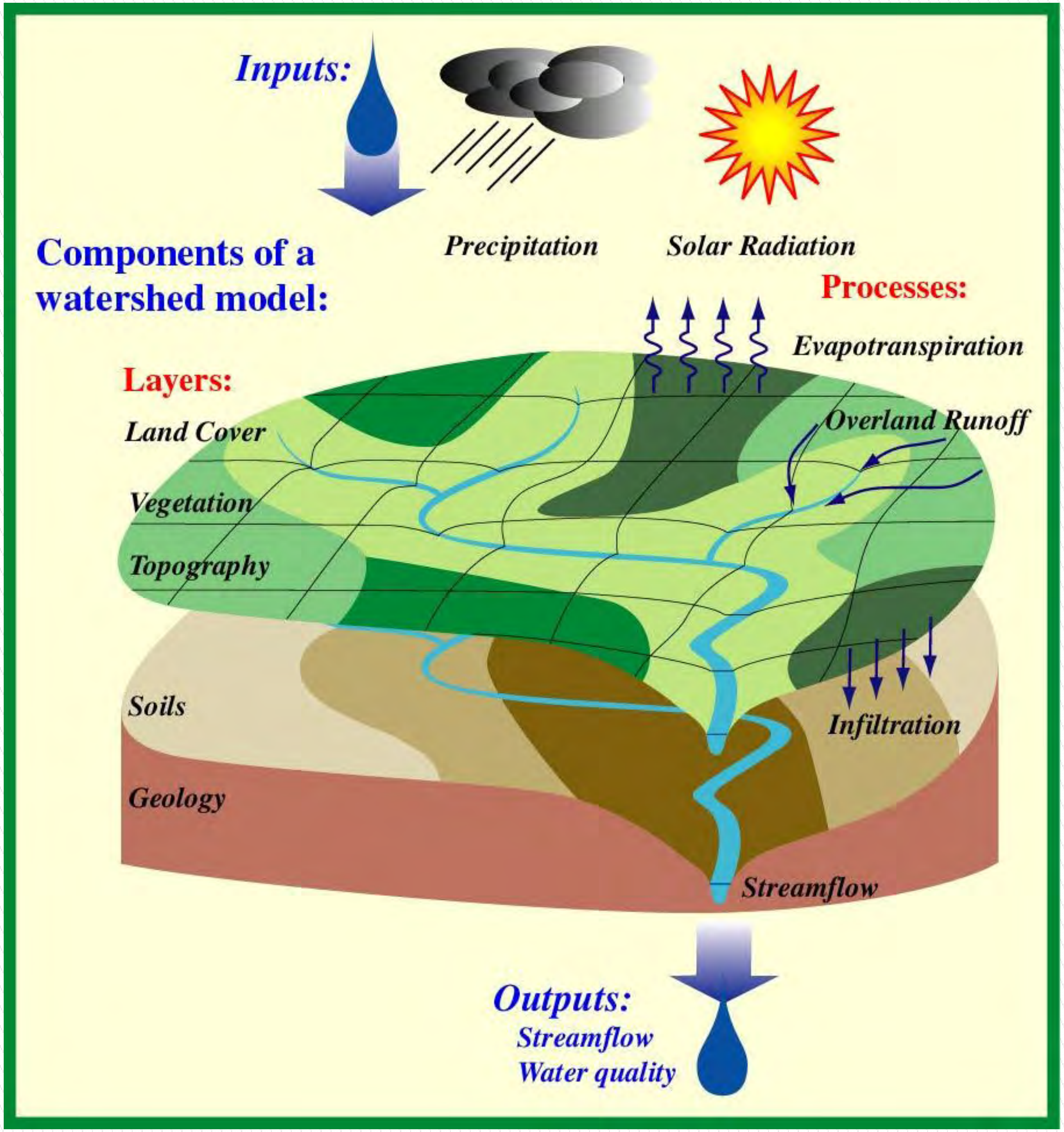
Hydrologic model and GIS



http:// swatmodel.tamu.edu

- A simplification generally used is to describe basically the overland runoff;

Hydrologic model and GIS



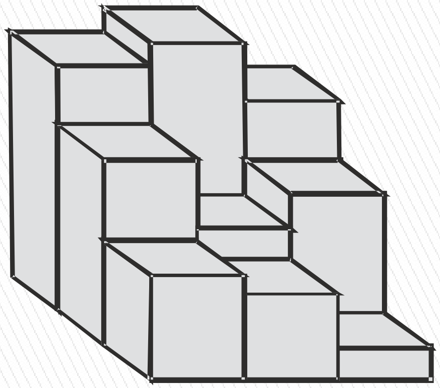
http:// swatmodel.tamu.edu

- Considering the precipitation falling on the land, it models the drainage network (the rivers), which then empty into the oceans;

Drainage network computation

71	72	67
68	62	65
63	61	58

DEM

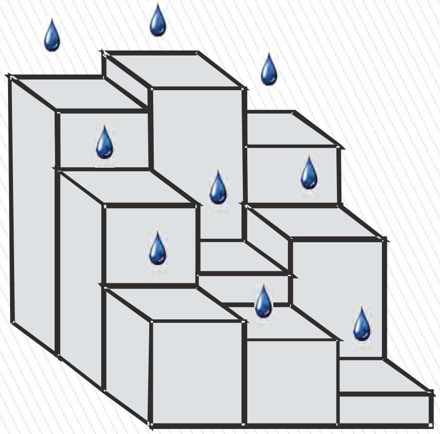


3D Viewing

Drainage network computation

71	72	67
68	62	65
63	61	58

DEM

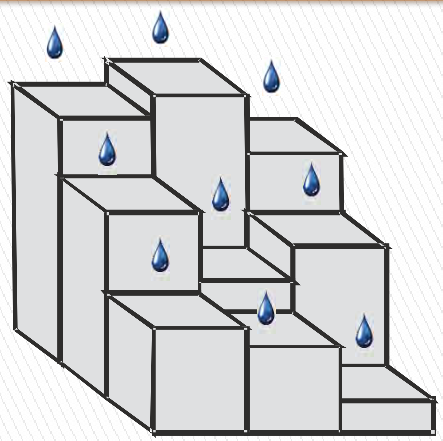


3D Viewing

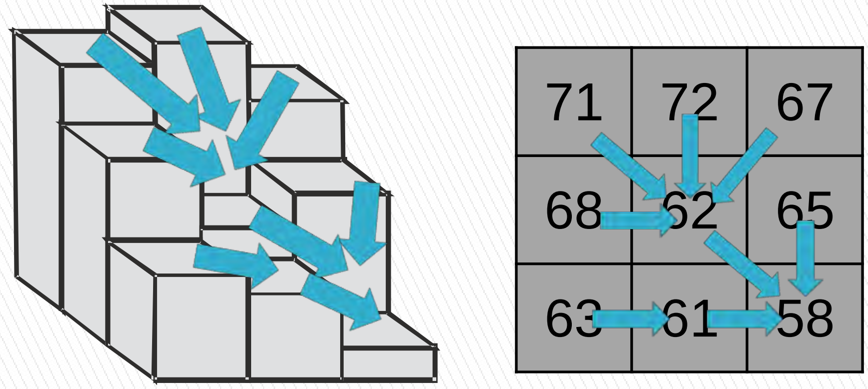
Drainage network computation

71	72	67
68	62	65
63	61	58

DEM



3D Viewing

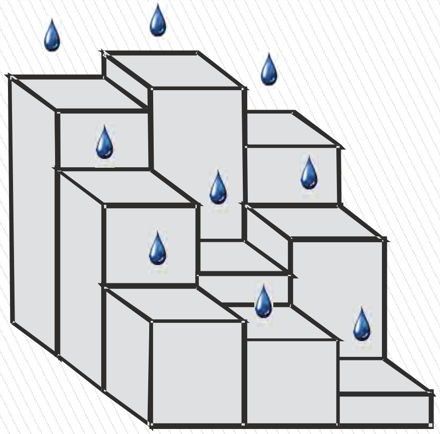


Flow direction

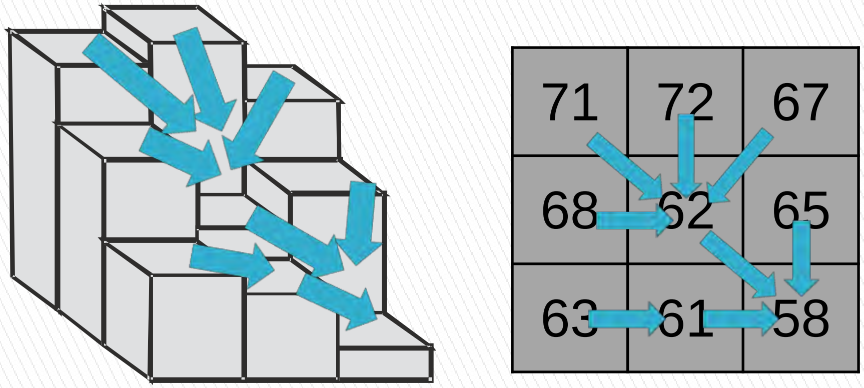
Drainage network computation

71	72	67
68	62	65
63	61	58

DEM



3D Viewing



Flow direction

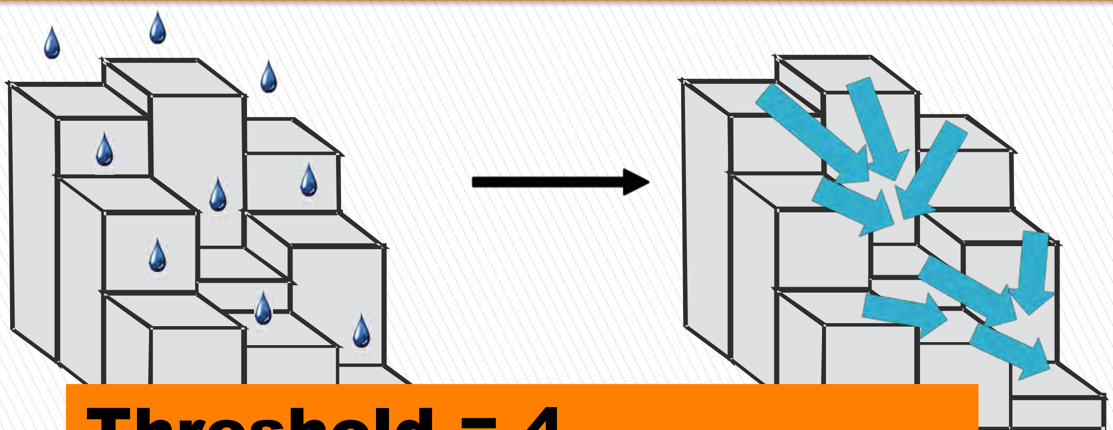
1	1	1
1	5	1
1	2	9

Flow accumulation

Drainage network computation

71	72	67
68	62	65
63	61	58

DEM



71	72	67
68	62	65
63	61	58

Flow direction

Threshold = 4
the drainage network is composed by all cells with "*flow accum ≥ 4*"

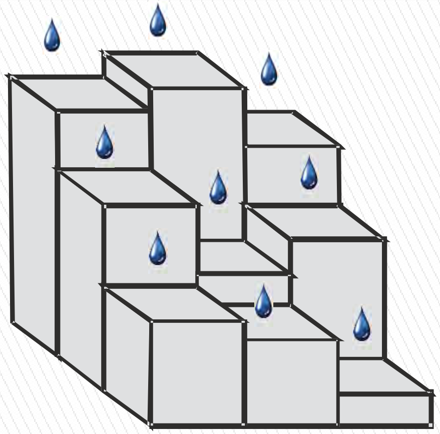
1	1	1
1	5	1
1	2	9

Flow accumulation

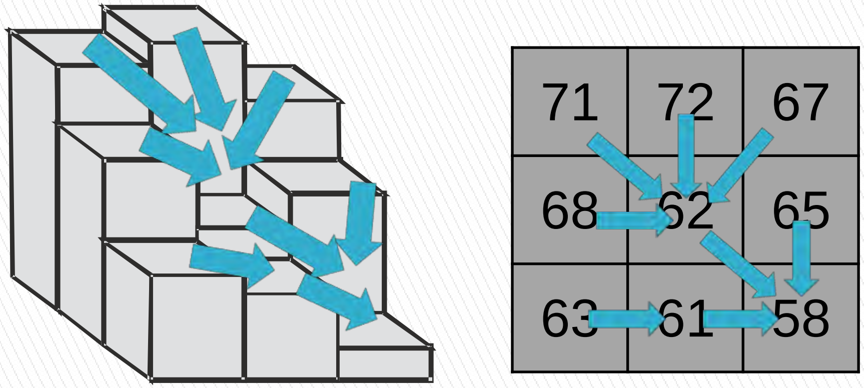
Drainage network computation

71	72	67
68	62	65
63	61	58

DEM



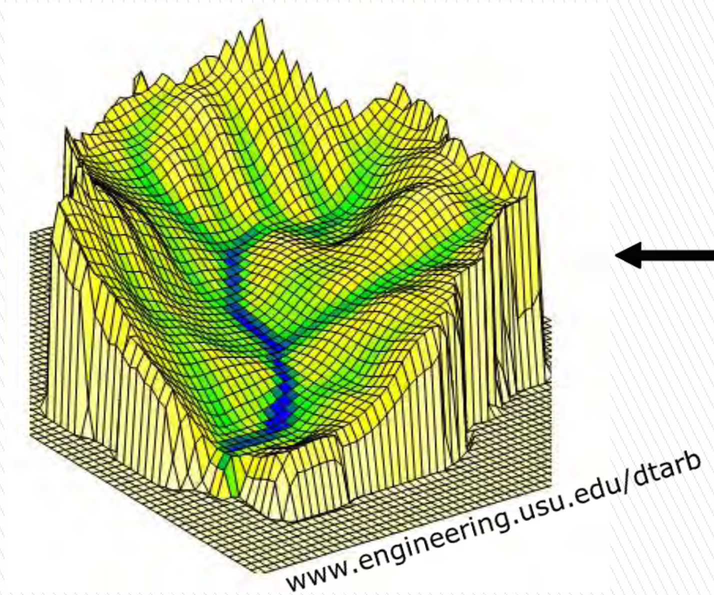
3D Viewing



Flow direction

1	1	1
1	5	1
1	2	9

Flow accumulation



Drainage network

Challenges

- In some cases, it is not possible to determine the flow direction in a cell:

71	72	67	71	72
68	62	65	68	62
63	61	58	63	61
68	62	65	68	62
71	72	67	71	72

Local minimum (depression)

71	72	67	71	72
67	68	68	68	62
63	68	68	68	61
68	62	65	68	62
71	72	67	71	72

Flat area

Challenges

- In some cases, it is not possible to determine the flow direction in a cell:

71	72	67	71	72
68	62	65	68	62
63	61	58	63	61
68	62	65	68	62
71	72	67	71	72

Local minimum (depression)

71	72	67	71	72
67	68	68	68	62
63	68	68	68	61
68	62	65	68	62
71	72	67	71	72

Flat area

- In general, these two cases are treated by a very time-consuming preprocessing step;

Preprocessing step

- A depression is removed by filling it; that is, its elevation is raised to the elevation of its lowest neighbor;

Preprocessing step

- A depression is removed by filling it; that is, its elevation is raised to the elevation of its lowest neighbor;

- And, the flow direction in flat areas is oriented to the lowest neighbor cell;

Preprocessing step

- A depression is removed by filling it; that is, its elevation is raised to the elevation of its lowest neighbor;
- And, the flow direction in flat areas is oriented to the lowest neighbor cell;
- In general, this preprocessing step takes about 50% of the total running time;

The *EMFlow* method

- To avoid this time-consuming preprocessing step, we developed the *RWFlood* method which is very efficient when the whole terrain fits in internal memory;

The *EMFlow* method

- To avoid this time-consuming preprocessing step, we developed the *RWFlood* method which is very efficient when the whole terrain fits in internal memory;

- But, it does not scale well for huge terrains requiring external memory processing;

The *EMFlow* method

- To avoid this time-consuming preprocessing step, we developed the *RWFlood* method which is very efficient when the whole terrain fits in internal memory;
- But, it does not scale well for huge terrains requiring external memory processing;
- Thus, the idea of this work (the *EMFlow* method) is to adapt the *RWFlood* for external processing;

RWFlood description

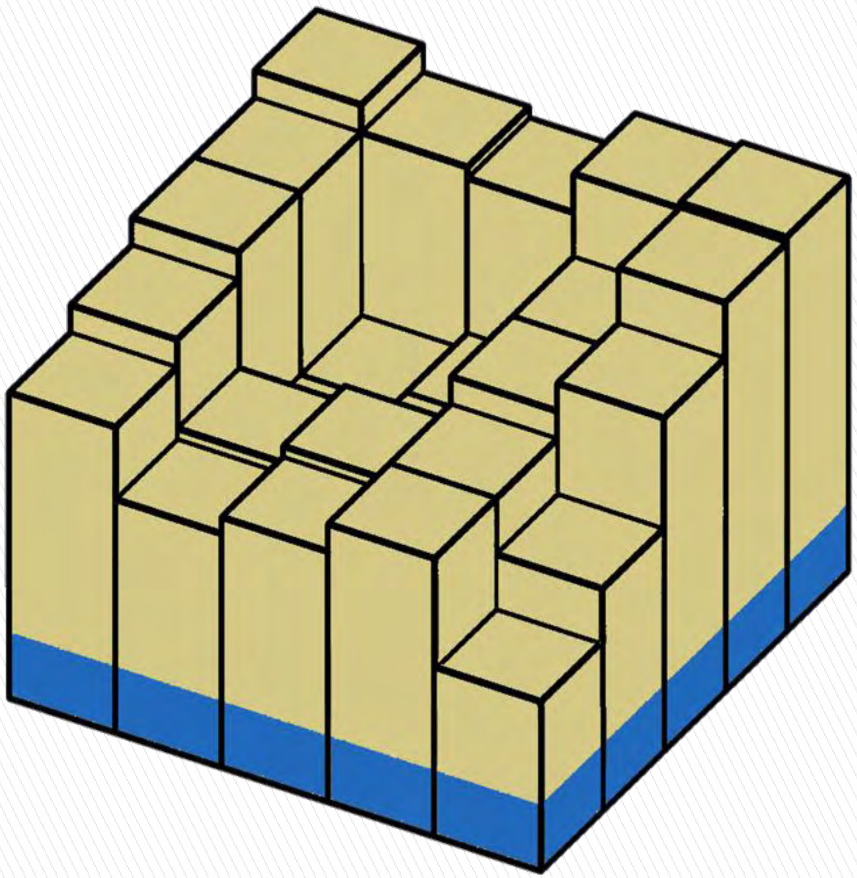
- The basic idea of *RWFlood* is:
  - supposing a terrain being flooded by water coming from outside and getting into the terrain through its boundary;

RWFlood description

- The basic idea of *RWFlood* is:
  - supposing a terrain being flooded by water coming from outside and getting into the terrain through its boundary;
  - the course of the water getting into the terrain will be the same as the water coming from rain and flowing downhill (that is, the flow direction);

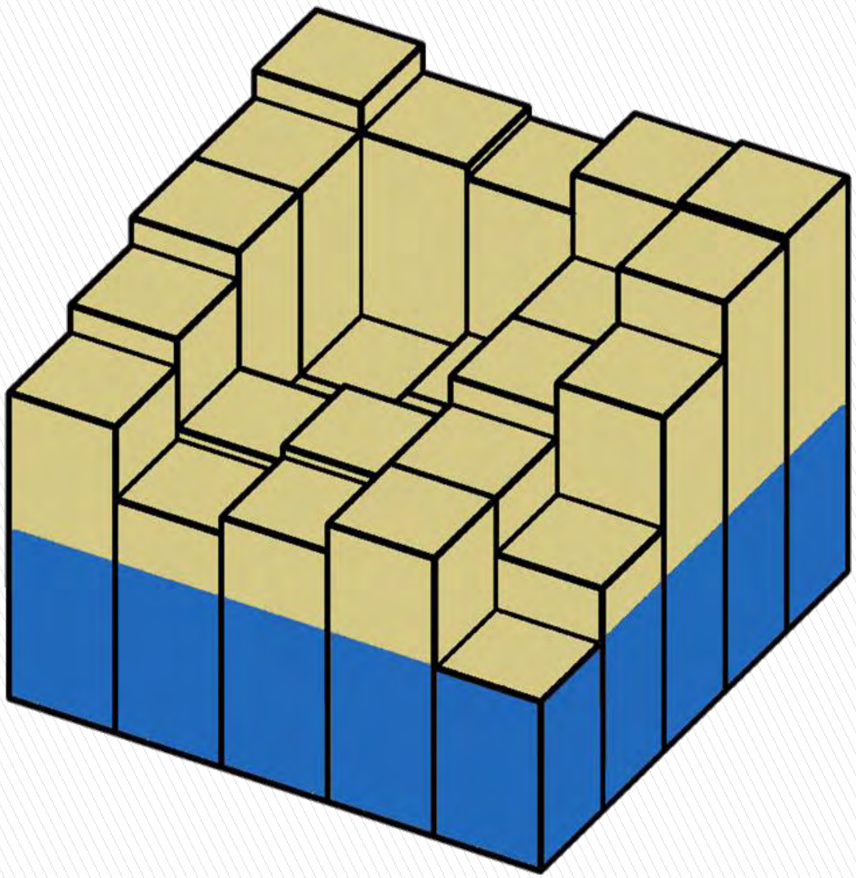
RWFlood description

- In other words, the idea is to suppose the terrain surrounded by water (as an island) and the flooding process is simulated raising the water level;



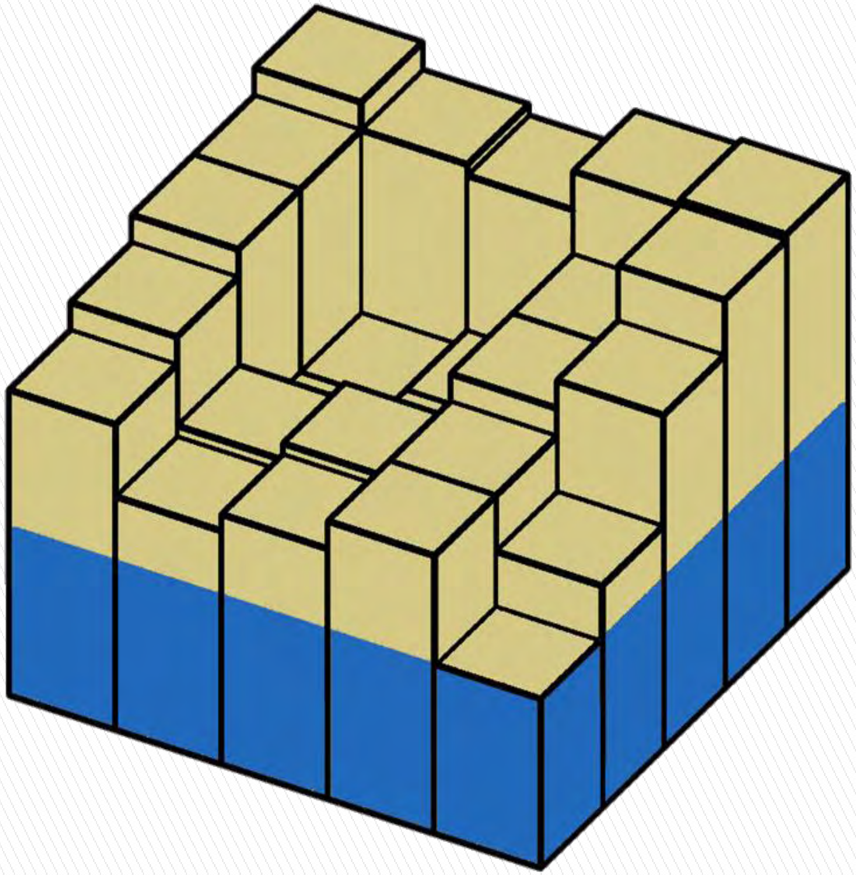
RWFlood description

- Initially, the water level is set to the elevation of the lowest cell in the terrain boundary;



RWFlood description

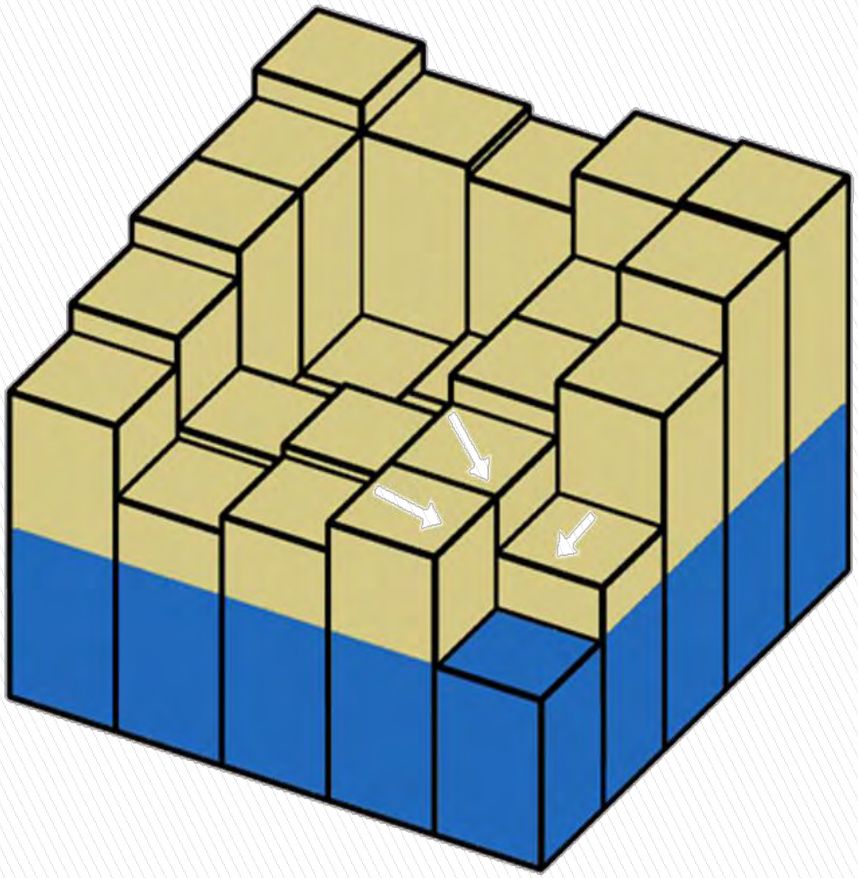
- Initially, the water level is set to the elevation of the lowest cell in the terrain boundary;
- Then, two actions are executed iteratively:
  - flooding a cell
  - raising the water level



RWFlood description

Flooding a cell *c*

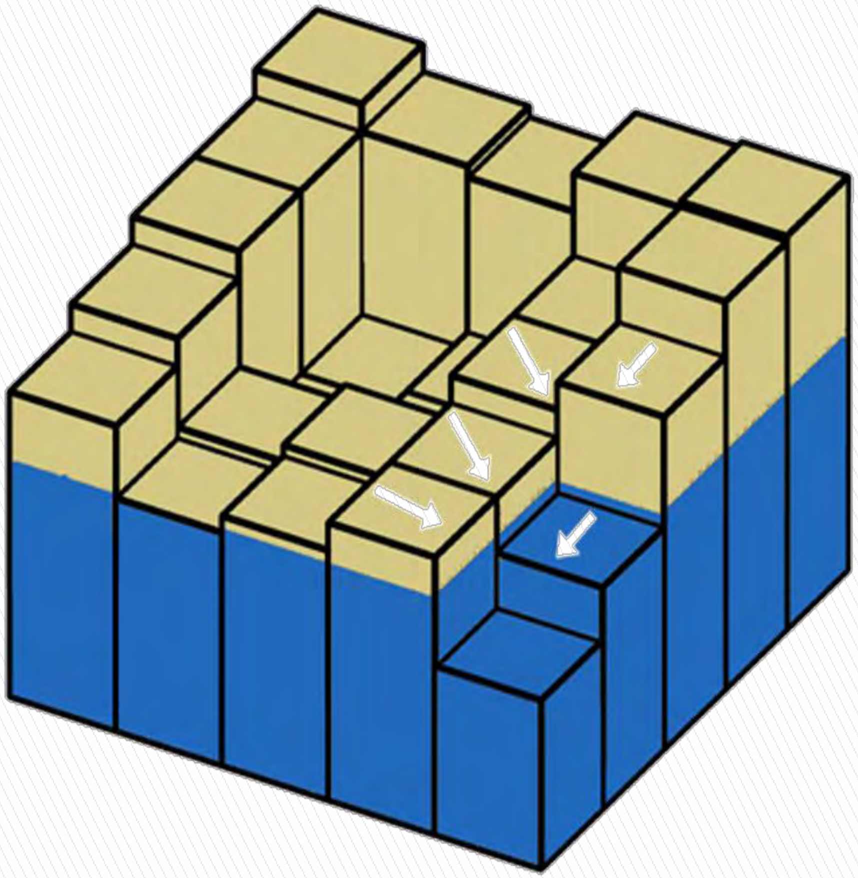
- For all cells *d* neighbors to *c* do:
  - if the elevation of *d* is smaller than the elevation of *c* then *d* is raised to the elevation of *c;*
  - the flow direction of *d* is set to the cell *c;*



RWFlood description

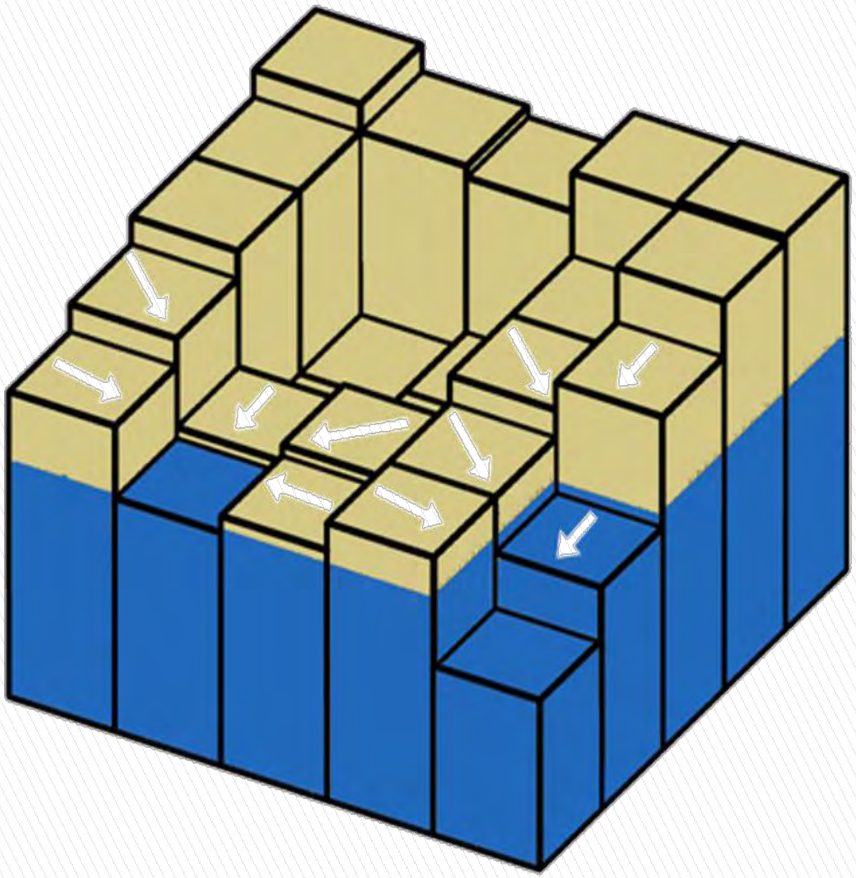
Raising the water level

- After flooding all cells with the same elevation as *c*, the water level is raised to the elevation of the lowest cell higher than *c*;



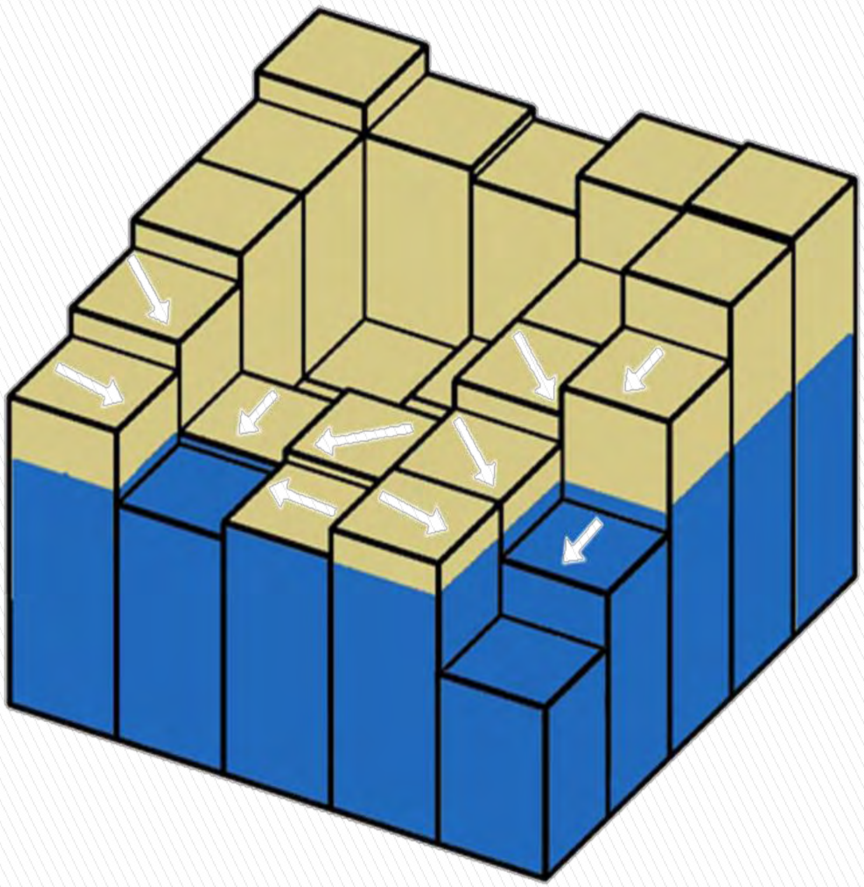
RWFlood description

- These cells are processed as previously and the level of the water is raised to the next level;



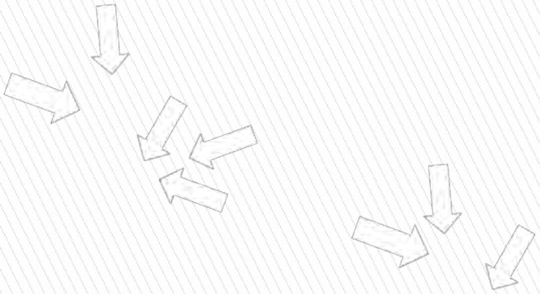
RWFlood description

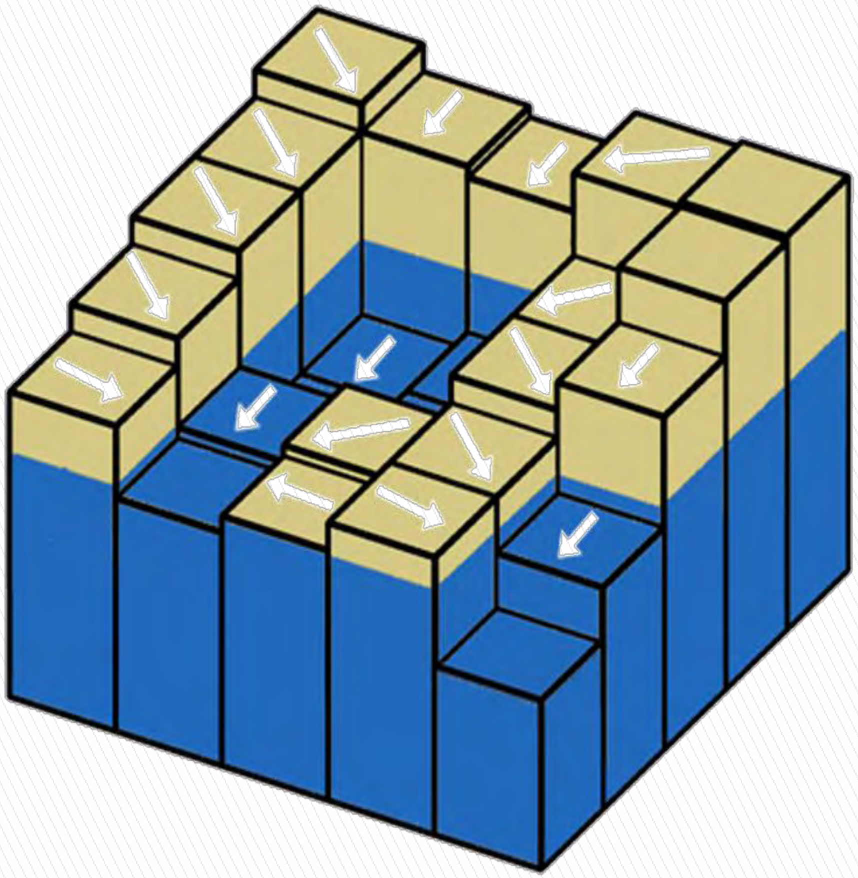
- Now, the cell to be processed has some neighbor cells whose elevation is smaller that the water level (a depression);



RWFlood description

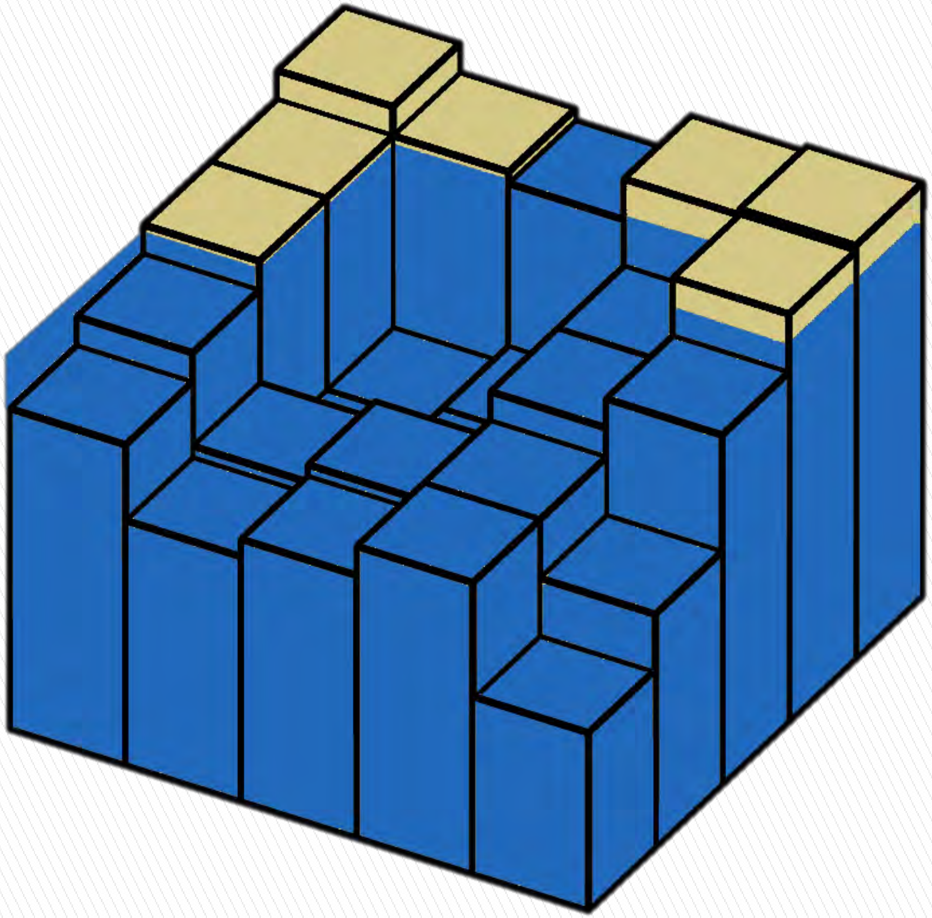
- The depression is filled;





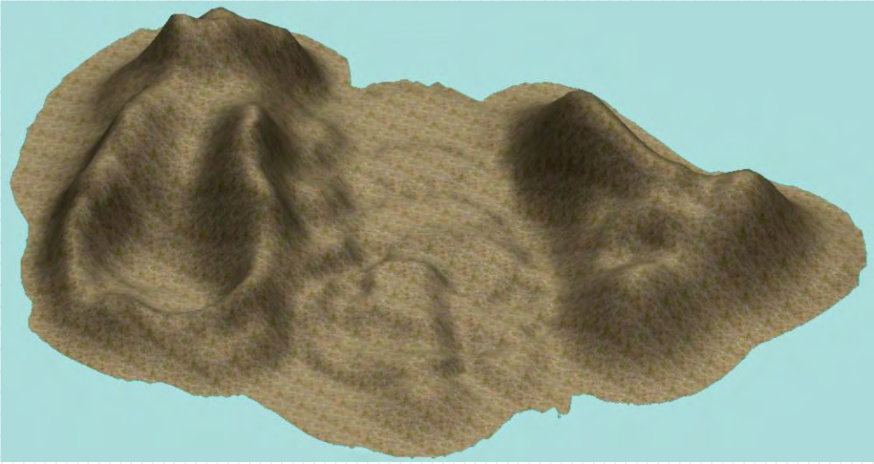
RWFlood description

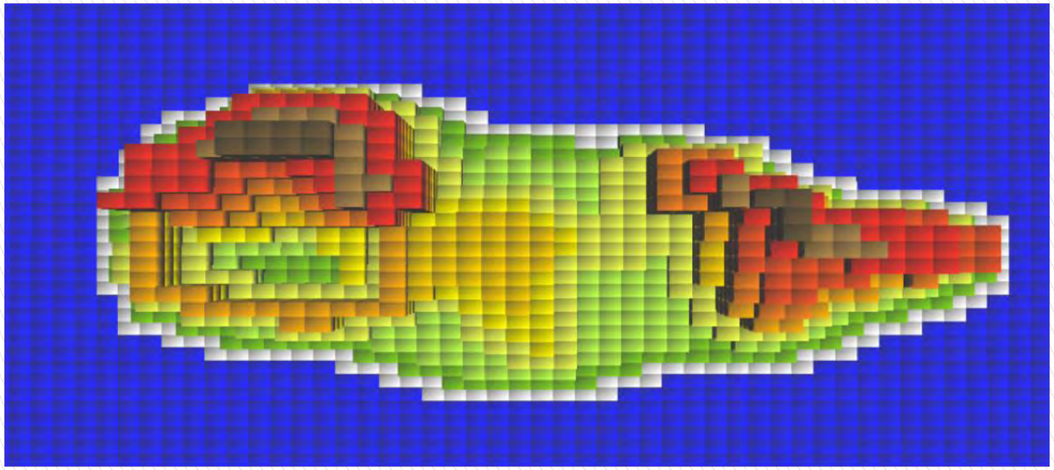
- Notice that the flooding process can creates islands;



RWFlood description

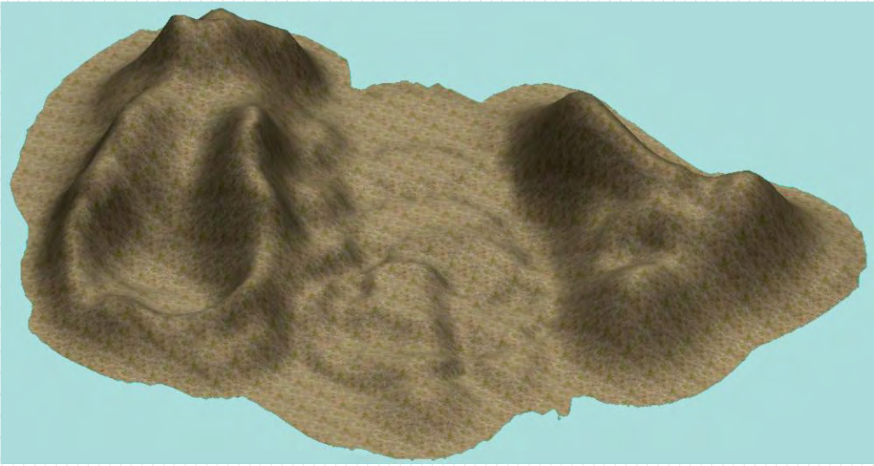
- Thus, the main idea of *RWFlood* is to store the cells in the boundary of flooded regions;

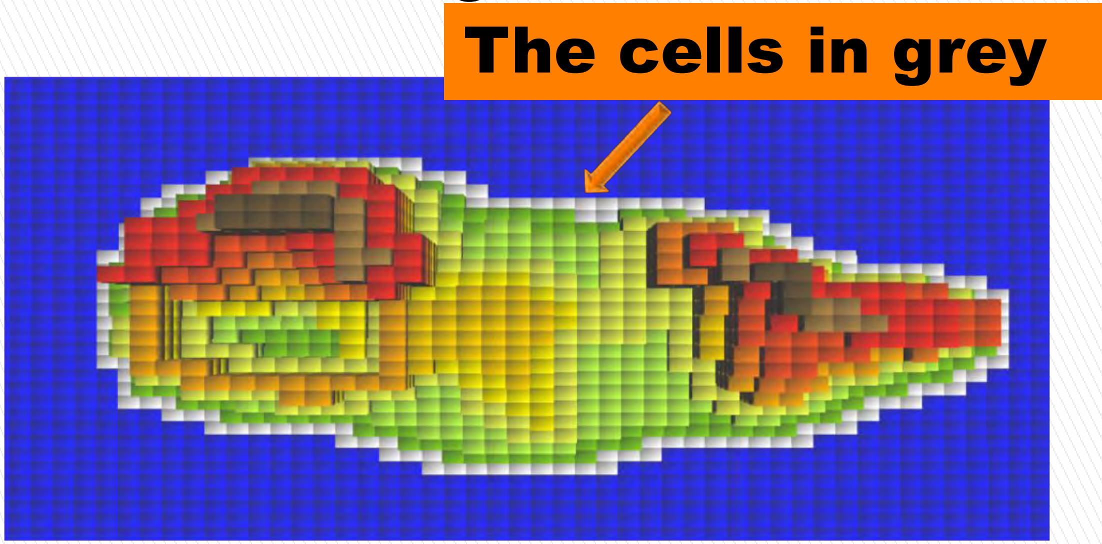




RWFlood description

- Thus, the main idea of *RWFlood* is to store the cells in the boundary of flooded regions;





- And, these cells are processed based on their elevation: from the lowest to the highest;

RWFlood description

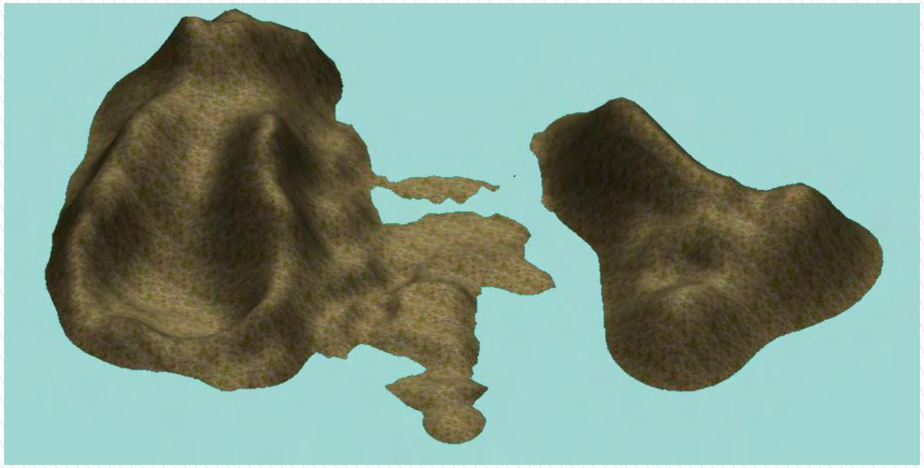
- When a cell *c* in the boundary is processed, this boundary "moves toward" the lowest neighbor cell of *c;*
- Which means the terrain matrix is accessed non-sequentially since the cells that are "neighbors" in the two-dimensional matrix representation may not be close in the memory;
- Thus, this process can be inefficient when the matrix is huge and is stored in external memory;

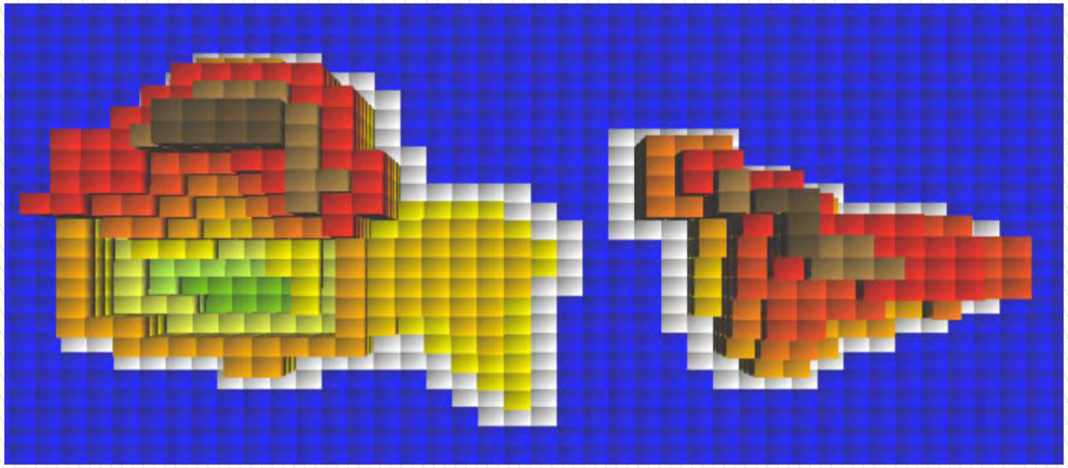
The *EMFlow algorithm*

- To reduce the number of disk accesses, we propose the *EMFlow* whose basic ideas are:
  - subdivide the terrain in smaller pieces which can be processed separately;
  - use a cache strategy to benefit from the spatial locality of reference present in the sequence of accesses;

The *EMFlow algorithm*

- <u>Terrain subdivision</u>: the flooding process can generate islands which can be processed separately;

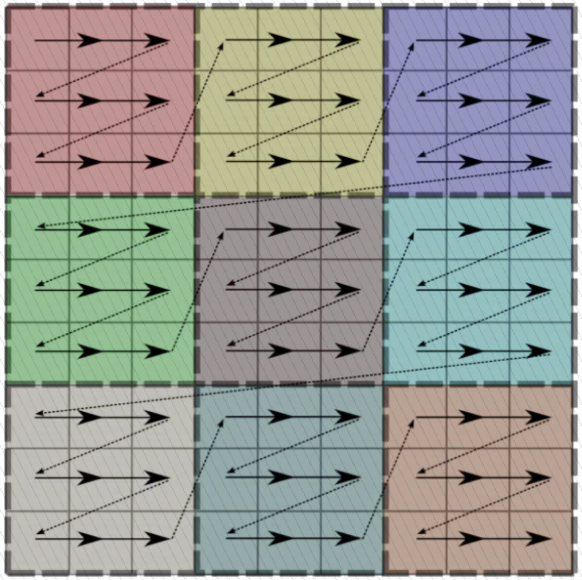




- An island is a maximal connected component of non processed (flooded) cells;

The *EMFlow algorithm*

- <u>Spatial locality of reference</u>: a special library, named *TiledMatrix*, is used to subdivide the matrix in squared blocks (of cells);



- Some blocks are stored in internal memory and are managed as a cache using the *LRU* policy;

Implementation details

- For performance improvements:
  - Islands identification: uses a lower resolution matrix;
  - Scheduling the islands processing: islands with a higher percentage of boundary cells stored in internal memory are processed first;
  - The islands boundary size: the number of islands which can be processed simultaneously is limited by a threshold;

Experimental results

- *EMFlow* was implemented in C++ and compiled with g++ 4.5.2;
- It was compared against *TerraFlow* and *r.watershed.seg* (both included in *GRASS)* and the tests were executed using an Intel Core 2 Duo 2,8 GHz machine running Ubuntu Linux 11.04 64 bits with a 5400 RPM SATA HD;
- Also, it was used different internal memory sizes: 1GB and 2GB to evaluate the algorithms performance in different scenarios;

Experimental results

- Processing time using 1GB of RAM

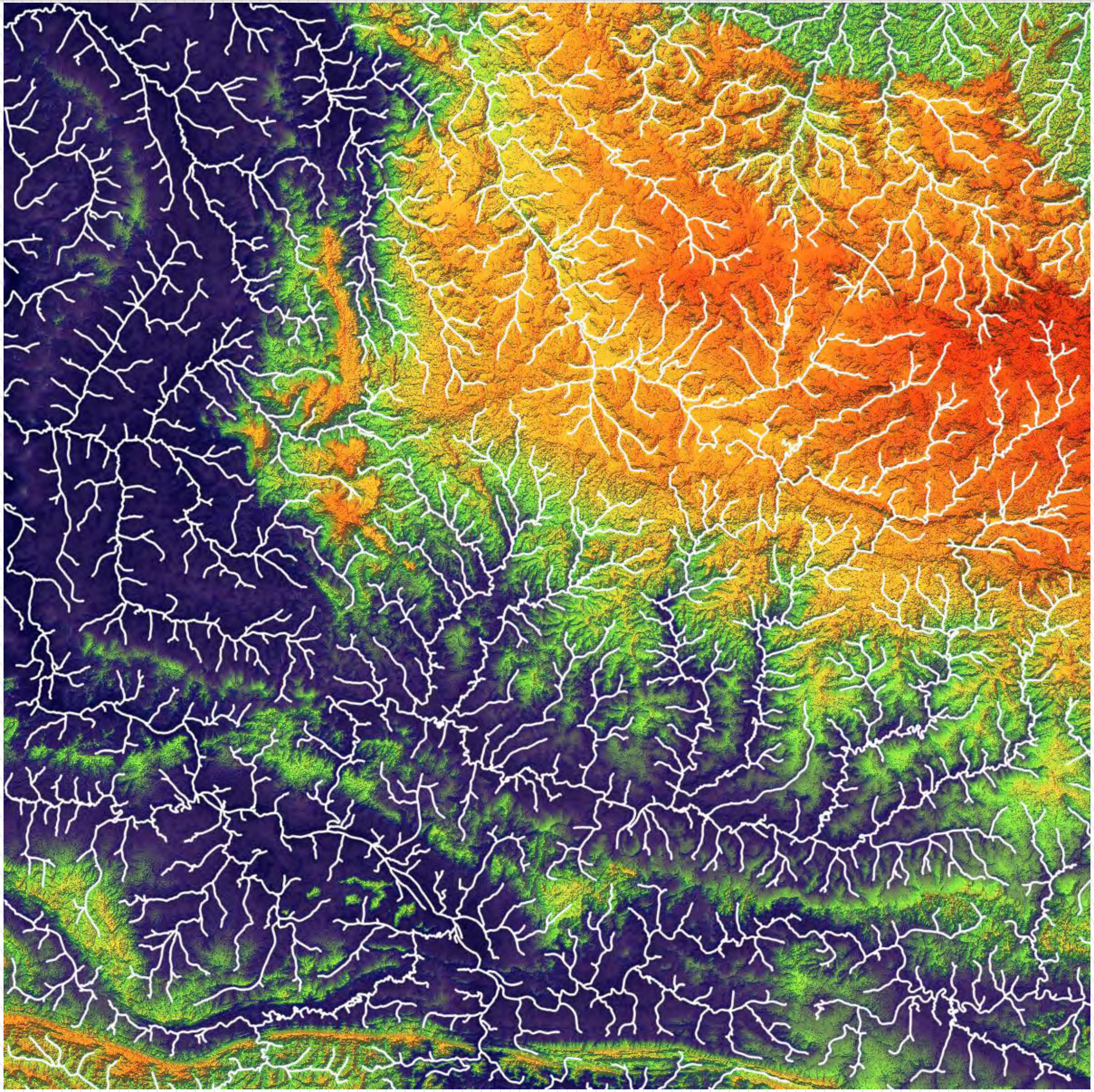
	Processing times (sec.)					
Terrain Size	Region R2			Region R3		
	EMFlow	*TerraFlow*	*r.wat.seg*	*EMFlow*	*TerraFlow*	*r.wat.seg*
1000^2	0,93	24,43	6,25	0,92	28,22	5,91
5000^2	18,80	661,37	622,66	19,11	907,50	508,90
10000^2	81,67	2329,71	25784,71	81,09	3358,42	55182,80
15000^2	251,14	7588,33	∞	248,39	9046,13	∞
20000^2	579,84	12937,30	∞	605,38	14404,76	∞
25000^2	980,14	22220,89	∞	1065,78	24974,77	∞
30000^2	1522,61	35408,11	∞	1890,35	41251,21	∞
40000^2	3055,39	67076,04	∞	4117,65	78056,28	∞
50000^2	7173,84	98221,64	∞	7618,78	110394,74	∞

Experimental results

- Processing time using 2GB of RAM

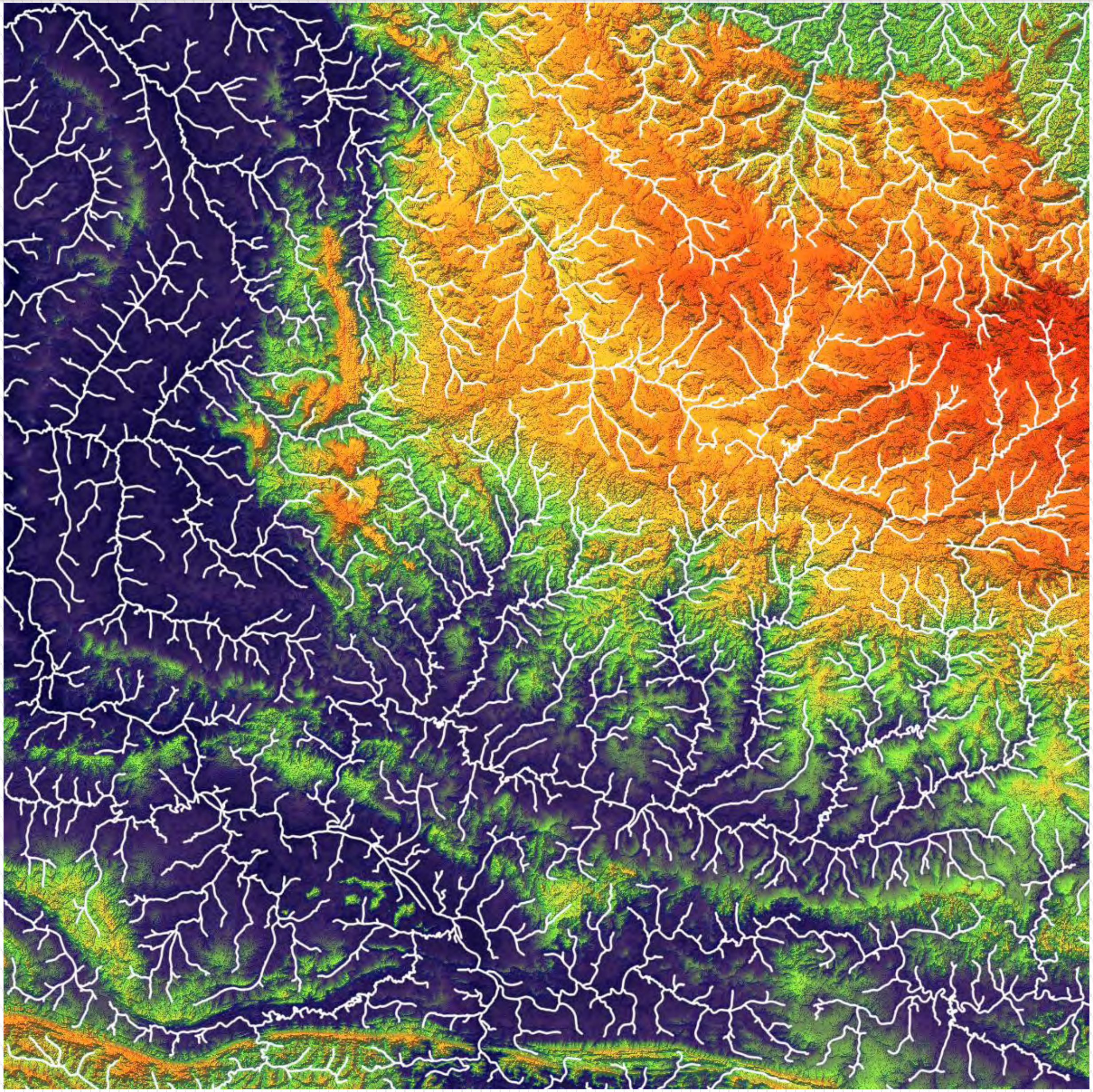
	Processing times (sec.)					
Terrain	Region R2			Region R3		
Size	*EMFlow*	*TerraFlow*	*r.wat.seg*	*EMFlow*	*TerraFlow*	*r.wat.seg*
1000^2	0,74	19,32	6,03	0,98	19,44	5,79
5000^2	20,02	400,84	630,60	19,98	442,97	513,88
10000^2	87,66	2251,66	5290,46	86,94	2552,93	3911,23
15000^2	209,02	5870,34	34252,23	202,36	6869,33	32518,89
20000^2	437,58	13066,63	∞	415,37	13873,60	∞
25000^2	776,98	19339,79	∞	764,86	22492,14	∞
30000^2	1179,31	30364,31	∞	1196,58	33337,07	∞
40000^2	2254,80	56421,36	∞	2162,17	59149,27	∞
50000^2	4011,72	82673,22	∞	3470,99	86670,30	∞

Experimental analysis



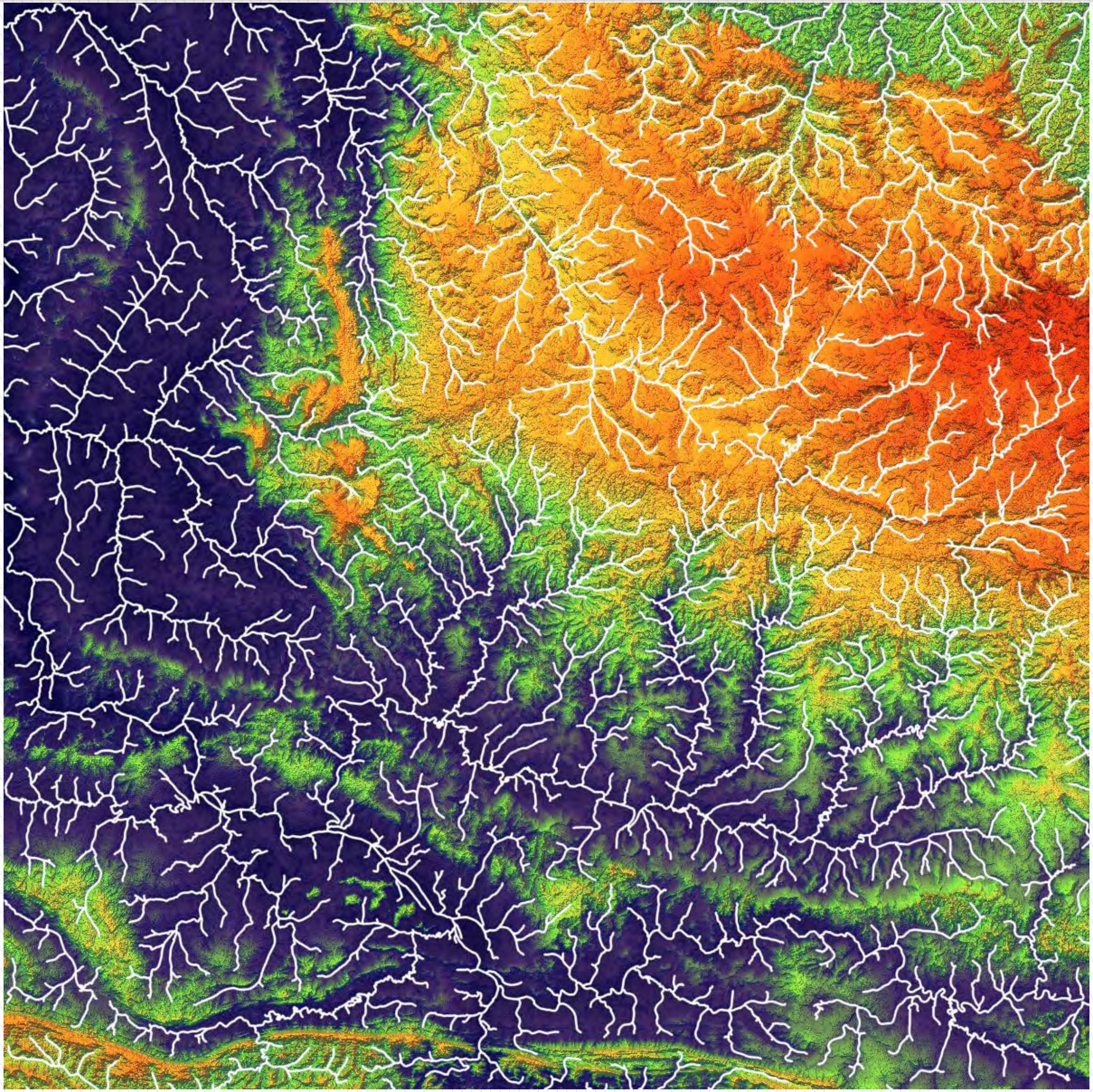
Drainage network on Tapajos basin computed by EMFlow

Experimental analysis



Drainage network on Tapajos basin computed by TerraFlow

Experimental analysis



Drainage network on Tapajos basin computed by r.watershed

Conclusions

- We developed a very fast and simple algorithm to compute the drainage network on huge terrains stored in external memory;
- The algorithm doesn´t require a preprocessing step to remove depressions and flat areas;
- It is linear in the number of cells in the terrain. That is, each terrain cell is read (and processed) only one time.

Conclusions

- *EMFlow* code, in C++, is available in:

  www.dpi.ufv.br/~marcus/EMFlowd

- Contact:

  marcus.ufv@gmail.com
  marcus@dpi.ufv.br

Acknowledgements

RENSSELAER POLYTECHNIC INSTITUTE • 1824 •

FAPEMIG

Fundação de Amparo à Pesquisa do Estado de Minas Gerais

CNPq

Conselho Nacional de Desenvolvimento Científico e Tecnológico

Universidade Federal de Viçosa

CAPES

FUNARBE

Fundaçao de Apoio à Universidade Federal de Viçosa

National Science Foundation

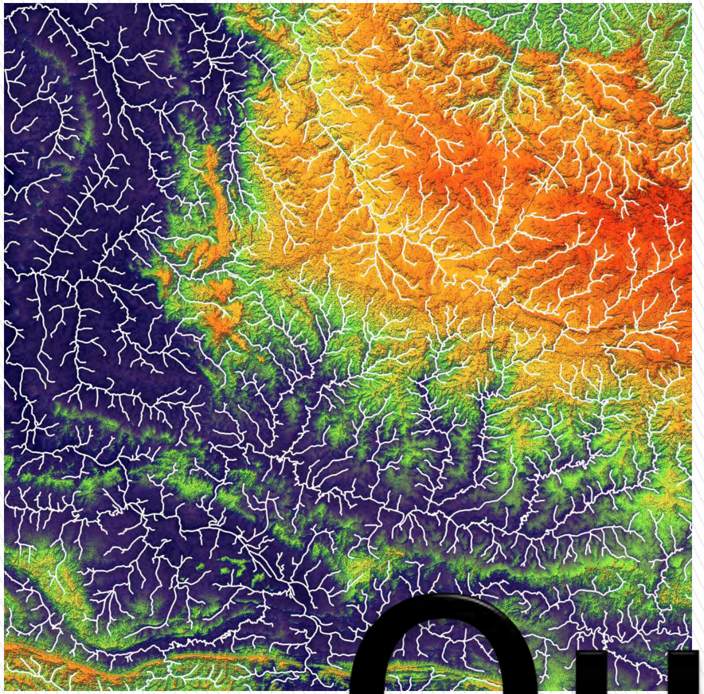
WHERE DISCOVERIES BEGIN

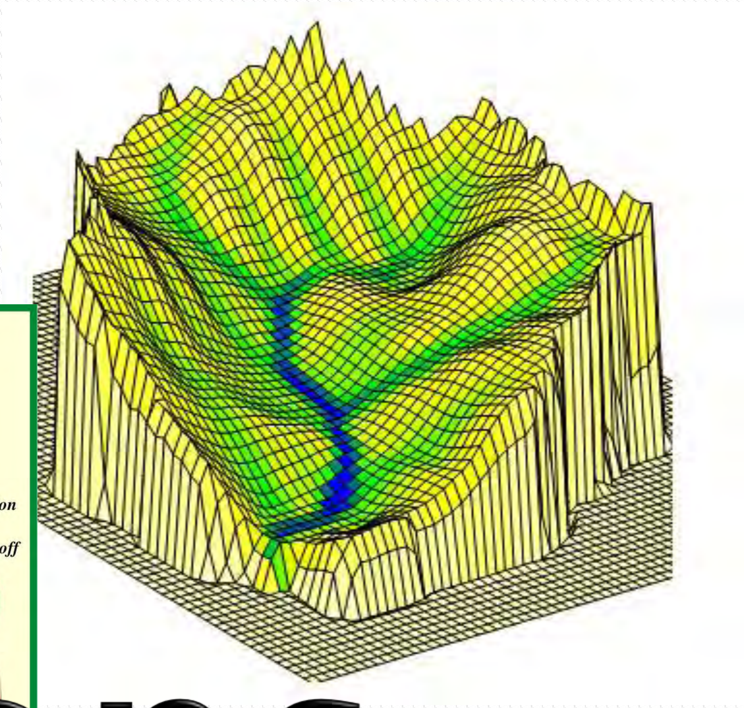
Gapso

TECNOLOGIA DA DECISÃO

Grants CMMI-0835762 and IIS-1117277

Questions






Inputs:
Precipitation
Solar Radiation
Components of a watershed model:
Processes:
Evapotranspiration
Layers:
Land Cover
Overland Runoff
Vegetation
Topography
Outputs:
Streamflow