Massively parallel Landscape-Evolution Modelling using General Purpose Graphical Processing Units

A.S. McGough, S. Liang, M. Rapoportas, D. Maddy, A. Trueman J. Wainwright, R. Grey and G. Kumar Vinod

19th December 2012

HiPC 2012 Pune, India

Newcastle University

Durham University

School of Computing Science

Digital Institute

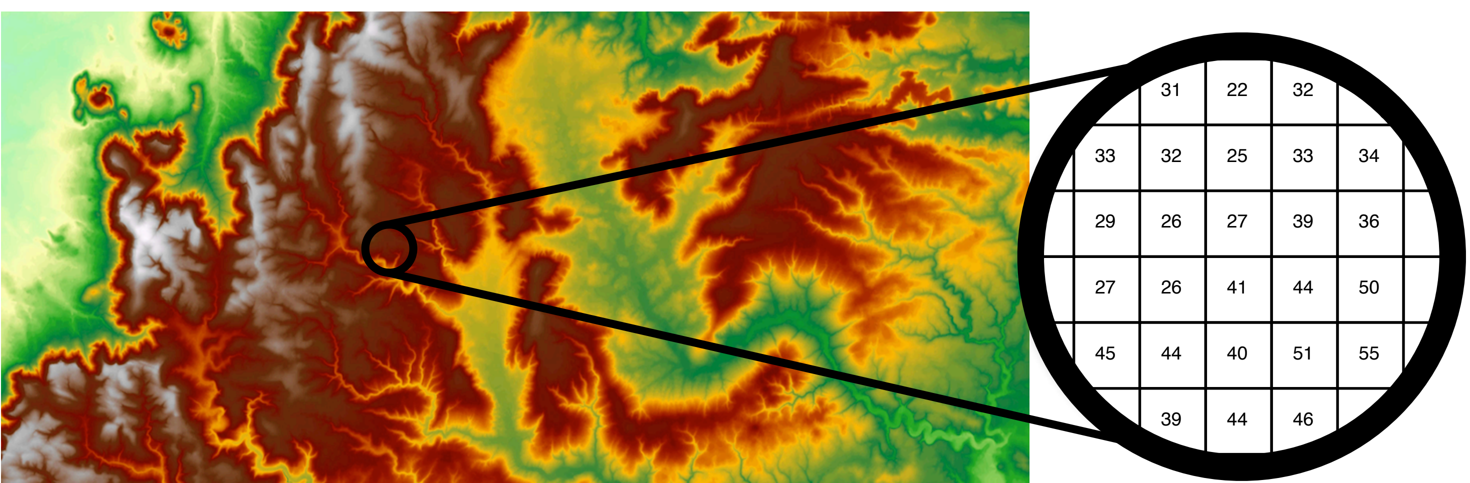
School of Geography, Politics and Sociology

Student Research Scholarships and Expeditions

Department of Geography

Landscape Evolution Modeling

- Landscapes change over time due to water
  - Physical and Chemical Weathering require water to break down material
  - Higher energy flowing water both Erodes and Transports material until decreasing energy conditions result in Deposition of material
- These processes take a long time
  - Many glacial-Interglacial Cycles
    - Cycles are ~100ka for last 800ka, prior to 800ka cycles were ~40ka in length
- We want to use retrodiction to work out how the landscape has changed
- Use a simulation to model how the landscape changes
  - 3D Landscape is descretized as a 2D grid (x,y) with cell values representing surface heights (z) derived from a digital elevation model (DEM)



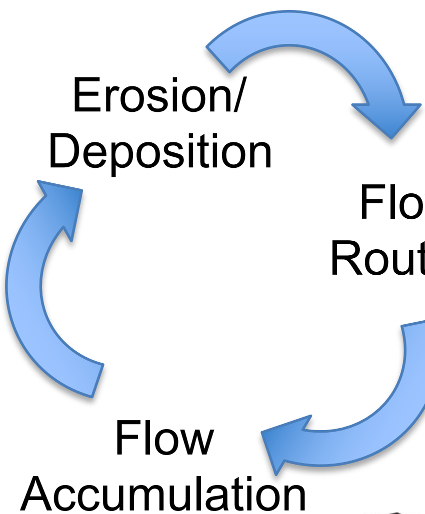
Landscape Evolution Modeling

Each iteration of the simulation:

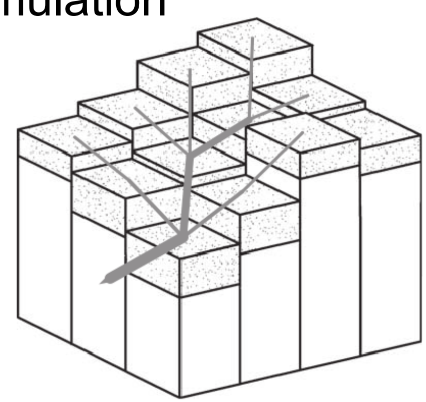
How much material will be removed?
How much material will be deposited?

7	8	5	9	6
4	6	7	8	4
5	8	9	10	9
7	10	7	8	7
9	8	4	6	5

Each step is 'fairly' fast…
But we want to do lots of them
120K to 1M years
On landscapes of 6-46M cells.
If we could simulate 1 year
in 1 minute this would take
83 – 694 days!
(assuming 1 year = 1 iteration,
may need more)



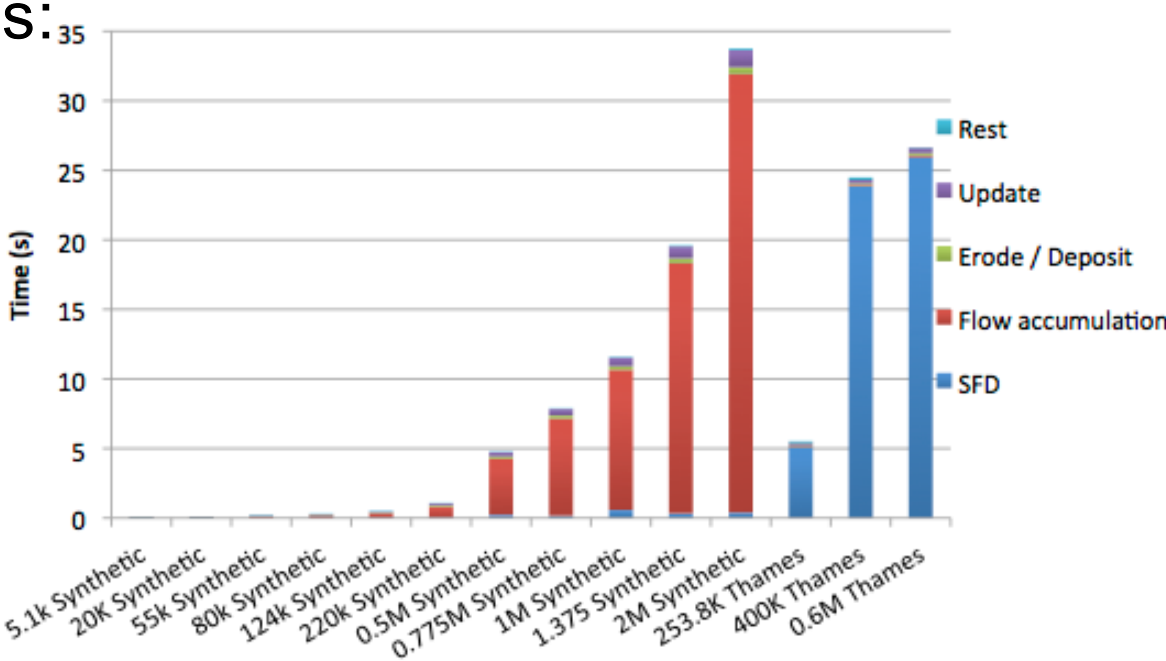
1	1	3	1	1
7	2	1	1	5
1	1	1	1	1
2	1	1	1	1
1	1	6	1	2



Current sequential version
is much slower than this…

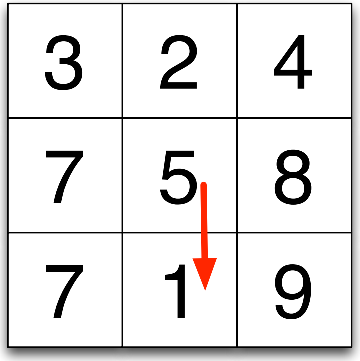
Execution analysis of Sequential LEM

- We started from an existing LEM
  - 51x100 cells took 72 hours
    - estimate for 25M cells 64,000 years
  - This was in-optimal code
    - Reduced execution time from 72 to 4.7 hours
    - 64,000 years down to 300 years
- But this is still not enough for our needs
- Performance Analysis:
- ~74% of time spent routing and accumulating
- Need orders of magnitude speedup
  - So look at these

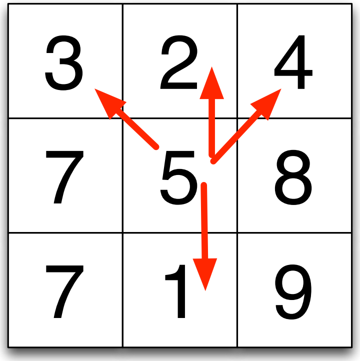


Parallel Flow Routing

- Each cell can be done independently of all others
  - SFD
    - 100% flow to the lowest neighbour



  - MFD
    - Flow is proportioned between all lower neighbours

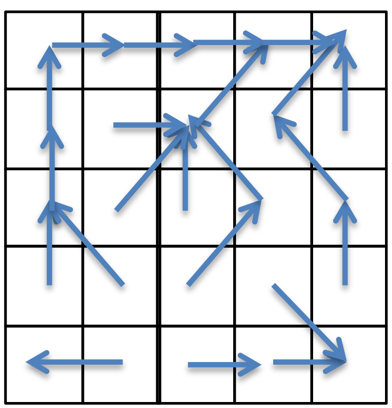


- Almost linear speed-up
  - Problems with code divergence
    - CUDA Warps split when code contains a fork

Parallel Accumulation: Correct Flow

- Iterate:
  - Do not compute a cell until it has no incorrect cells flowing into it
  - Sum all inputs and add self

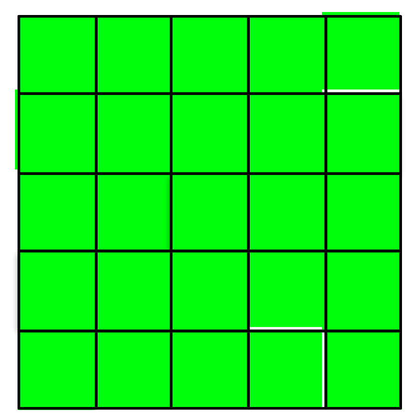
Flow Routing



Accumulation

5	6	7	14	19
4	1	6	3	1
3	1	1	2	2
1	1	1	1	1
2	1	1	2	4

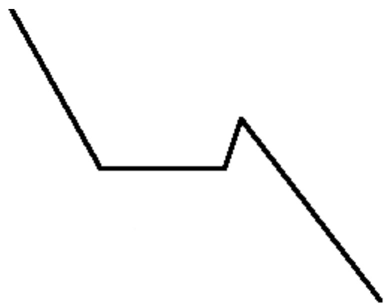
Correct

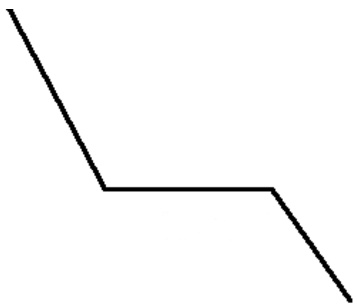


Cell values are not normally 1, but the value from the flow routing

Not the whole story…

- Sinks and Plateaus

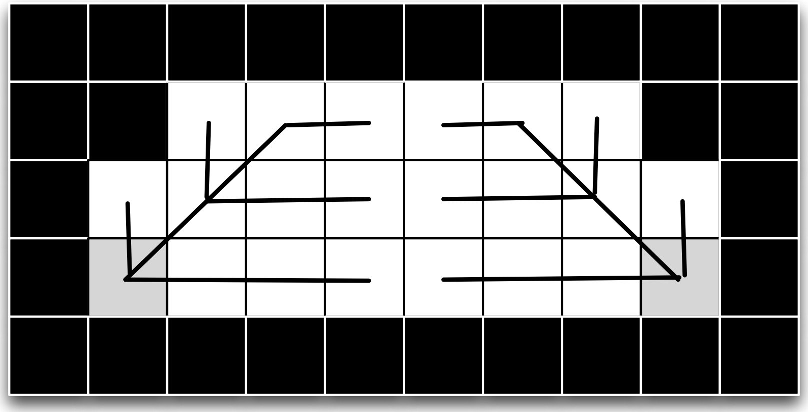


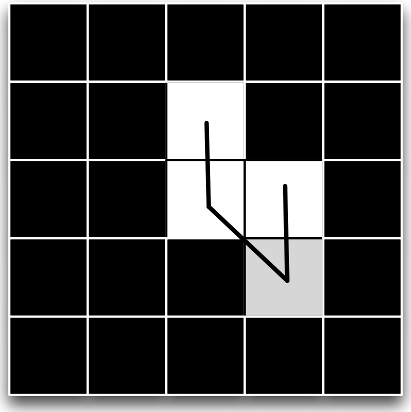


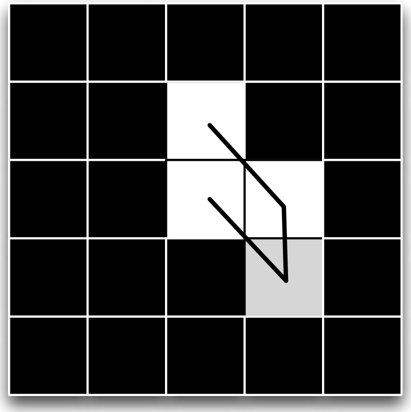
- Can’t work out flow routing on sinks and plateaus
- Need to ‘fake’ a flow routing
  - Fill a sink until it can flow out
  - Fake flow directions on a plateau to the outlet
- Single flow direction vs multiple flow direction
  - MFD is better but much more complex

Parallel Plateau routing

- Need to find the outflow of a plateau and flow all water to it
- A common solution is to use a breadth first search algorithm
  - Parallel implementation
  - Though result does look ‘unnatural’
  - Alternative patterns are possible – but acceptable
- We are investigating alternative solutions

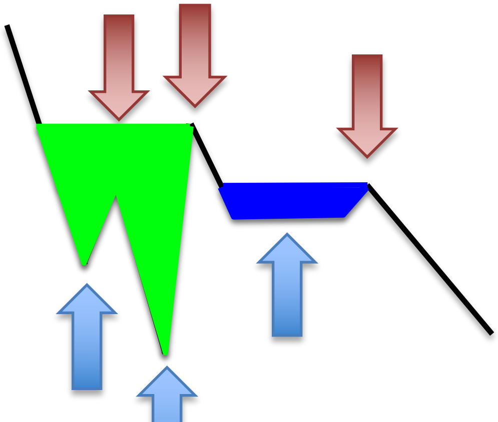






Sink filling

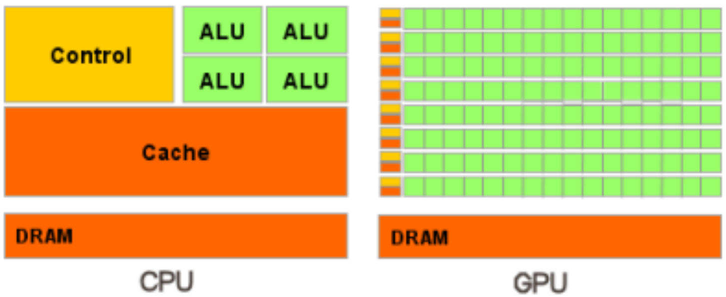
- Dealing with a single sink is (relatively) simple
  - Fill sink until we end up with a plateau
- But what if we have multiple nested sinks?
- Implemented parallel version of the sink filling algorithm proposed by Arger et al [2003]
  - Identify each sink (parallel)
  - Determine which cells flow into this sink - watershed (parallel)
  - Determine the lowest cell joining each pair of sinks (parallel/sequential)
  - Work out how high cells in each sink need to be raised to allow all cells to flow out of the DEM (sequential)
  - Fill all sink cells to this height (parallel)



GPGPU Solution

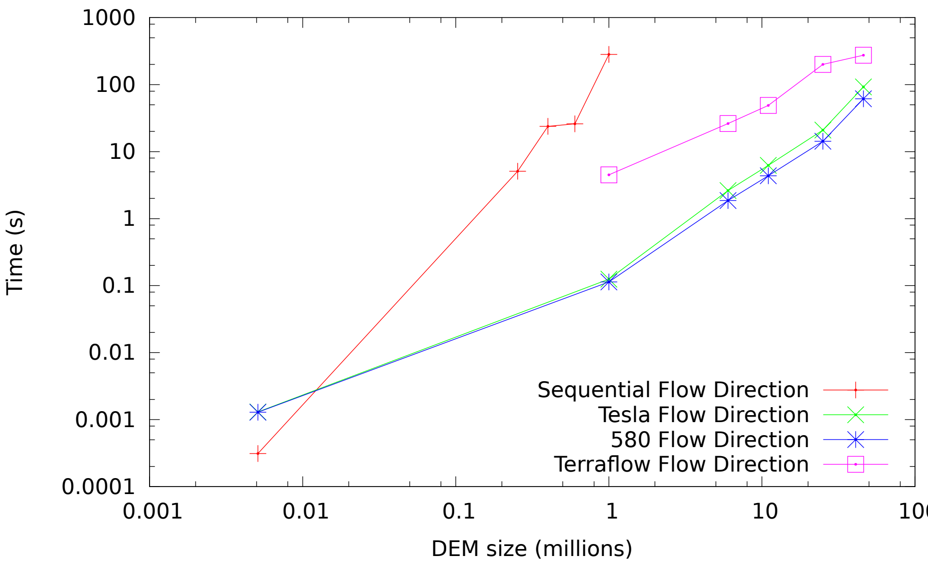
- Massively parallel version of the LEM
  - For Direction (including plateau and sinks) and Accumulation
    - Process has now been parallelized
  - on NVIDIA Fermi based graphics cards
    - Tesla C2050, GTX580
  - ~two orders of magnitude speedup over the optimized sequential code (up to 46m cells)
  - CUDA based

Card	Memory	Cores
GTX580	3GB	512
C2050	3GB	448

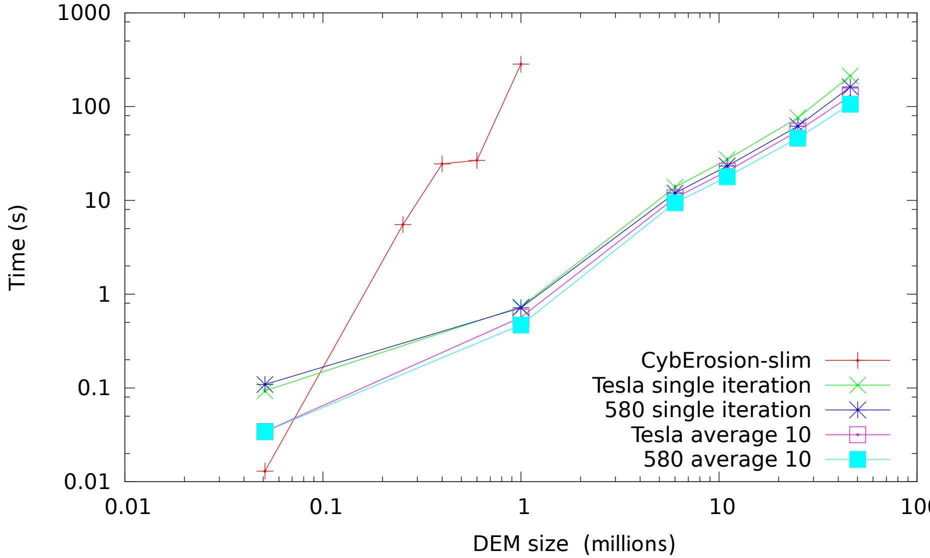


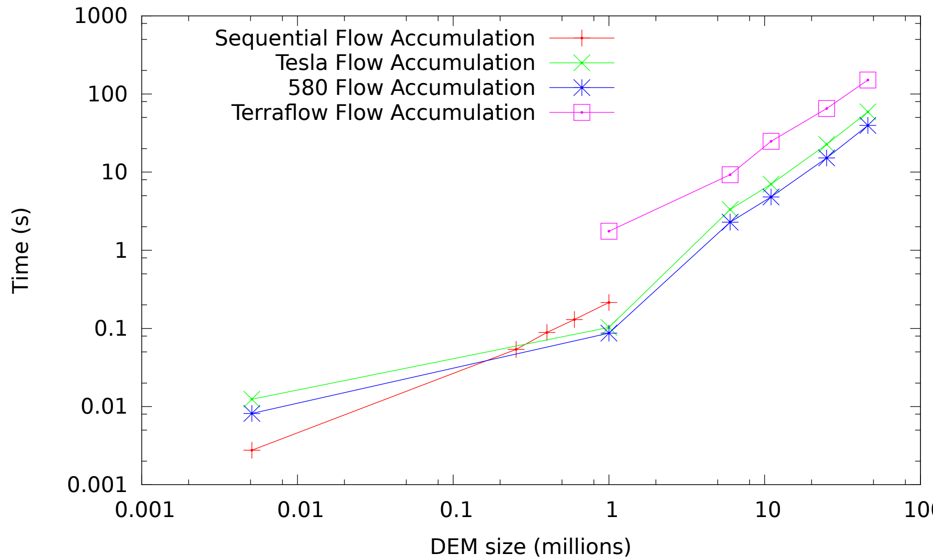
Results

- Overall performance



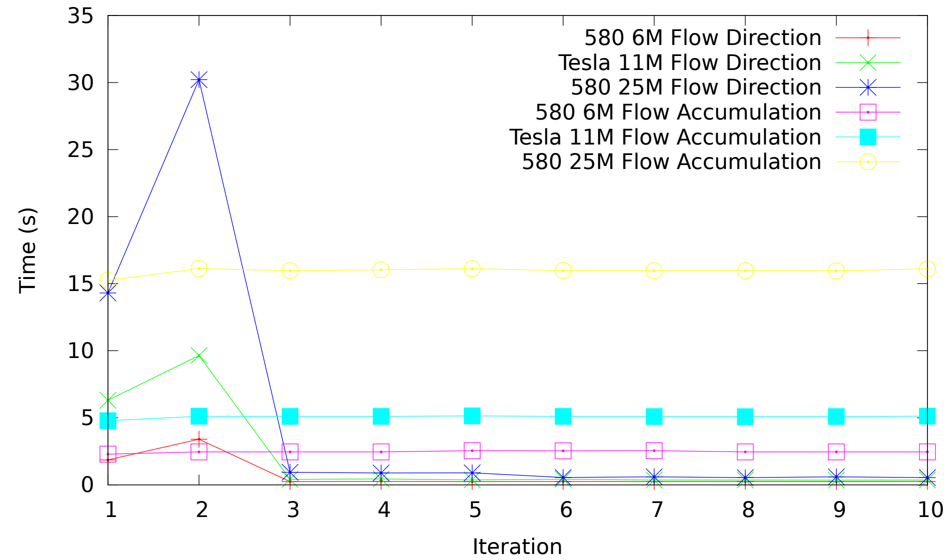
- Flow Direction
  - Inc sink & plateau
- Flow Accumulation

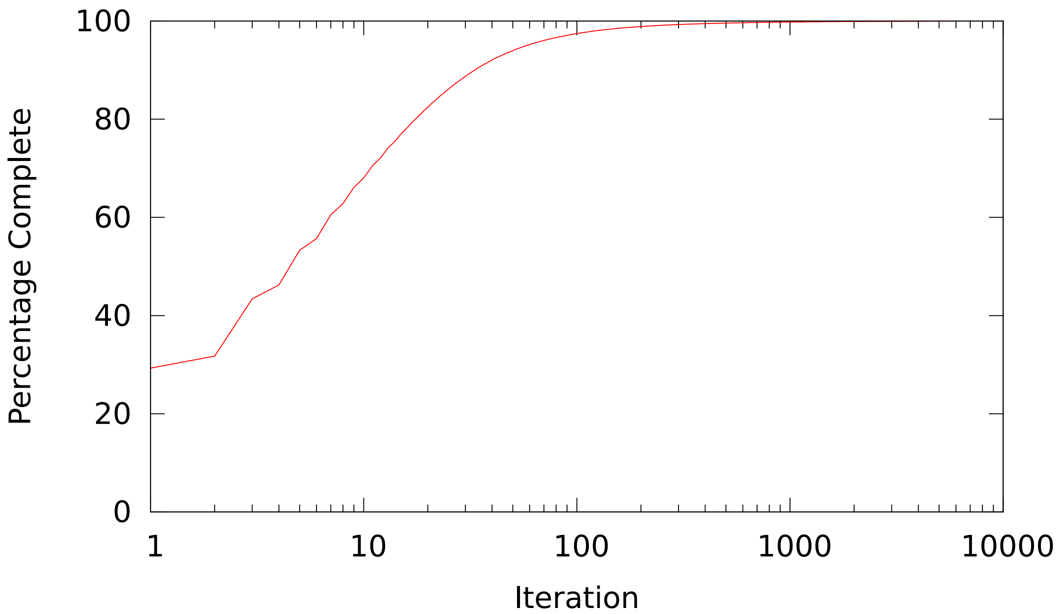




Results

- Comparison over iterations





- Correct flow completion profile

QUESTIONS?

A.S. McGough, S. Liang, M. Rapoportas,
D. Maddy, A. Trueman, J. Wainwright,
R. Grey, G. Kumar Vinod

stephen.mcgough@ncl.ac.uk

School of Computing Science

Student Research Scholarships and Expeditions

Department of Geography