MADALGO
CENTER FOR MASSIVE DATA ALGORITHMICS


Lars Arge
Aarhus University

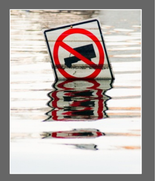

Mathias Rav
Aarhus University


Sarfraz Raza
Aarhus University


Morten Revsbæk
SCALGO

Danmarks Grundforskningsfond
Danish National Research Foundation

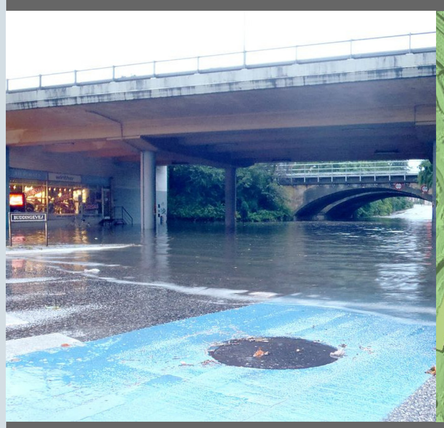
# I/O-efficient Event Based Depression Flood Risk

## Flash flood risk assessment

Our algorithm computes the areas of a terrain that are flooded in a given flash flood event by simulating how water flows on the terrain and fills up depressions.

Previous work only considered events where rain falls at a constant uniform rate on the entire terrain, whereas our algorithm allows different rainfall in each cell.



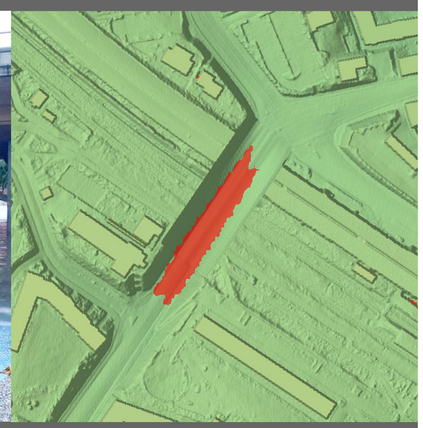


Photo credit: Finn Årup Nielsen, CC BY-SA, via Wikimedia Commons

Left: In 2011, Copenhagen was hit by a massive flash flood.

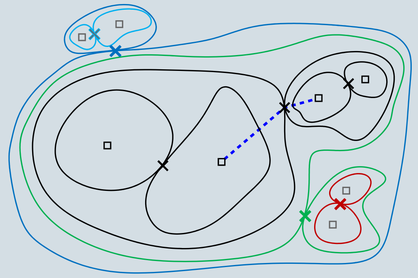
Right: The result of our algorithm on a typical 5-year event, red marking the cells that are flooded by ≥ 50 cm of rain water.

Our algorithm is **I/O-efficient**, meaning it is able to handle terrain models that are too large to fit in the RAM, which is the case for the 1.6 m resolution model of Denmark. Our algorithm can process the 1.6 m model in 2 hours on a standard workstation, which shows that our algorithm can be useful in near-term flood risk assessment.

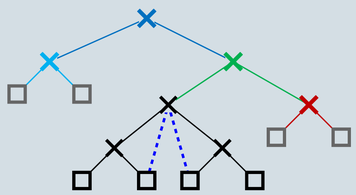
## Merge Tree

We assume that water only flows on the surface of the terrain. When a depression fills up, remaining water flows across a **saddle cell** of the terrain into a neighboring depression.

The **merge tree** represents how depressions are nested.



**Sample terrain**



**Merge tree**

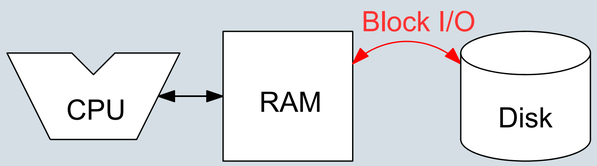
$X$ = number of sinks in terrain
$H$ = height of merge tree

The **topological complexity** is $X$, the number of sinks, which is the number of leaves of the merge tree.

The performance of our algorithm depends on the topological complexity of the terrain.

## External Memory Model

We analyze our algorithms in the external memory model. The cost of an algorithm is the number of **I/Os** it performs.




$N$ = # of items in input
$B$ = # of items per disk block
$M$ = # of items that fit in main memory

Scan($N$) = $N/B$, the cost of reading $N$ items sequentially
Sort($N$) = $N/B \cdot \log_{M/B} (N/B)$, the cost of sorting $N$ items

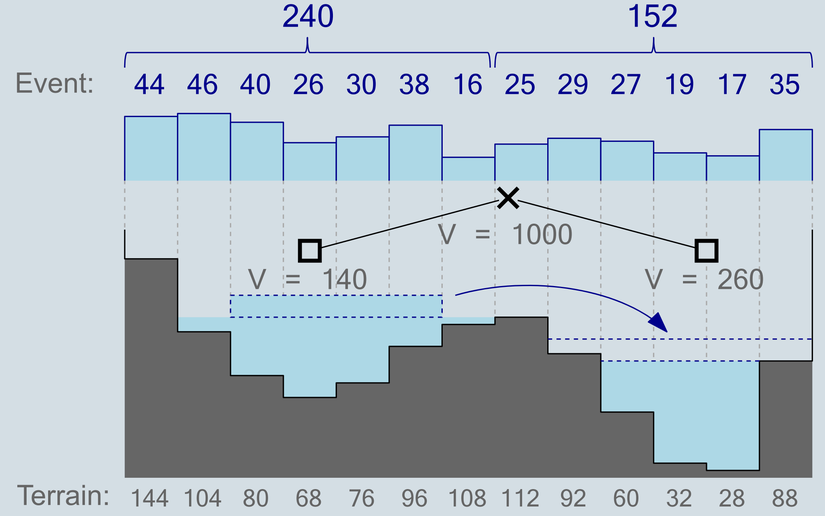
## Algorithm

Previous work has shown how to compute the **watersheds and merge tree** of the terrain in $O$(Sort($N$)) I/Os. This computation is independent of the flash flood event and only needs to be done once.

The key insight in the algorithm is that we in $O$(Scan($X$)) I/Os can compute the **excess volume** of each depression, which is **positive** if it is full and **negative** if it has capacity for more rain.

## Example in 1 Dimension

In the following example, the two elementary depressions have volumes 140 and 260, and they receive rain amounts 240 and 152 respectively.




First, our algorithm computes the excess volumes by subtracting the rain amounts (240 and 152) from the depression volumes (140 and 260).

Since the left depression has a positive excess of +100, and the right depression has a negative excess of −108, the flash flood event **floods the left depression completely**, and the right depression receives an **extra** 100 volume of rain.

In Scan($N$) I/Os we then compute for each cell if it is flooded.