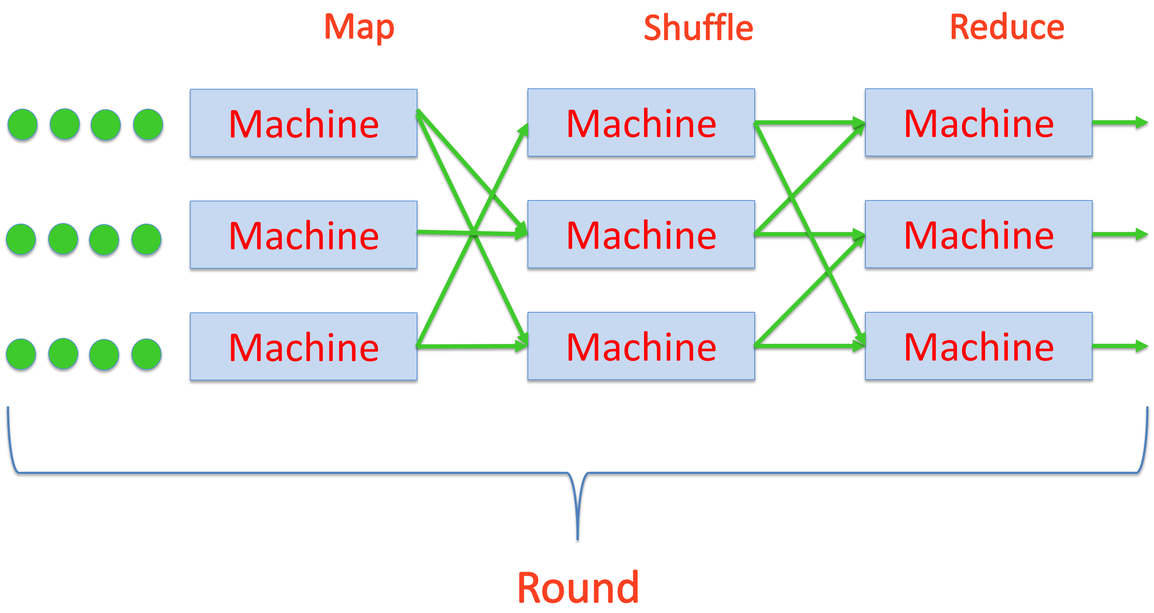
The weight of an MST in the MapReduce model

(Łącki, Mądry, Mitrović, Onak, and Sankowski.
Done while visiting University of Warsaw.)

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

MapReduce model



- N items
- Each machine has $\Theta(N^\alpha)$ memory, usually $0 < \alpha < 1$
- Goal: solve a given problem in **as few** rounds **as possible**

Problem: Estimate the MST weight

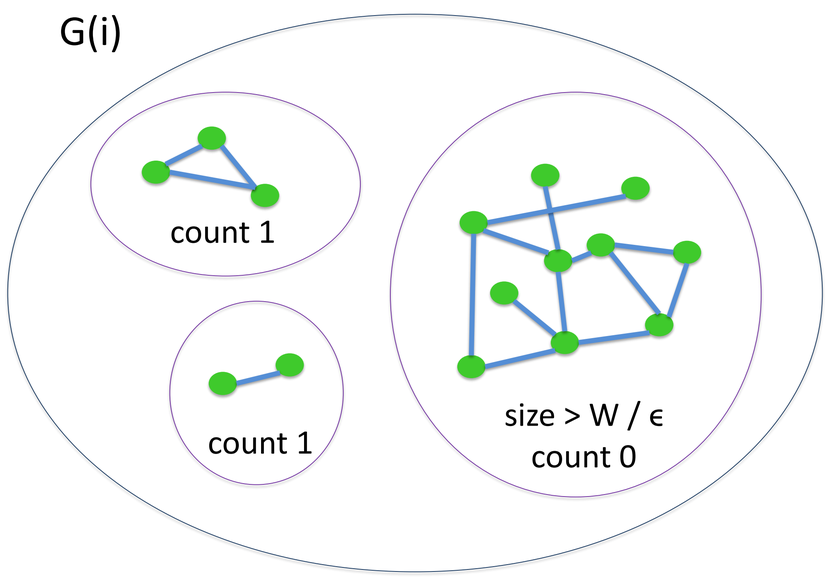
Input: G = (V, E, w), w(e) in {1, …, W}
Output: (1 + ϵ)-approximation to the MST weight

Algorithm (based on [3]):

- G(i) = the subgraph of G restricted to edge-weights < i
- C(i) = # of connected components in G(i)
- w(MST) = $\sum_{i=1}^{W}(C(i) - 1)$
- Approximate C(i) to get approximate w(MST)

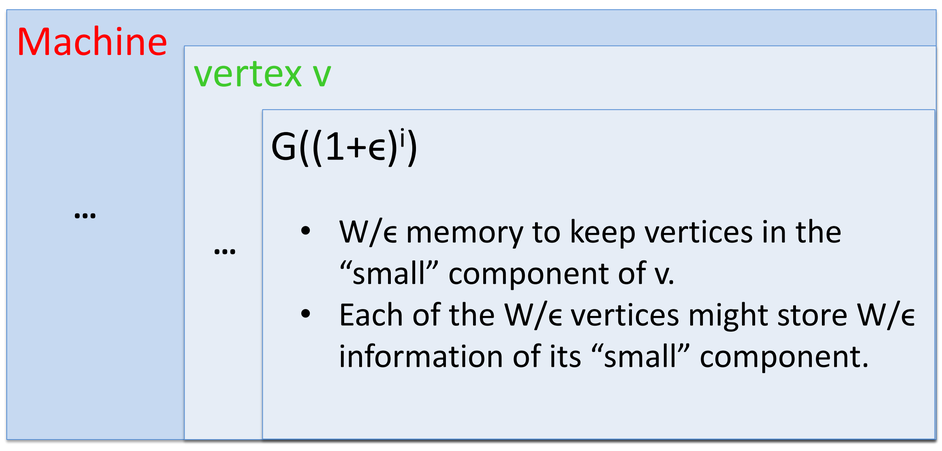
Analysis

1. A contraction based algorithm to estimate C(i).
2. Ignore large components – count only “small” ones.



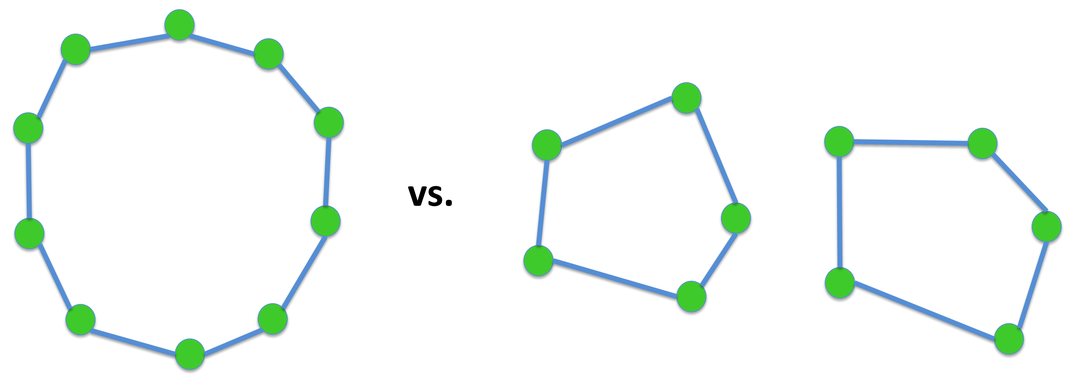
3. A small component can be explored in O(log(W / ϵ)) rounds.
4. The number of large components <= ϵ n / W.
5. Each large component incurs error of at most W => total error of at most W * ϵ n / W = ϵ n. We have w(MST) >= n - 1.
6. Compute in parallel for all G(i).
7. Use [4] to organize communication.

Memory:



Can we do better?

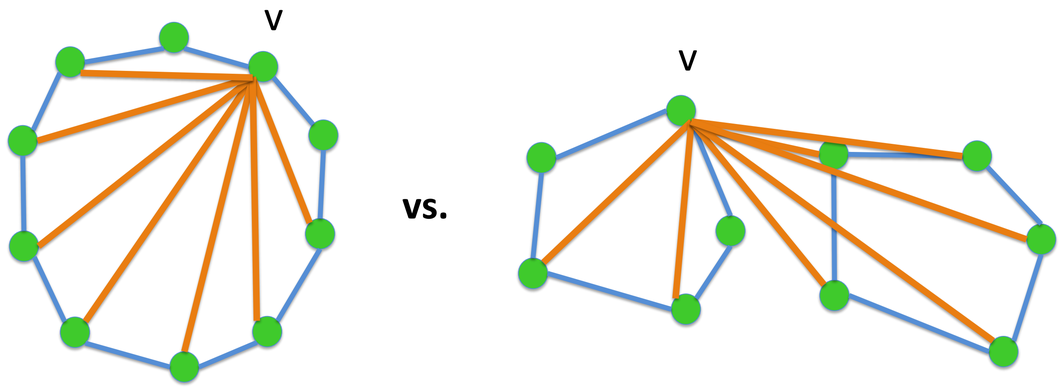
Sparse Connectivity Conjecture



- Can be decided in log N rounds and space $\Theta(N^\alpha)$, $\alpha < 1$, per machine. N is the number of vertices.
- Widely believed that a super-constant number of rounds is needed.

Reduce to MST

- Pick a vertex v.
- For every u, if there is no edge {u, v}, add a very **heavy** edge {u, v}, e.g. of weight N^2.



- A (1 + ϵ)-approximate weight of an MST can be used to answer Sparse Connectivity.
- So, if the conjecture is true, we should not hope for an algorithm that estimates the weight of an MST in O(log(1 / ϵ)) rounds.
- Assuming the conjecture, if each machine has sublinear memory, the number of rounds needed to estimate the weight of an MST has to even mildly depend on W.

References

[1] H. Karloff, S. Suri, and S. Vassilvitskii, *A model of computation for MapReduce*, 2010
[2] S. Lattanzi, B. Moseley, S. Suri, and S. Vassilvitskii, *Filtering: a method for solving graph problems in mapreduce*, 2011
[3] B. Chazelle, R. Rubinfeld, and L. Trevisan, *Approximating the minimum spanning tree weight in sublinear time*, 2005
[4] M. T. Goodrich, N. Sitchinava, and Q. Zhang, *Sorting, searching, and simulation in the mapreduce framework*, 2011
[5] K. Onak, (Slides) *Parallel Algorithms for Graphs on a Very Large Number of Nodes*, 2015, http://grigory.us/mpc-slides/krzysztof.pdf

slobodan.mitrovic@epfl.ch