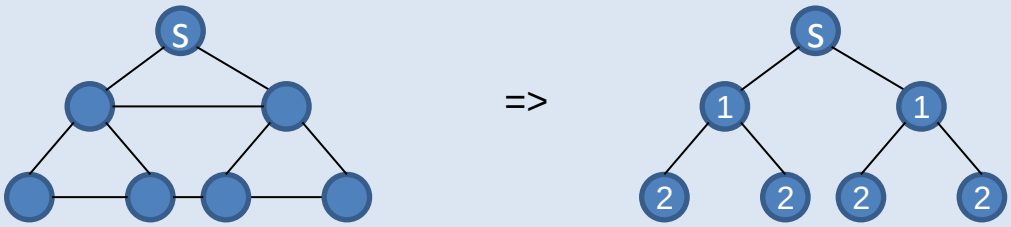
I/O efficient External-Memory Diameter Approximation
The accurate approximation

Diameter Approximation with Breadth-First-Search (BFS)

The diameter D of a graph is bounded by the height H of its BFS-tree as follows:

$$H \leq D \leq 2 \cdot H$$

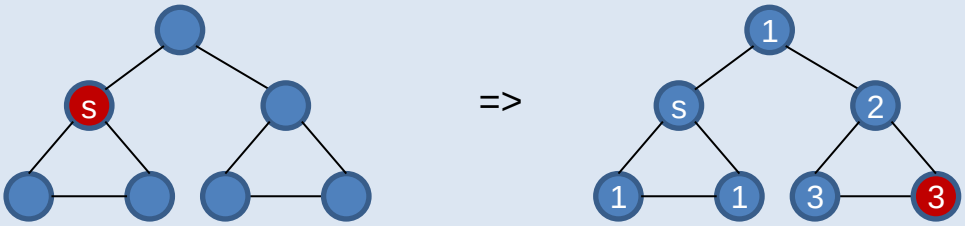
However in worst case the I/O complexity of BFS is $O(N/\sqrt{B})$ I/Os for sparse graphs. This is very slow compared to other simple graph algorithms as Spanning Trees which can be solved in sorting complexity [$O(\frac{N}{B} \cdot \log_{\frac{M}{B}}(\frac{N}{B}))$ I/Os] w. h. p.



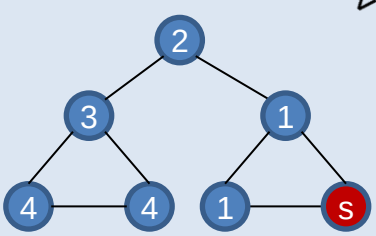
2 ≤ D ≤ 4

Double sweep lower bound

Improve the lower bound of the BFS heuristic: Compute a BFS from an arbitrary source and then compute a second BFS from a vertex with farthest distance to the first source. In practice the result is often very tight and for some graph classes the exact diameter can be computed with this heuristic. Nevertheless, for some graph classes the bound cannot be improved by double sweep lower bound with high probability.



3 ≤ D ≤ 6



4 ≤ D ≤ 6

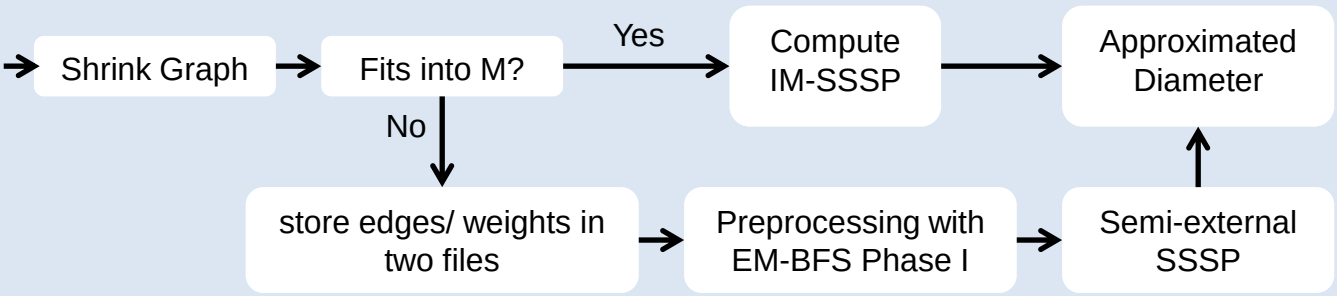
M. Sc. David Veith Goethe-Universität Frankfurt am Main, dveith@cs.uni-frankfurt.de

I/O efficient External-Memory Diameter Approximation
The fast approximation – Part I

Parallel Cluster Growing Algorithm

Shrink input with parallel cluster growing algorithm and compute a Single Source Shortest Path on the condensed Graph. Expected approximation error: $O\left(\sqrt{k} \cdot \log(k)\right)$

I/O-complexity: $O\left(k \cdot scan(N) + sort(N) + N \cdot \sqrt{\log(k)/(k \cdot B)}\right)$



Results

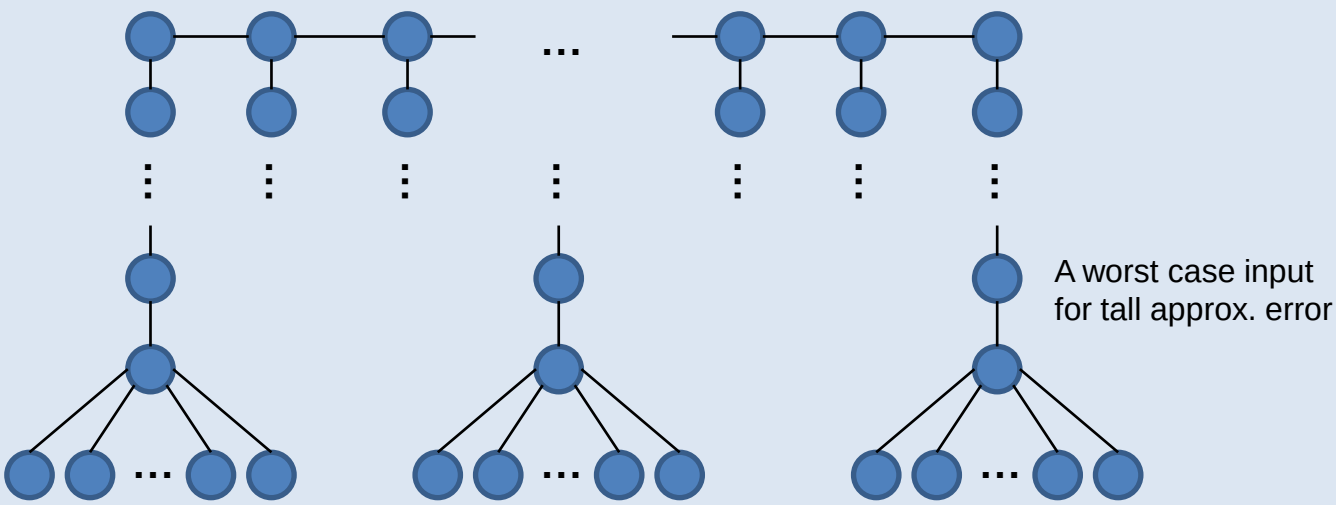
For input with a small diameter the clustering can be computed fast no matter the size of *k*. However, the approximation error is very close to the expected error.

For graphs with larger diameter as $\Theta(\sqrt{n})$ or $\Theta(n)$ the approximation result is tight.

However, the condensed $\Theta(\sqrt{n})$-graph produces a huge set of edges even if the number of vertices is shrunken to 1% of the original graph. Therefore choose a small value for *k*.

For graphs with a diameter of $\Theta(n)$ the clustering can become slow using a small *k*.

\+ Real world graphs usually have a small diameter
– Not optimal for each graph class with a fixed value for *k*



M. Sc. David Veith Goethe-Universität Frankfurt am Main, dveith@cs.uni-frankfurt.de

I/O efficient External-Memory Diameter Approximation
The fast approximation – Part II

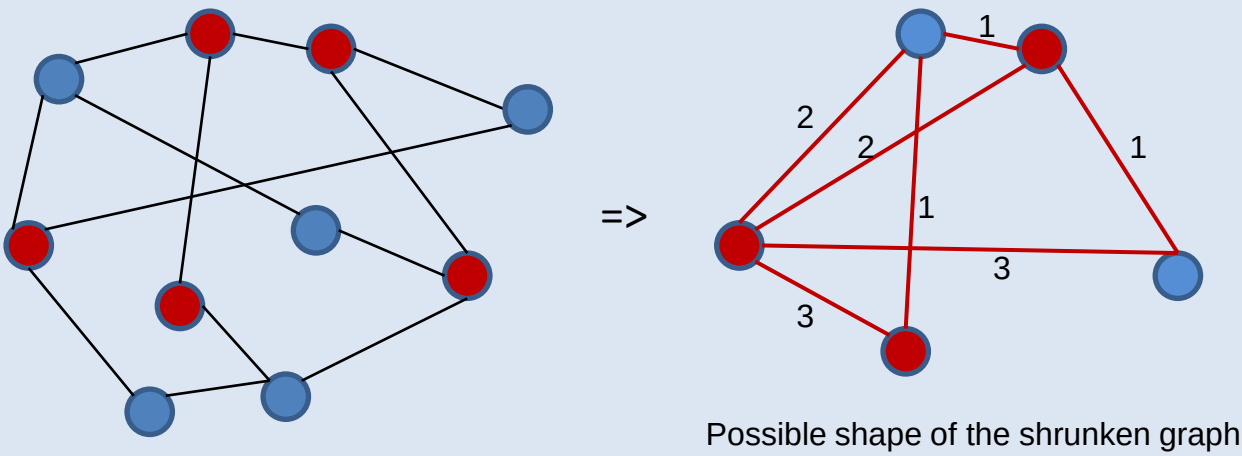
Recursive Cluster Growing

Parallel Cluster Growing has the huge disadvantage that the choice of a reasonable k depends on the diameter for a fast computation. Otherwise it can be as slow as EM-BFS or even slower.

We decided to develop a recursive cluster growing. A small k_1 is chosen for the first clustering. The shrunken graph is clustered again until it fits into main memory and an internal-memory SSSP can be computed. In the last clustering iteration i we have a simple adaptive rule that ensures the right choice of k_i.

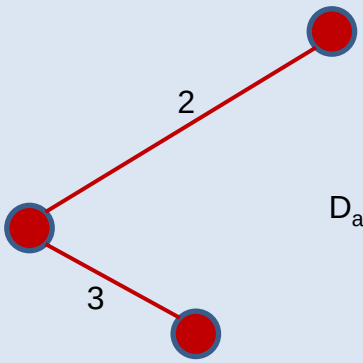
For our current input ($\leq$128 GB and M=4GB) two iterations with $k_1 = 16$ sufficed to execute internal-memory SSSP on the shrunken graph with a speedup of 70 to 85% compared to EM-BFS.

Approximation error: We found a graph class that has an expected approximation error of $\Omega(k^{4/3-\varepsilon})$ for two iterations. However, a tight bound for all graph classes has not been proved yet. Nevertheless, the experimental results are viable.



Possible shape of the shrunken graph

II
V



$D_{approxed} = 5$

Usually to the result an error correcting factor is added. In this example it would be 4.

M. Sc. David Veith Goethe-Universität Frankfurt am Main, dveith@cs.uni-frankfurt.de

Current and Future Work

Current work – Dynamic BFS

In 2008 Meyer has published an I/O-efficient algorithm for dynamic BFS. We will implement this algorithm now to proof that it is viable. An I/O-efficient implementation of Dynamic BFS is important to answer real time queries in large networks. Such queries have become an important application for social networks.

Current work – Static BFS

In 2011 one of our master students has developed a hierarchical clustering algorithm which is able to increase / decrease the number of clusters in scanning complexity. This is important for the dynamic BFS implementation, too.

Future work – External-memory SSSP implementation

Currently our SSSP implementation is limited in the input size. We need at least one bit for each vertex in main memory. Furthermore, new heuristics working with hyperbolic metrics can answer SSSP queries very fast. We plan to analyze if it could be used in implementation for some new heuristics.

M. Sc. David Veith Goethe-Universität Frankfurt am Main, dveith@cs.uni-frankfurt.de