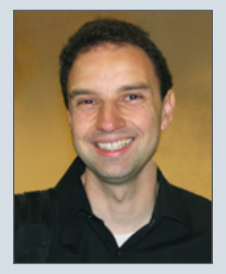
MADALGO CENTER FOR MASSIVE DATA ALGORITHMICS

Deepak Ajwani
Bell Labs, Ireland


Ulrich Meyer
Goethe University

David Veith
Goethe University

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

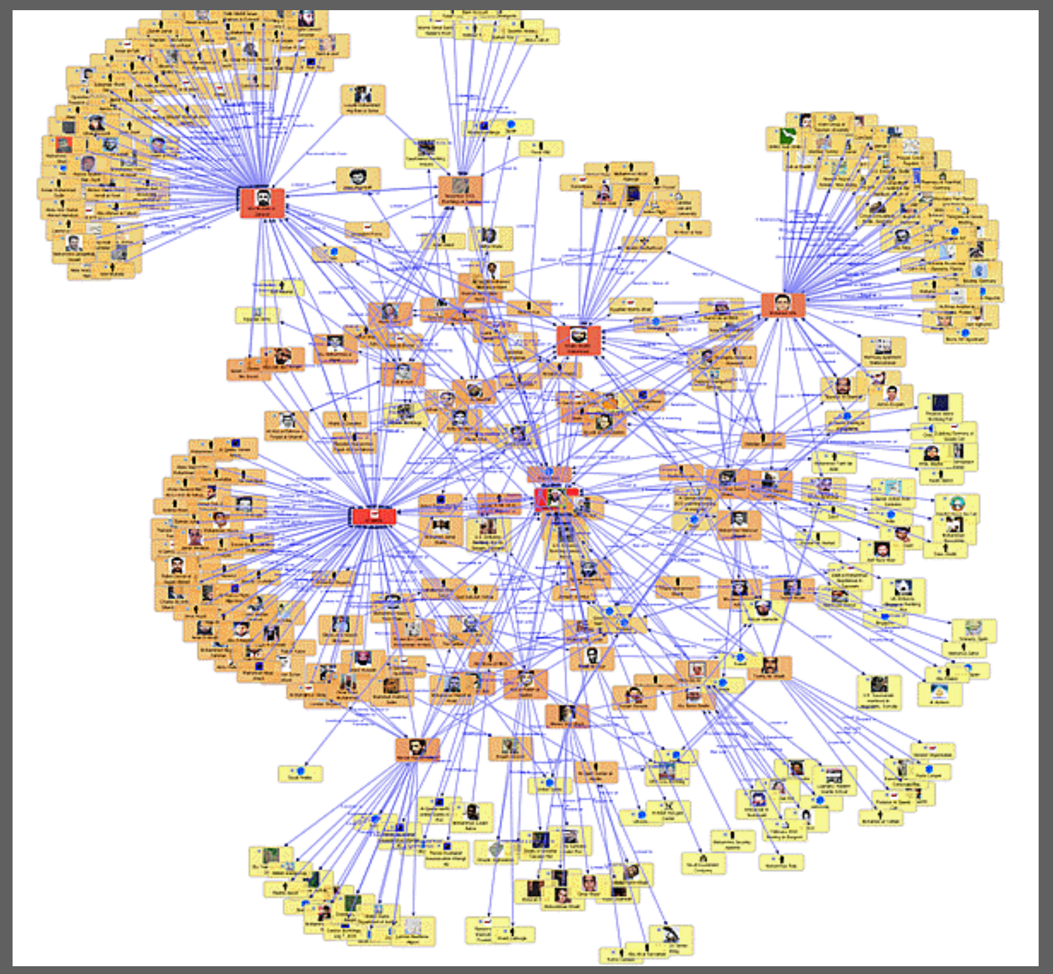
Danmarks Grundforskningsfond
Danish National Research Foundation

I/O-efficient Approximate Distance Oracle for Real-world Graphs

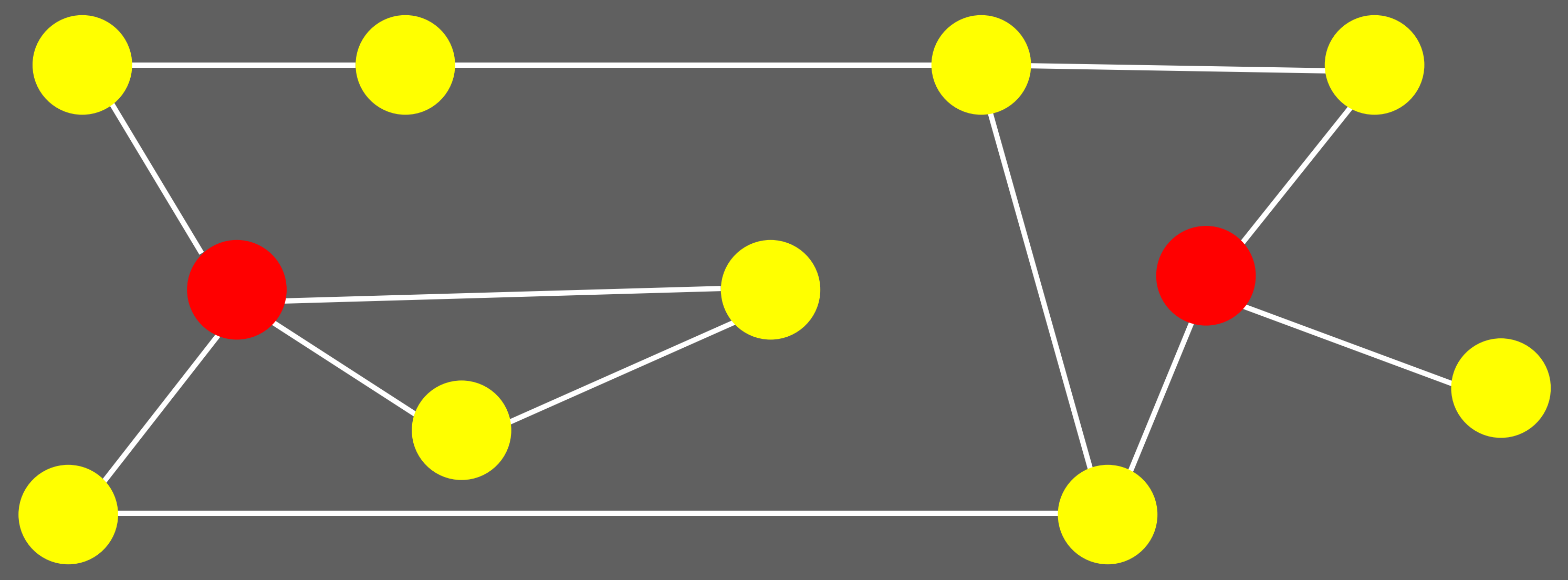
Path Computation in External Memory

In this work we focus on undirected, unweighted graphs with small diameter. This fits well for real world graph data as social network or web graphs. Determining the distance from one vertex to all other vertices in such a network is possible by a Breath-First Search (BFS).

The best known BFS implementation for general graphs has an I/O-complexity of $\Omega(N/\sqrt{B})$ I/Os. For graphs with small diameter exists an implementation with an I/O-complexity of $O(d \cdot \text{scan}(N)+\text{sort}(M))$ I/Os.


Social networks usually have a small diameter in $O(\log(N))$

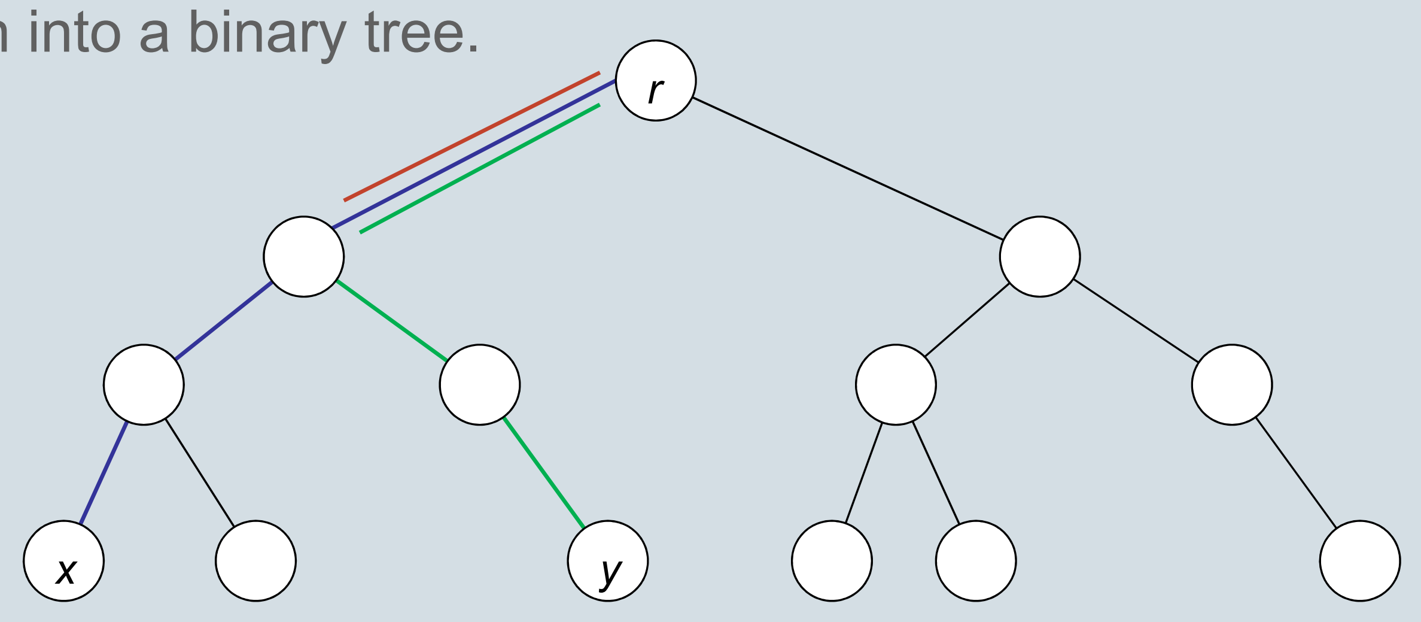
For our work we use the idea of a distance oracle published by Ajwani et al. in [1]. For the construction of their oracle the authors compute the BFS-tree of a small subset of the vertices and determine the distance of any pair of vertices u and v by dist(u)+dist(v)-2·dist(LCA) where LCA is the last common ancestor of u and v.



We compute a set of BFS-trees with high degree vertices as root vertex. In real world data such vertices are often a centre vertex – an important part of the graph structure [1].

The Distance Oracle

Our distance oracle shall answer a single query using $O(1)$ I/Os and $O(n)$ batched queries in a row using $O(1/B)$ I/Os. To achieve approximated distances close to the exact distance we determine for u and v their LCA in each tree. This is done by combining their inorder numbers in the BFS tree with the XOR operation [1]. For the construction of the inorder numbers we use a Huffman encoding for binary trees. Therefore some LCA vertices will be "dummy vertices" that have to be memorized due to the transformation into a binary tree.



$d(x,r) + d(y,r) - 2 \cdot d(\text{lca}(x,y),r)$

The result of the distance oracle is stored in two data structures: _IN and _DIST. In _IN for each vertex its distance to the root and its inorder number is stored for each tree. And in _DIST for each we store the inorder numbering and distance for all vertices including the dummy vertices. We use _DIST as a lookup table to determine the distance of the LCA.

To achieve $O(1/B)$ I/Os we scan _IN and _DIST (now as a vector in sorted order) in parallel for each tree separately and determine distance of u, v, the LCA and its distance with a few scanning and sorting steps. The best distance over all rounds is kept as the approximated distance.

Preprocessing & Dynamic Aspects

- For our distance oracle we have to compute a set of BFS trees once
- To gain a faster preprocessing we can build a hierarchy instead
  - Compute a single large BFS tree and compute $\log(n)$ multi-BFS[2] tree sets with decreasing size of the trees
- We restrict ourselves on graphs with small diameter. Therefore we can use MR-BFS instead of the more expensive MM-BFS.
- Social network graphs are frequently updated. Therefore we investigate how to deal with that aspect.
  - One possibility is to update each BFS tree independently and do not wait until all trees of the set are updated.
  - For our hierarchical preprocessing dynamic updates can be easily done for the smaller sub trees using a constant fraction of I/Os. Only the few larger trees might take longer.

Preliminary Results and Future Work

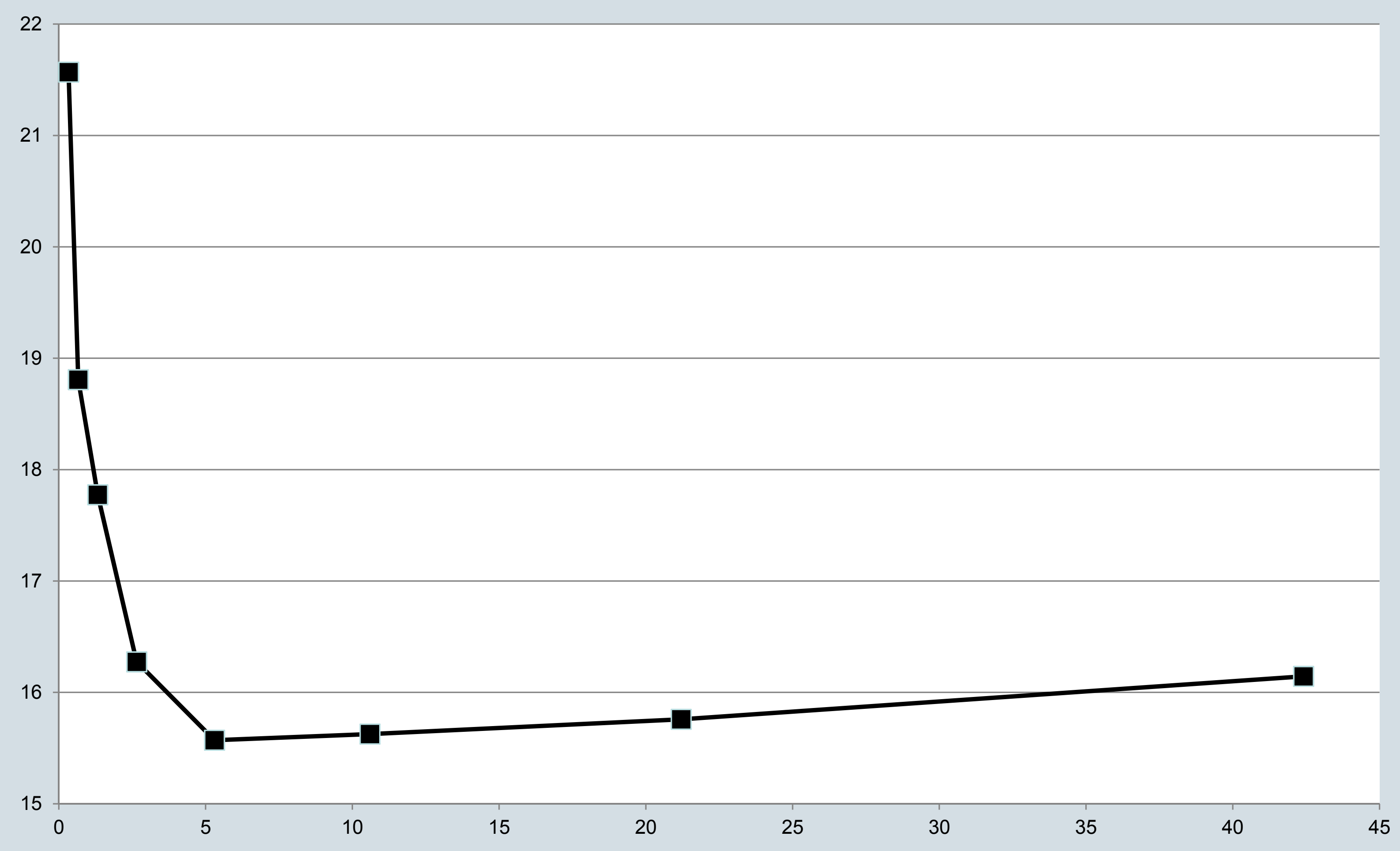
For our experiments we focused on SSDs. As in [1] we use 20 full BFS trees to build our oracle. The results on this poster are based on experiments with a web graph called sk-2005 with 50 million vertices and 1.8 billion edges (law.di.unimi.it/webdata/sk-2005/). The preprocessing result for 20 BFS trees has a size of 35 GB for sk-2005.

We tested on a machine with a 4.1GHz AMD quad core, 6 SSDs and 32 GB main memory (but we restricted ourselves to small main memory usage on this machine).

For a single query we were able to achieve a answer time of 6ms in average by using a small block size of a 4 KB. About 400 mostly random I/Os were obtained for 20 BFS trees per query.

For the batched queries we can accelerate the average time per query to 16µs for up to 42·|V| queries.

avg per query (µs) for different $c \cdot n$



Future work

This work is still in progress. We want to investigate the accuracy of our distance oracle for different data sets and plan to reduce the average time per query by some new tricks.

References

[1] Deepak Ajwani, W. Sean Kennedy, Alessandra Sala, Iraj Saniee. *A Distance Approximation Oracle for Large Real-World Graphs Based on Graph Hyperbolicity*. Unpublished manuscript.

[2] Deepak Ajwani, Ulrich Meyer, David Veith. *I/O-efficient Hierarchical Diameter Approximation*. ESA 2012.