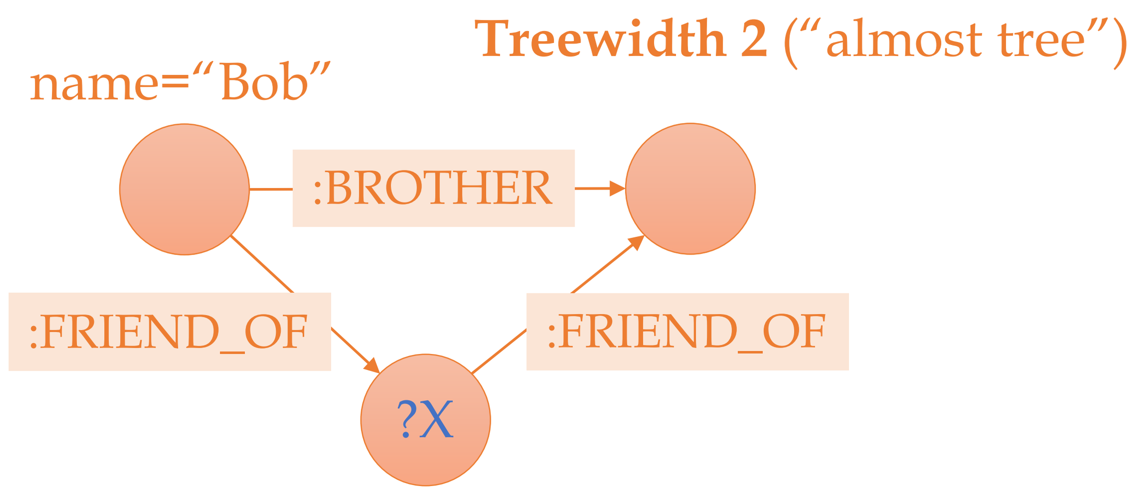
Graph Database Query Engine Based on Tree Decompositions

Karel Horák, supervised by Radomír Černoch, MSc.

Graph databases are a promising branch of storage systems. We have proposed a new algorithm for querying such databases with the aim to ensure the best possible worst-case time complexity. Our approach is based on the recent results from the field of graph homomorphism and uses them in the context of graph databases. By a series of experiments we have shown that our algorithm is faster than the leading graph database Neo4j in some of the scenarios.

Ask the graph

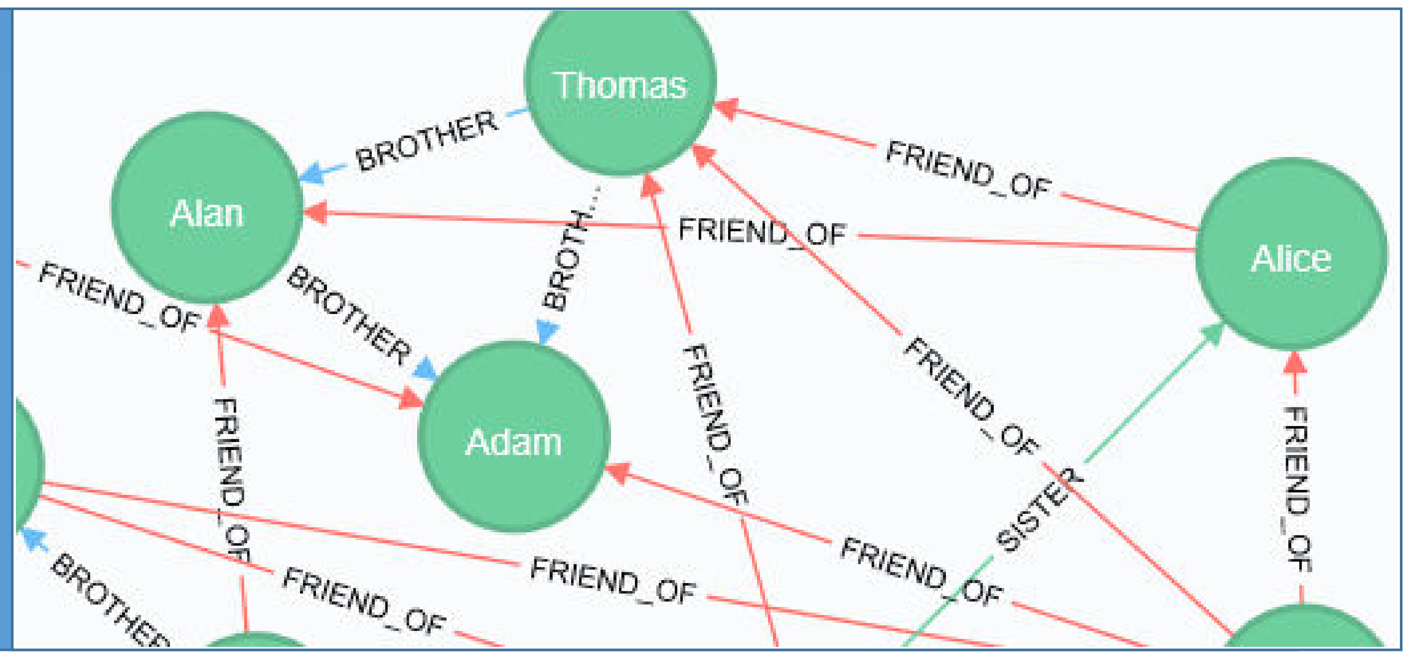
QUERY (Graph G)



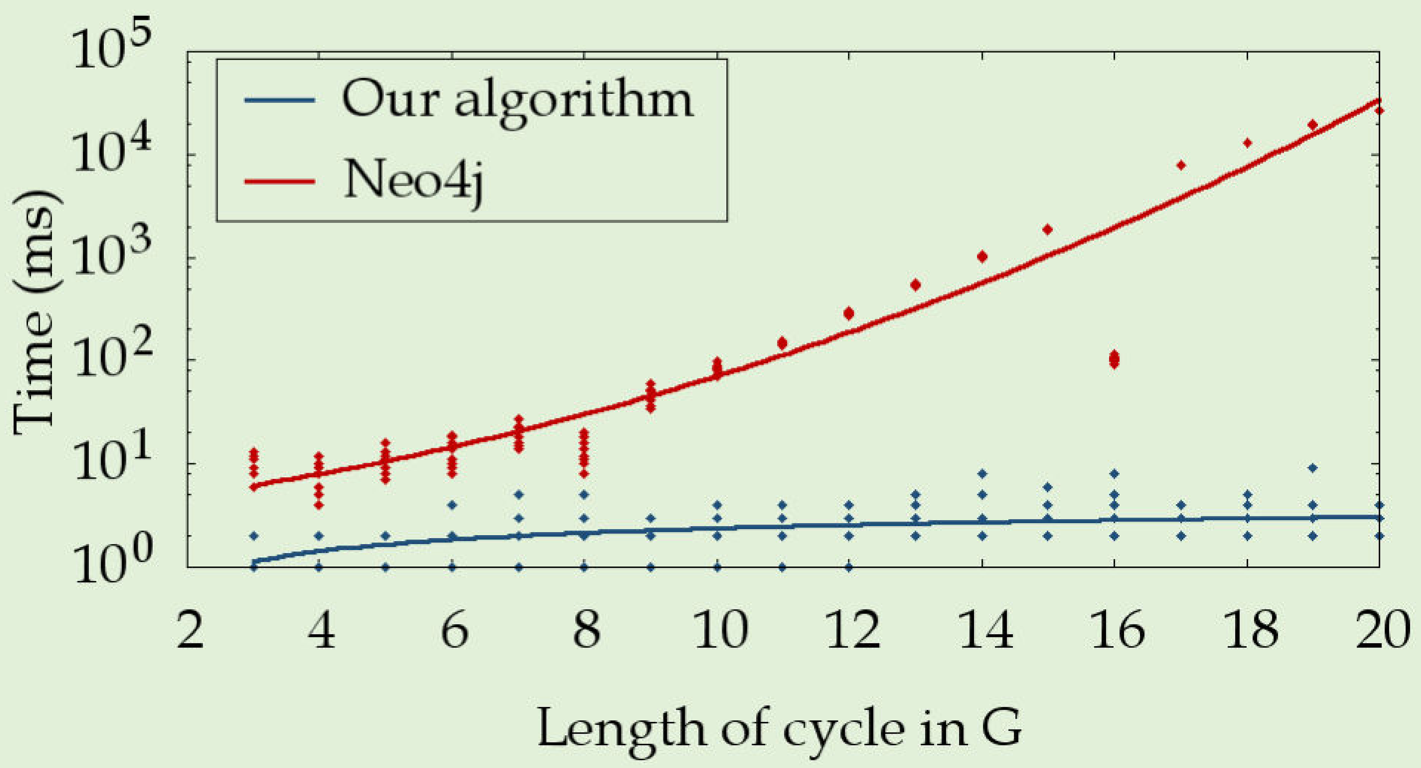
Give me mutual friends of Bob and his brother!

Homomorphism

DATABASE (Graph H)



Best result



Finding directed cycles of an increasing length in a DAG

Solution

1) Read the query graph

2) Compute a tree decomposition

"nodes" of the tree decomposition

3) Compute the solution for one of the nodes

→ Conventional methods may be used.

4) Use resolved nodes to solve neighbours.

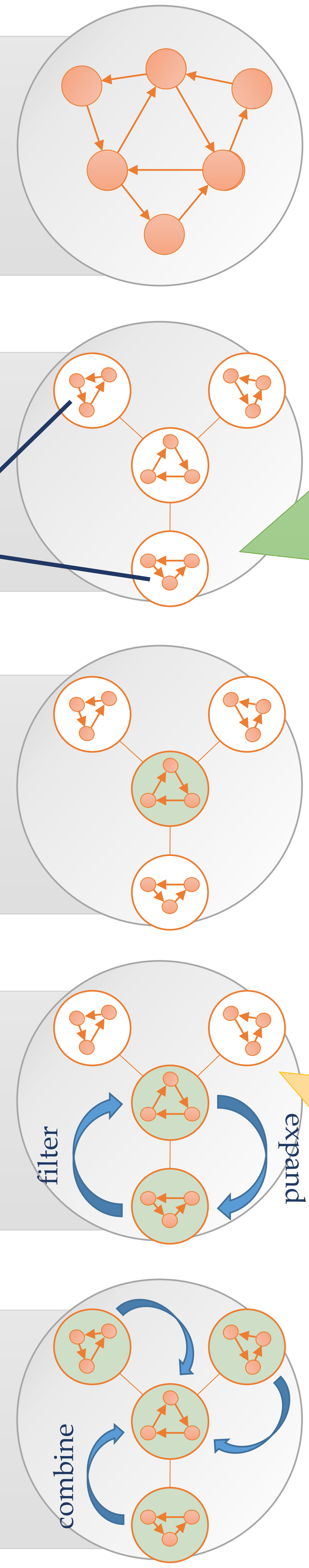
→ **Trick:** Dismiss solutions that are known to be irrelevant for the original problem.

→ This is a **local** property within the tree.

Repeat until an unresolved node exists.

5) Enumerate solutions

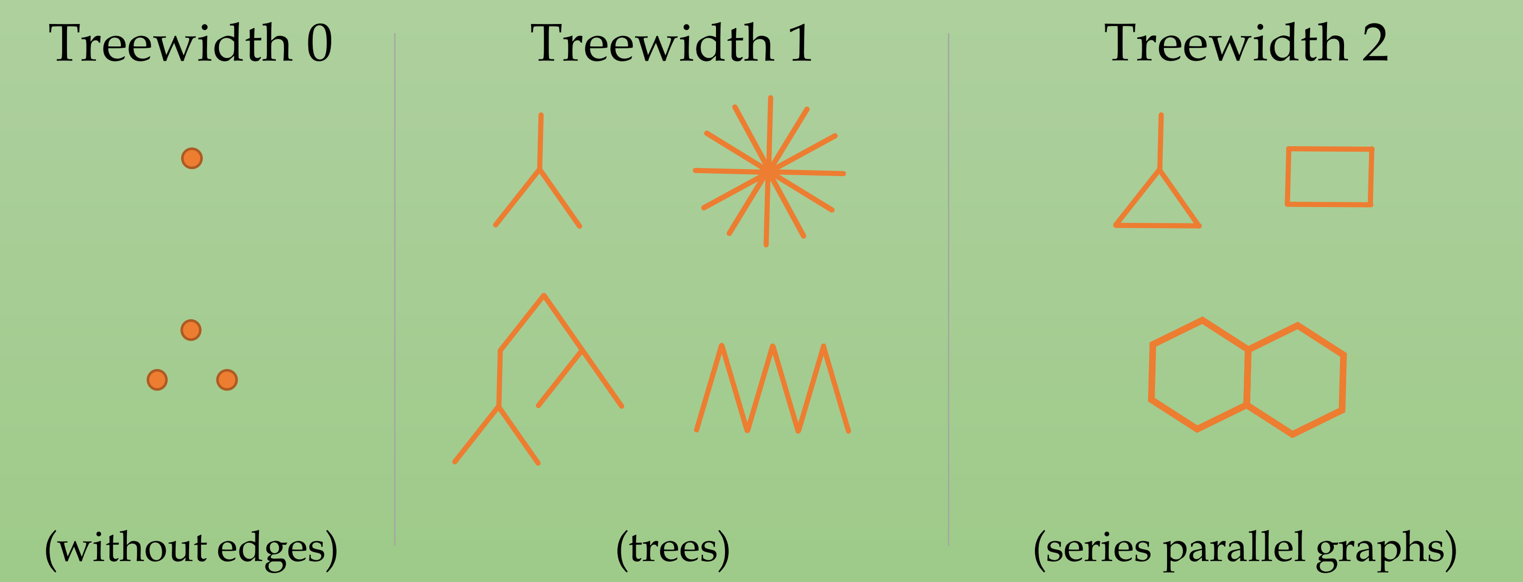
→ Combine matching solutions from individual nodes.



Treewidth

→ Measures similarity of a graph to a tree.

→ The size of the largest portion of a graph, that has to be treated together



Known results

For simple undirected graphs the decision problem (i.e. decide whether a homomorphism exists) is in:

- $\mathcal{NP}$-$\mathcal{C}$ for unrestricted queries (unless database is a bipartite graph) [1990, Hell and Nešetřil]

- PTIME for queries of bounded treewidth
- PTIME iff queries have bounded treewidth modulo homomorphic equivalence (decision variant only) [2002, Dalmau et al. ; 2007, Grohe]

$$O(V(T) \cdot \boldsymbol{tw}(G) \cdot V(H)^{\boldsymbol{tw}(G)+1} + OUT)$$

Exponential only in the treewidth

$V(T)$ - number of nodes, $\boldsymbol{tw}(G)$ – treewidth of the query,
$V(H)$ - number of vertices in the database, OUT - number of results