Finding optimal tree decompositions

Tuukka Korhonen

HIIT, Department of Computer Science, University of Helsinki

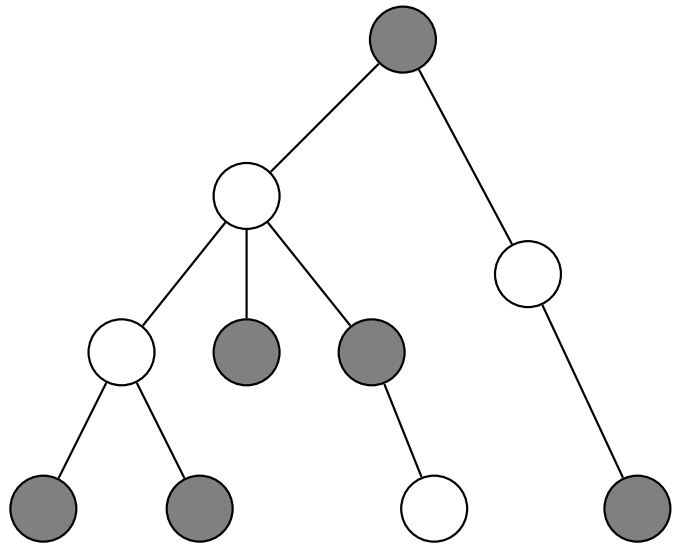
CS Colloquium
February 28, 2020

Outline

- Master's thesis on practical algorithms for finding optimal tree decompositions w.r.t. different notions of "optimal"
- Contributions
  - State-of-the-art implementations for exact computing of:
    - ★ Minimum fill-in
    - ★ Total table size
    - ★ Generalized and fractional hypertreewidth
    - ★ Maximum compatibility of multi-state phylogenetic characters
  - Papers in ACM Journal of Experimental Algorithms (2019), IJCAI 2019 and AAAI 2020
- Organization of the talk
  1. What are tree decompositions and why?
  2. Finding optimal tree decompositions

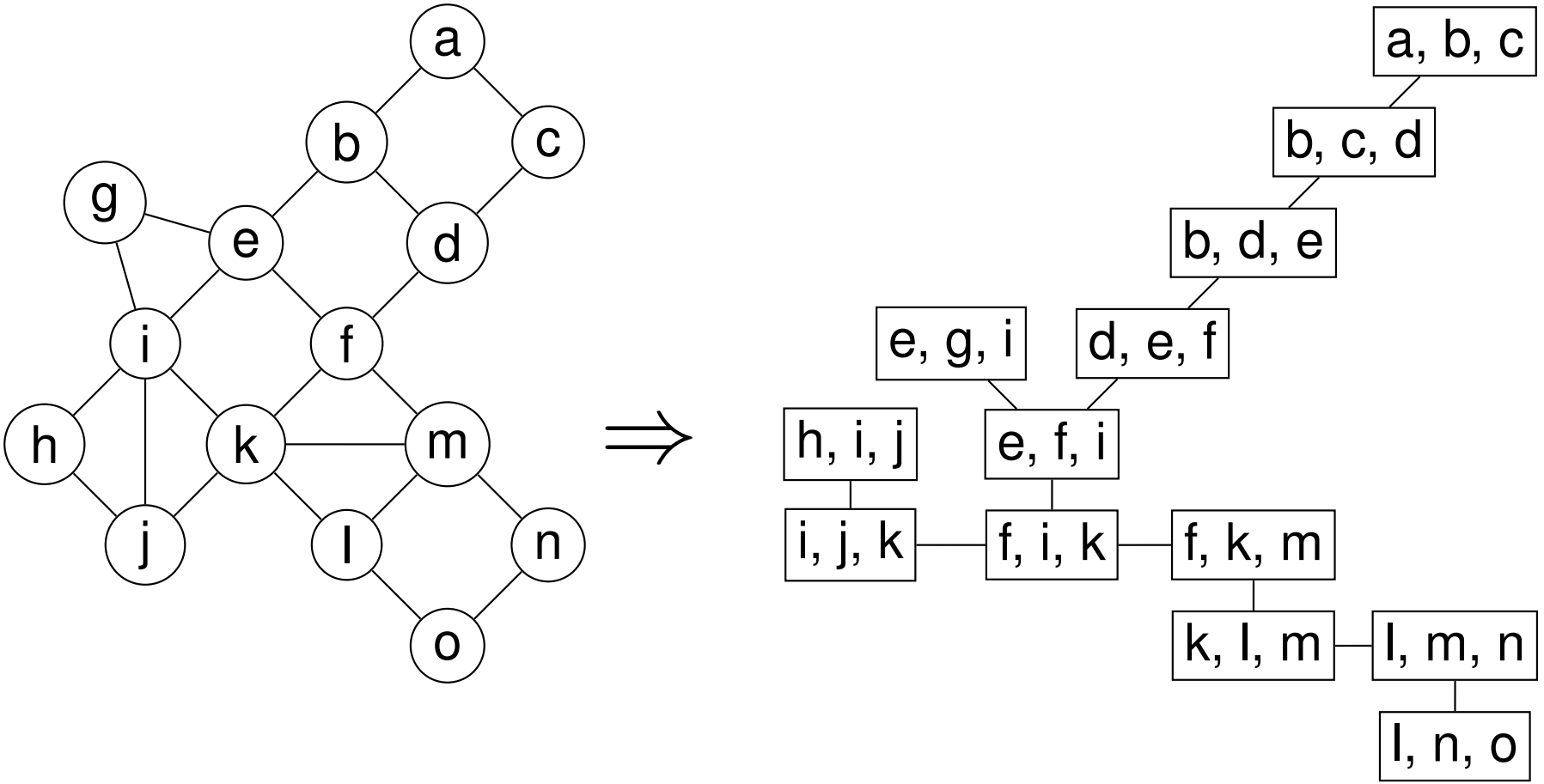
Motivation

- Hard problems are easy when data has special structure
- Example: Maximum independent set
  - Find the largest set of vertices with no edges in between
  - NP-hard in graphs in general
  - Linear-time in trees



Tree decompositions

- What if the graph is not a tree, but almost?
- Tree decomposition with low treewidth



Applications

- Theory: Huge number of problems become easy when underlying graph has low treewidth
  - $O(n)$ algorithms for constant treewidth
- Successful applications in practice:
  - Probabilistic inference
  - Solving quantified Boolean formulas
  - Compiler optimization

Beyond treewidth

- Different problems require different kinds of tree decompositions
- In constraint solving:
  - Generalized hypertreewidth
  - Fractional hypertreewidth
- Motivated by sparse matrix computations (linear programming):
  - Minimum fill-in
- Motivated by probabilistic inference (junction tree algorithm):
  - Total table size
- Problems in phylogenetics formulated with tree decompositions:
  - Perfect phylogeny
  - Maximum compatibility of phylogenetic characters

Part 2

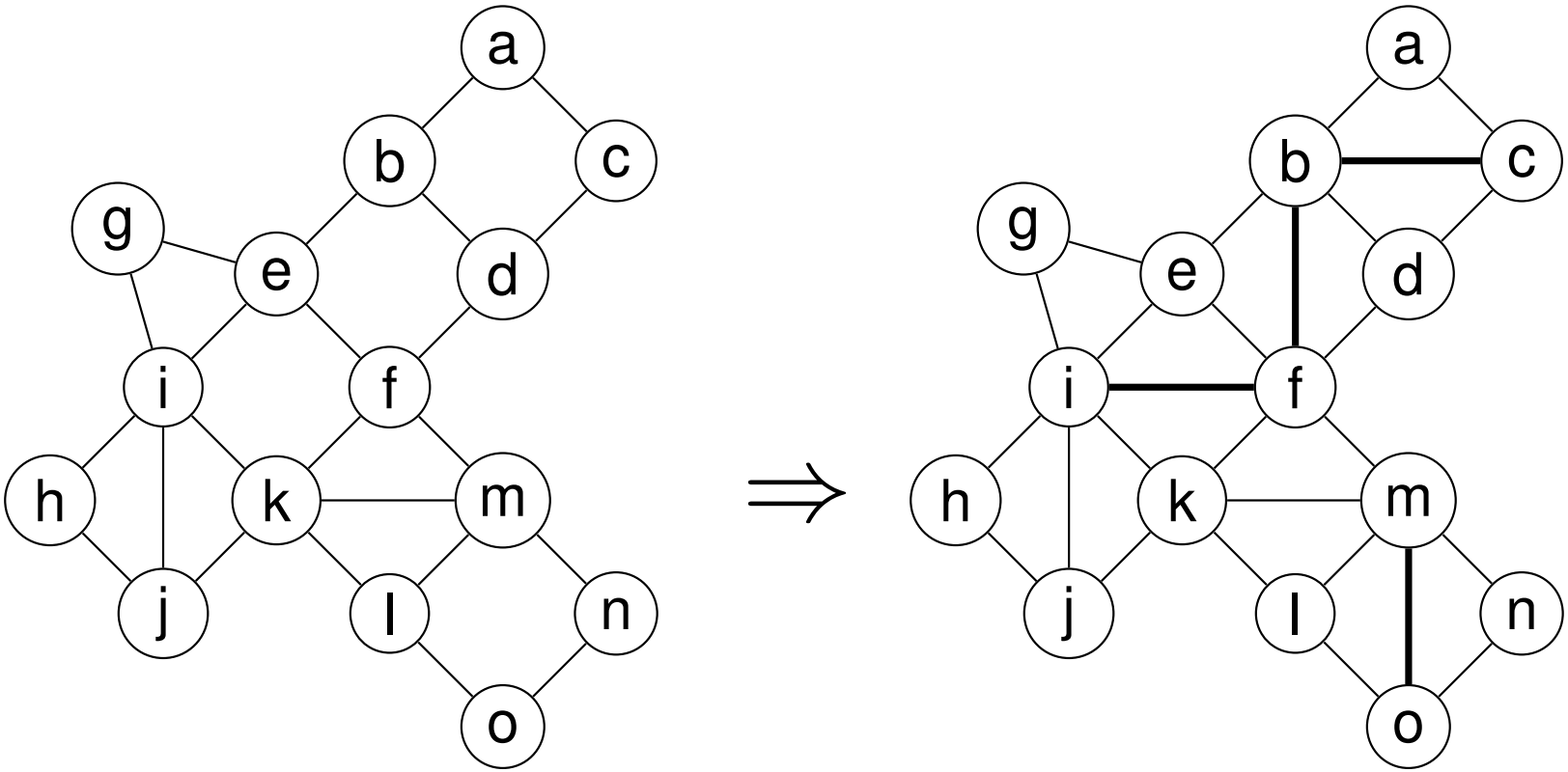
Optimal tree decompositions w.r.t. treewidth and all other parameters of the previous slide can be computed with **Bouchitté–Todinca algorithm**

BT algorithm – History

- New algorithm for treewidth and minimum fill-in using *potential maximal cliques* [Bouchitté and Todinca, 2001]
- Many theoretical results based on potential maximal cliques [Fomin et al., 2008; Fomin and Villanger, 2013]
- Best implementations in PACE 2017 challenge for treewidth and minimum fill-in use potential maximal cliques [Dell et al., 2017]
- Potential maximal cliques yield state-of-the-art practical algorithms in many more related problems [Korhonen et al., 2019b; Korhonen and Järvisalo, 2020]

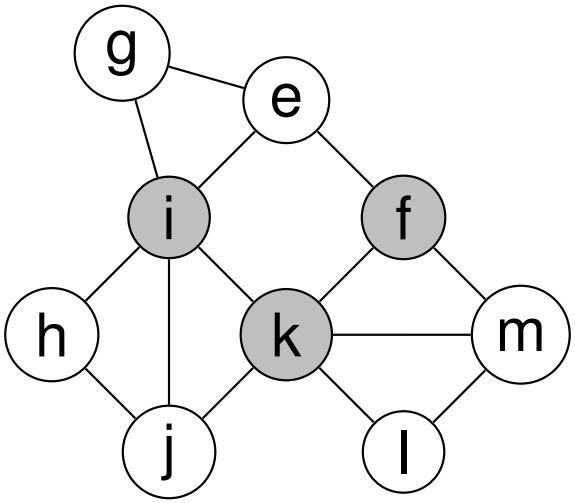
Triangulations of graphs

- Finding optimal tree decompositions is equivalent to finding optimal *triangulations* of graphs
- Triangulation: Add edges so that graph becomes *chordal*
- Chordal: All cycles of length ≥ 4 have a chord



Potential maximal cliques

- Potential maximal cliques are building blocks of triangulations
- A vertex set is a potential maximal clique if it is a maximal clique in some minimal triangulation
- Highly related to minimal separators
  - $O(|\Delta|^2 n^2 m)$ algorithm to list all potential maximal cliques, where Δ is the set of minimal separators



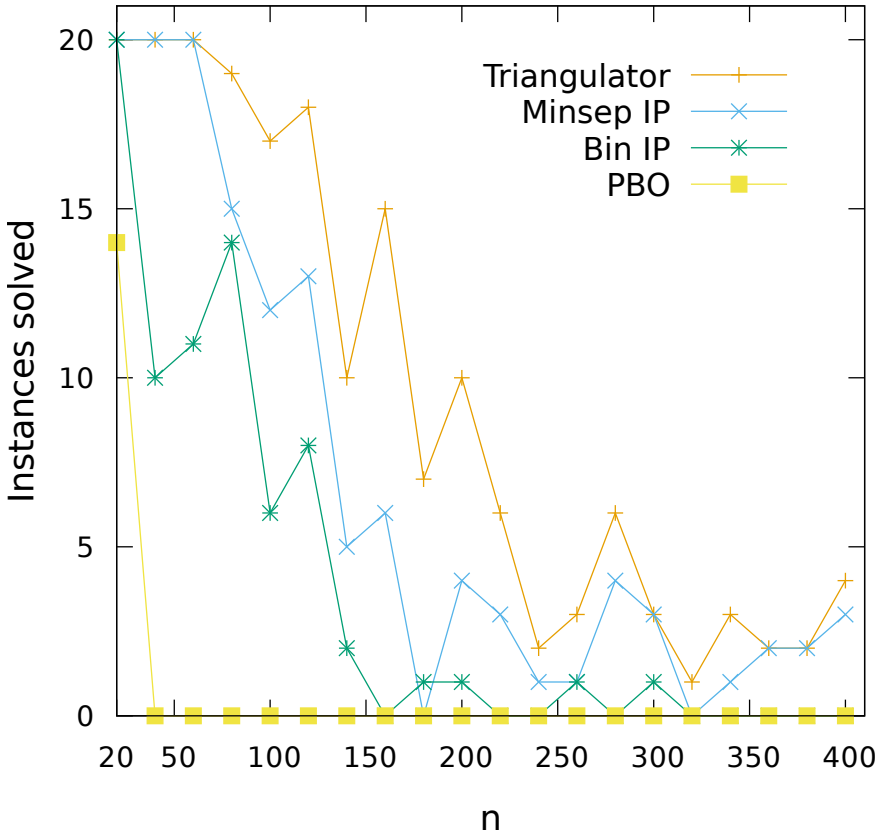
For example $\{i, k\}$, $\{k, f\}$ and $\{i, f\}$ are minimal separators and $\{i, f, k\}$ is a potential maximal clique.

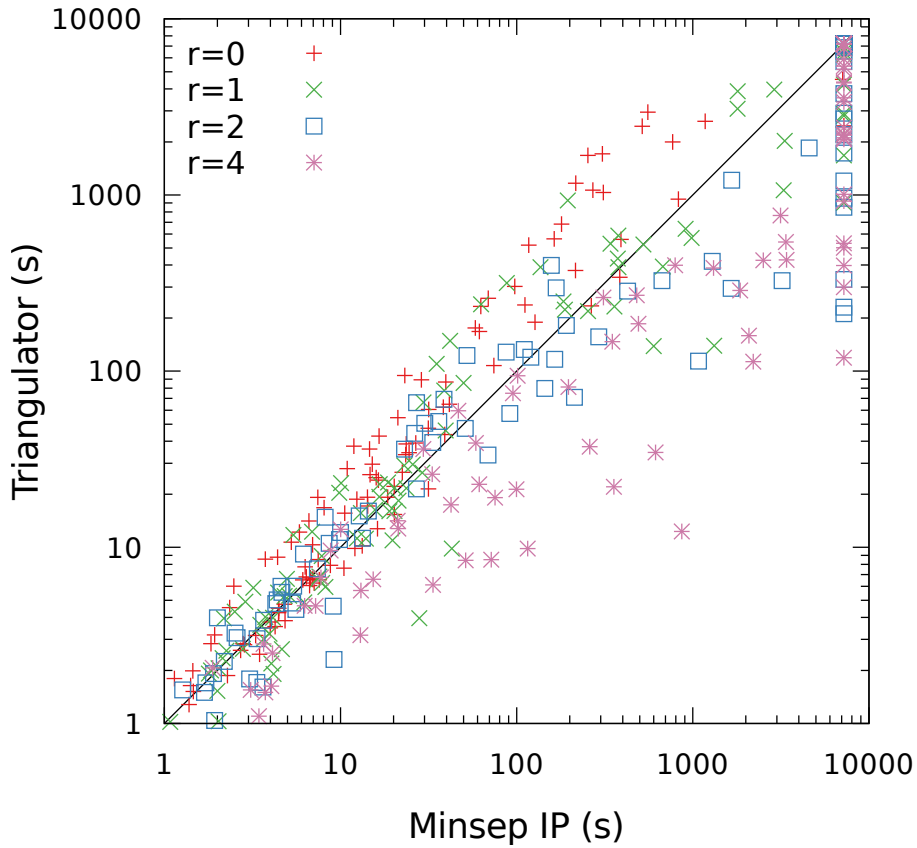
BT algorithm

- Given the potential maximal cliques Π, an optimal minimal triangulation can be found in $O(|\Pi|n^3)$ time
- BT algorithm:
  1. Enumerate potential maximal cliques
  2. Dynamic programming over potential maximal cliques to find an optimal triangulation
- In practice, phase 1 is the running time bottleneck

Experimental results

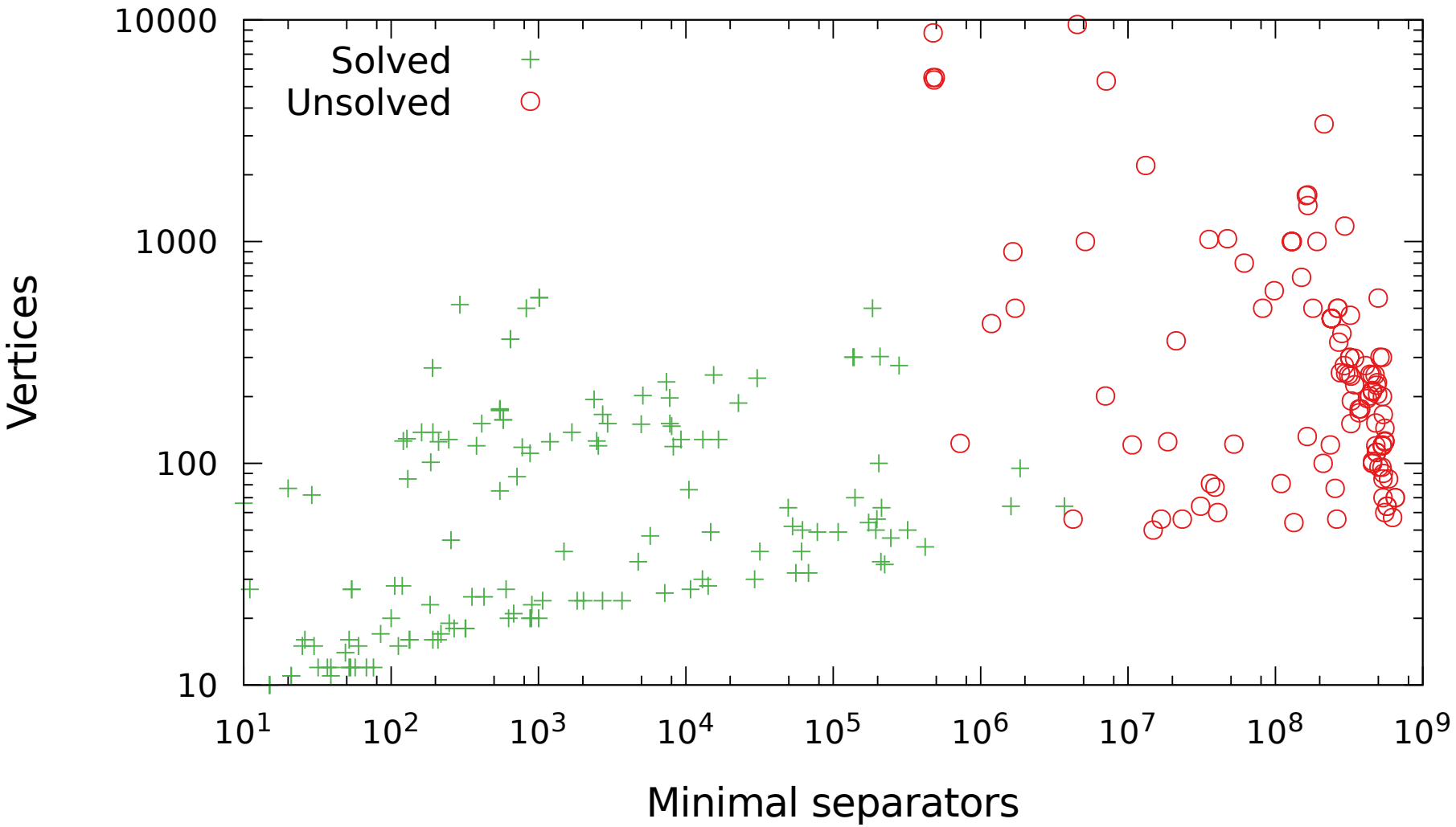
Comparison to other algorithms on Maximum compatibility problem of phylogenetic characters





Experimental results

How large instances can we solve? (Minimum fill-in)



The end

Thank you for your attention!

Bibliography

Vincent Bouchitté and Ioan Todinca. Treewidth and minimum fill-in: Grouping the minimal separators. *SIAM Journal on Computing*, 31(1):212–232, 2001.

Holger Dell, Christian Komusiewicz, Nimrod Talmon, and Mathias Weller. The PACE 2017 parameterized algorithms and computational experiments challenge: The second iteration. In Daniel Lokshtanov and Naomi Nishimura, editors, *12th International Symposium on Parameterized and Exact Computation, IPEC 2017, September 6-8, 2017, Vienna, Austria*, volume 89 of *LIPIcs*, pages 30:1–30:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi: 10.4230/LIPIcs.IPEC.2017.30. URL https://doi.org/10.4230/LIPIcs.IPEC.2017.30.

Fedor V. Fomin and Yngve Villanger. Subexponential parameterized algorithm for minimum fill-in. *SIAM Journal on Computing*, 42(6):2197–2216, 2013. doi: 10.1137/11085390X. URL https://doi.org/10.1137/11085390X.

Fedor V. Fomin, Dieter Kratsch, Ioan Todinca, and Yngve Villanger. Exact algorithms for treewidth and minimum fill-in. *SIAM Journal on Computing*, 38(3):1058–1079, 2008.

Tuukka Korhonen and Matti Järvisalo. Finding most compatible phylogenetic trees over multi-state characters. In *Proc. AAAI (to appear)*, 2020. URL https://www.cs.helsinki.fi/u/mjarvisa/papers/kj.aaai20.pdf.

Tuukka Korhonen, Jeremias Berg, and Matti Järvisalo. Enumerating potential maximal cliques via SAT and ASP. In Sarit Kraus, editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 1116–1122. ijcai.org, 2019a. doi: 10.24963/ijcai.2019/156. URL https://doi.org/10.24963/ijcai.2019/156.

Tuukka Korhonen, Jeremias Berg, and Matti Järvisalo. Solving graph problems via potential maximal cliques: An experimental evaluation of the bouchitté-todinca algorithm. *ACM Journal of Experimental Algorithmics*, 24(1):1.9:1–1.9:19, 2019b. doi: 10.1145/3301297. URL https://doi.org/10.1145/3301297.