


Tweed: A Heuristic Solver for Treedepth (Submitted to PACE 2020 under username *peaty*)

James Trimble 

School of Computing Science, University of Glasgow
Glasgow, Scotland, UK
j.trimble.1@research.gla.ac.uk

Abstract

We introduce Tweed, a heuristic solver for the treedepth problem. The solver uses two well-known approaches to create an initial elimination tree: nested dissection (making use of the Metis library) and the minimum-degree heuristic. After creating an elimination tree of the entire input graph, the solver attempts to improve it by applying nested dissection and the minimum-degree heuristic to subtrees of the elimination tree.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Theory of computation → Algorithm design techniques

Keywords and phrases Treedepth, Elimination Tree, Graph Algorithms

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

Supplementary Material DOI: <https://doi.org/10.5281/zenodo.3881441>; GitHub: <https://github.com/jamestrimble/pace2020-treedepth-solvers>

Funding *James Trimble*: This work was supported by the Engineering and Physical Sciences Research Council (grant number EP/R513222/1).

1 The Tweed and Tweed-Plus Solvers

This note introduces two heuristic solvers, Tweed and Tweed-Plus, for the treedepth problem. Both solvers try two well-known approaches—the minimum degree heuristic [1] and nested dissection—to find a good initial decomposition. They then repeatedly find a subtree containing a vertex of maximum depth and use one of the two well-known heuristics to try to replace the subtree with an improved decomposition.

The algorithm makes use of the Metis library [2] to find nested dissection orderings.

Tweed’s version of the minimum degree heuristic differs slightly from standard versions: when a vertex is deleted from the graph, the Tweed algorithm does not decrement the degrees of its neighbours.

The Tweed algorithm includes an additional improvement step that is not present in Tweed-Plus. This is a somewhat rough-and-ready function that finds a chain in the decomposition (a path of vertices of increasing depth such that each parent in the path has only one child vertex in the tree), and seeks to reorder it in a way that reduces the depth of the decomposition. This step aims to be in the spirit of the literature on rotations of elimination trees [3], but the simple approach implemented in Tweed does not appear to make a substantial improvement to the depth of the best tree found.

2 Timings

The algorithm is designed to be run for 30 minutes per instance. During the first 28 minutes, it spends approximately:



© James Trimble;
licensed under Creative Commons License CC-BY
42nd Conference on Very Important Topics (CVIT 2016).

Editors: John Q. Open and Joan R. Access; Article No. 23; pp. 23:1–23:2



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1. the first 3 minutes running the minimum degree heuristic without subtree improvements, then
2. 8 minutes running nested dissection without subtree improvements
3. 9 minutes running nested dissection with subtree improvements
4. 3 minutes running minimum degree without subtree improvements
5. 5 minutes running minimum degree with subtree improvements

The timings were tuned by hand based on the PACE Challenge public instances, and could be improved upon. In each phase, the algorithm runs many times (with different results due to randomisation), provided the graph is sufficiently small. The best solution so far is stored.

For the last two minutes, the algorithm attempts to improve upon the best solution found so far. This final improvement step appears to be useful in some cases; for example, it brings the best depth found down from 138 to 135 on PACE Challenge public instance 035. Although the idea of intensive improvement of the best instance found is a simple idea, I suspect that it may be the most useful idea that can be taken from Tweed and applied to other heuristic solvers.

3 Data structures

Small adjacency lists (up to 32 vertices) are stored as unsorted lists. Larger adjacency lists are stored as bitsets.

4 Dependencies

Tweed and Tweed-Plus use code from Nauty 2.6r12 [4] for the bitset data structure and random number generator (the latter of which is based in turn on code by Donald Knuth). The random number generator is only used in the heuristic presolve.

Both solvers use Metis 5.1.0 [2] to find a nested dissection ordering during the heuristic presolve.

References

- 1 Alan George and Joseph W. H. Liu. The evolution of the minimum degree ordering algorithm. *SIAM Review*, 31(1):1–19, 1989. URL: <https://doi.org/10.1137/1031001>, doi:10.1137/1031001.
- 2 George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Scientific Computing*, 20(1):359–392, 1998. URL: <https://doi.org/10.1137/S1064827595287997>, doi:10.1137/S1064827595287997.
- 3 Joseph W. H. Liu. Reordering sparse matrices for parallel elimination. *Parallel Computing*, 11(1):73–91, 1989. URL: [https://doi.org/10.1016/0167-8191\(89\)90064-1](https://doi.org/10.1016/0167-8191(89)90064-1), doi:10.1016/0167-8191(89)90064-1.
- 4 Brendan D. McKay and Adolfo Piperno. Practical graph isomorphism, II. *J. Symb. Comput.*, 60:94–112, 2014. URL: <https://doi.org/10.1016/j.jsc.2013.09.003>, doi:10.1016/j.jsc.2013.09.003.