Bute: An Bottom-Up Exact 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 ——

This note introduces the exact solver *Bute* for the exact treedepth problem, along with two variants of the solver. Each of these solvers computes an elimination tree in a bottom-up fashion by finding sets of vertices that induce subgraphs of small treedepth, then combining sets of vertices together with a root vertex to produce larger sets. The algorithms make use of a trie data structure to reduce the effort required to determine acceptable combinations of subtrees. Bute-Plus and Bute-Plus-Plus add a heuristic presolve step, which can quickly find a treedepth decomposition of optimal depth for many instances.

1 Brief Description of the Bute Algorithm

This paper introduces a family of three solvers—Bute, Bute-Plus, and Bute-Plus-Plus—that find an elimination tree of minimum height (which is also an optimal treedepth decomposition [6, 5]) for a given connected graph. The solvers take their name is from the Isle of Bute (an island not far from Glasgow). The name is also an almost-acronym for the solvers' bottom-up approach to constructing an elimination tree.

The optimisation problem is solved as a sequence of decision problems. The solver attempts to find an elimination tree of depth 1, then of depth 2, and so on until it is successful.

For the decision problem of whether an elimination tree of depth k exists, the algorithm works downwards for $i = k, \ldots, 1$, finding all sets S of vertices such that both (1) S induces a subgraph of treedepth no greater than $k - i + 1$ and (2) the neighbourhood of S has fewer than i vertices. These sets include the vertex set of every subtree rooted at depth i of an elimination tree of the input graph.

At level i, the algorithm constructs sets in two ways. The first of these is simply by choosing vertices with a sufficiently small neighbourhood. The second is by finding a subset of sets at level $i + 1$, and a root vertex v, which together induce a subgraph of small treedepth.

2 Trie data structure

We assume that the vertices are integers. Let $N(v)$ denote the neighbourhood of vertex v, and let $N(S)$ denote those vertices that are not in S but are adjacent to some member of S.

To find sets of sets of vertices from level $i+1$ that can be combined to form a set of vertices at level i, the algorithm uses a trie data structure to represent sets. This allows the algorithm to quickly determine, for a given set U, sets S at level $i+1$ such that $|N(S) \cup N(U)| < i$. Our data structure is an augmented Set-Trie [7]. The task it carries out is similar to that of the trie data structure used by Tamaki in a positive-instance driven treewidth solver [8], although the implementation details of the Bute solver's data structure are more similar to a data structure for subset queries posted on Stack Overflow by the user PengOne [2].

If the set to be searched is small, the algorithm reverts to a brute-force search to avoid the overhead of constructing and searching the trie.

3   Domination rules

The Bute algorithm uses a domination-breaking rule that extends a rule by Ganian et al. [1]. It is always possible to construct an optimal elimination tree such that for distinct vertices v, w with $N(v) \setminus \{w\} \supset N(w) \setminus \{v\}$ we have that w is not an ancestor of v. Furthermore, if $N(v) \setminus \{w\} = N(w) \setminus \{v\}$ we can break symmetries by only allowing w to be an ancestor of v if $w > v$.

4   Versions with Heuristic Presolve

Bute-Plus runs a variant of the Tweed-Plus heuristic (submitted to the heuristic track of PACE 2020) for one minute to try to find an elimination tree of the optimum depth.

Bute-Plus-Plus is a minor variant on Bute-Plus. It runs the heuristic for two minutes rather than one, makes slightly less use of the trie data structure, and has a slighty modified query function for the trie that is optimised for larger graphs.

5   Dependencies

Bute has no dependencies.

Bute-Plus and Bute-Plus-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.

Bute-Plus and Bute-Plus-Plus use Metis 5.1.0 [3] to find a nested dissection ordering during the heuristic presolve.

────── **References** ──────

1   Robert Ganian, Neha Lodha, Sebastian Ordyniak, and Stefan Szeider. SAT-encodings for treecut width and treedepth. In Stephen G. Kobourov and Henning Meyerhenke, editors, *Proceedings of the Twenty-First Workshop on Algorithm Engineering and Experiments, ALENEX 2019, San Diego, CA, USA, January 7-8, 2019.*, pages 117–129. SIAM, 2019. URL: `https://doi.org/10.1137/1.9781611975499.10`, `doi:10.1137/1.9781611975499.10`.

2   PengOne (https://stackoverflow.com/users/544050/pengone).   Fast   data structure for finding strict subsets (from a given list).   Stack Overflow. URL:https://stackoverflow.com/a/6514445/3347737   (version: 2011-06-29).   URL: `https://stackoverflow.com/a/6514445/3347737`,   `arXiv:https://stackoverflow.com/a/6514445/3347737`.

3   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`.

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`.

5   Asier Mujika. About tree-depth. Bachelor's thesis, Universidad del País Vasco, 2015.

6   Jaroslav Nesetril and Patrice Ossona de Mendez. *Sparsity - Graphs, Structures, and Algorithms*, volume 28 of *Algorithms and combinatorics*. Springer, 2012. URL: `https://doi.org/10.1007/978-3-642-27875-4`, `doi:10.1007/978-3-642-27875-4`.

7   Iztok Savnik. Index data structure for fast subset and superset queries. In Alfredo Cuzzocrea, Christian Kittl, Dimitris E. Simos, Edgar R. Weippl, and Lida Xu, editors, *Availability, Reliability, and Security in Information Systems and HCI - IFIP WG 8.4, 8.9, TC 5 International Cross-Domain Conference, CD-ARES 2013, Regensburg, Germany, September 2-6, 2013. Proceedings*, volume 8127 of *Lecture Notes in Computer Science*, pages 134–148. Springer, 2013. URL: `https://doi.org/10.1007/978-3-642-40511-2_10`, `doi:10.1007/978-3-642-40511-2\_10`.

8   Hisao Tamaki. Positive-instance driven dynamic programming for treewidth. *J. Comb. Optim.*, 37(4):1283–1311, 2019. URL: `https://doi.org/10.1007/s10878-018-0353-z`, `doi:10.1007/s10878-018-0353-z`.