PACE Solver Description: SMS

Tuukka Korhonen
Department of Computer Science, University of Helsinki, Finland
`https://tuukkakorhonen.com`
tuukka.m.korhonen@helsinki.fi

Abstract

We describe SMS, our submission to the exact treedepth track of PACE 2020. SMS computes the treedepth of a graph by branching on the **S**mall **M**inimal **S**eparators of the graph.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis

Keywords and phrases Treedepth, PACE 2020, SMS, Minimal separators

Digital Object Identifier 10.4230/LIPIcs.IPEC.2020.30

Related Version A version of this solver description with an appendix containing additional details is available in `https://arxiv.org/abs/2006.07302`.

Supplementary Material The source code of SMS is available in [7] and in `https://github.com/Laakeri/pace2020-treedepth-exact`.

Funding This work has been financially supported by Academy of Finland (grant 322869).

1 Overview

SMS is an exact algorithm implementation for computing treedepth. SMS was developed for the 5th Parameterized Algorithms and Computational Experiments challenge (PACE 2020). The main algorithm implemented in SMS is a recursive procedure that branches on minimal separators [4]. Two variants of the branching algorithm are implemented, one with a heuristic algorithm for enumerating minimal separators and one with an exact algorithm [9]. Several lower bound techniques are implemented within the branching algorithm. Before applying the branching algorithm, preprocessing techniques are applied and a heuristic upper bound for treedepth is computed.

2 Notation

Let G be a graph with vertex set $V(G)$ and edge set $E(G)$. The graph $G[X]$ is the induced subgraph of G with vertex set X. The set $N(v)$ is the neighborhood of a vertex v and $N(X)$ is the neighborhood of a vertex set X. The treedepth of G is denoted by $\mathtt{td}(G)$. A minimal a, b-separator of G is a subset-minimal vertex set S such that the vertices a and b are in different connected components of $G[V(G) \setminus S]$. The set of minimal separators of G for all pairs $a, b \in V(G)$ is denoted by $\Delta(G)$ and the set of minimal separators with size at most k by $\Delta_k(G)$. The set of vertex sets of connected components of G is denoted by $\mathcal{C}(G)$.

3 The Algorithm

3.1 Branching

SMS is based on the following characterization of treedepth.

15th International Symposium on Parameterized and Exact Computation (IPEC 2020).
Editors: Yixin Cao and Marcin Pilipczuk; Article No. 30; pp. 30:1–30:4
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

▶ **Proposition 1** ([4]). *Let G be a graph. If G is complete then $\mathtt{td}(G) = |V(G)|$. Otherwise*

$$\mathtt{td}(G) = \min_{S \in \Delta(G)} \left(|S| + \max_{C \in \mathcal{C}(G[V(G) \setminus S])} \mathtt{td}(G[C]) \right).$$

Proposition 1 is implemented as a recursive algorithm that takes a vertex set X as input and computes $\mathtt{td}(G[X])$ by first enumerating the minimal separators of $G[X]$ and then branching from each minimal separator S to smaller induced subgraphs $G[C]$ for each component $C \in \mathcal{C}(G[X \setminus S])$. We make use of upper bounds by implementing Proposition 1 as a decision procedure which, given a vertex set X and a number k, decides if $\mathtt{td}(G[X]) \leq k$. Clearly, in this case we may consider only the minimal separators in $\Delta_{k-1}(G[X])$. Moreover, we handle the minimal separators with sizes $k-1$ and $k-2$ as special cases and thus consider only the minimal separators in $\Delta_{k-3}(G[X])$ in the main recursion. A minimal separator S with $|S| = k-1$ such that $\mathtt{td}(G[X \setminus S]) = 1$ must be a vertex cover of $G[X]$ and therefore is a neighborhood of a vertex. A minimal separator S with $|S| = k-2$ such that $\mathtt{td}(G[X \setminus S]) \leq 2$ has also a somewhat special structure, and we handle them with a modification of Berry's algorithm [1] for enumerating minimal separators.

3.2 Enumerating Small Minimal Separators

SMS spends most of its runtime in a subroutine which given a number k and a graph G enumerates $\Delta_k(G)$. To make use of the fact that heuristic enumeration of small minimal separators is more efficient than exact enumeration, two variants of the main branching algorithm are ran: first a variant using a heuristic minimal separator enumeration algorithm and then a variant using an exact minimal separator enumeration algorithm.

The heuristic enumeration algorithm is a simple modification of Berry's algorithm [1]. The modification prunes all minimal separators with more than k vertices immediately during the execution, outputting a set $\Delta'_k \subseteq \Delta_k(G)$ in $O(|\Delta'_k|n^3)$ time. As observed in [9], there are cases in which $\Delta'_k \neq \Delta_k(G)$. However, in practice the algorithm seems to often find all small minimal separators on the values of k that are relevant.

As an exact small minimal separator enumeration algorithm we implement the algorithm of Tamaki [9], including also the optimizations discussed in the paper. To the best of our knowledge there are no better bounds than $n^{k+O(1)}$ for the runtime of this algorithm. In practice it appears to usually have only a factor of 2-10 runtime overhead compared to the heuristic algorithm.

In cases when $G[C]$ is a child of G in the recursion, obtained by branching on a minimal separator $N(C) \in \Delta(G)$, and $|C| > |V(G)|/2$ we make use of the small minimal separators of G to enumerate the small minimal separators of $G[C]$. In particular, for all minimal separators $S \in \Delta_k(G[C])$, there exists a minimal separator $S' \in \Delta_{k+|N(C)|}(G)$ such that $S = C \cap S'$. Note that in this case $|N(C)|$ is exactly the difference in the values of k in recursive calls on $G[C]$ and G, and therefore $\Delta_{k+|N(C)|}(G)$ is already enumerated.

3.3 Lower Bounds

To avoid unnecessary re-computation, the known upper and lower bounds for $\mathtt{td}(G[X])$ are stored for each handled induced subgraph $G[X]$. To this end, an open addressing hashtable with linear probing is implemented. Also, we implement an ad-hoc data structure so that given a vertex set X, a vertex set $X' \subset X$ with the highest known lower bound for $\mathtt{td}(G[X'])$ can be found. This data structure uses the idea of computing subset-preserving hashes by using the intersection $X \cap V'$, where V' is a subset of vertices with size $O(\log n)$, where n is the number of elements in the data structure. Other implemented algorithms for computing lower bounds on $\mathtt{td}(G[X])$ are the MMD+ algorithm [3] which finds large clique minors, a depth-first search algorithm which finds long paths and cycles, and a graph isomorphism hashtable which finds already processed induced subgraphs $G[X']$ that are isomorphic to $G[X]$ and applies the lower bounds of $G[X']$ to $G[X]$.

3.4 Preprocessing Techniques

The preprocessing techniques implemented in SMS are *tree elimination* and the kernelization procedures described in [6]. Tree elimination finds a subgraph $G[T]$ such that $G[T]$ is a tree and $|N(V(G) \setminus T)| = 1$, i.e., the subgraph is attached to the rest of the graph only on a single vertex. Then it uses an exact algorithm to compute a list of length $\mathtt{td}(G[T])$ that characterizes the behavior of $G[T]$ with respect to treedepth of G [8], and replaces $G[T]$ with a construction of $O(\mathtt{td}(G[T])^2)$ vertices whose behavior is the same. The simplicial vertex kernelization rule from [6] is implemented as it is described there, but the shared neighborhood rule is generalized. In particular, if there are two non-adjacent vertices $u, v \in V(G)$, and the minimum u,v-vertex cut is at least k, where k is an upper bound for treedepth, then an edge can be added between u and v.

3.5 Upper Bounds

To compute upper bounds on treedepth we implement a novel heuristic algorithm. The algorithm first finds a triangulation (chordal completion) H of G using the LB-Triang algorithm [2] with a heuristic aiming to minimize the number of fill-edges in each step. Then it uses the branching algorithm, with some additional heuristics making it non-exact, to compute a treedepth decomposition of H. Any treedepth decomposition of H is also a treedepth decomposition of G. The properties of chordal graphs interplay nicely with the branching algorithm: chordal graphs have a linear number of minimal separators and the treewidth of a chordal graph can be computed in linear time [5]. Moreover, there exists a triangulation H of G with $\mathtt{td}(H) = \mathtt{td}(G)$, because treedepth can be formulated as a completion problem to a graph class that is a subset of chordal graphs [4].

References

1 Anne Berry, Jean-Paul Bordat, and Olivier Cogis. Generating all the minimal separators of a graph. *International Journal of Foundations of Computer Science*, 11(3):397–403, 2000. doi:10.1142/S0129054100000211.

2 Anne Berry, Jean-Paul Bordat, Pinar Heggernes, Geneviève Simonet, and Yngve Villanger. A wide-range algorithm for minimal triangulation from an arbitrary ordering. *Journal of Algorithms*, 58(1):33–66, 2006. doi:10.1016/j.jalgor.2004.07.001.

3 Hans L. Bodlaender and Arie M. C. A. Koster. Treewidth computations II. Lower bounds. *Information and Computation*, 209(7):1103–1119, 2011. doi:10.1016/j.ic.2011.04.003.

4 Jitender S. Deogun, Ton Kloks, Dieter Kratsch, and Haiko Müller. On the vertex ranking problem for trapezoid, circular-arc and other graphs. *Discrete Applied Mathematics*, 98(1-2):39–63, 1999. doi:10.1016/S0166-218X(99)00179-1.

5 Philippe Galinier, Michel Habib, and Christophe Paul. Chordal graphs and their clique graphs. In Manfred Nagl, editor, *Proceedings of the 21st International Workshop on Graph-Theoretic Concepts in Computer Science*, volume 1017 of *Lecture Notes in Computer Science*, pages 358–371. Springer, 1995. doi:10.1007/3-540-60618-1_88.

6 Yasuaki Kobayashi and Hisao Tamaki. Treedepth parameterized by vertex cover number. In Jiong Guo and Danny Hermelin, editors, *Proceedings of the 11th International Symposium on Parameterized and Exact Computation*, volume 63 of *LIPIcs*, pages 18:1–18:11. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.IPEC.2016.18.

7 Tuukka Korhonen. PACE 2020 exact treedepth submission: SMS, 2020. doi:10.5281/zenodo.3872898.

8 Alejandro A. Schäffer. Optimal node ranking of trees in linear time. *Information Processing Letters*, 33(2):91–96, 1989. doi:10.1016/0020-0190(89)90161-0.

9 Hisao Tamaki. Computing treewidth via exact and heuristic lists of minimal separators. In Ilias Kotsireas, Panos M. Pardalos, Konstantinos E. Parsopoulos, Dimitris Souravlias, and Arsenis Tsokas, editors, *Proceedings of the Special Event on Analysis of Experimental Algorithms*, volume 11544 of *Lecture Notes in Computer Science*, pages 219–236. Springer, 2019. doi:10.1007/978-3-030-34029-2_15.