BreakingTheCycle: An Exact Solver for Calculating the Minimum Directed Feedback Vertex Set in the 2022 PACE Challenge

Jonathan Gutermuth
Goethe University Frankfurt, Germany

Lars Huth
Goethe University Frankfurt, Germany

Marius Lotz
Goethe University Frankfurt, Germany

Johannes Meintrup
THM, University of Applied Sciences Mittelhessen, Giessen, Germany

Timo Mertin
Goethe University Frankfurt, Germany

Manuel Penschuck
Goethe University Frankfurt, Germany

Hung Tran
Goethe University Frankfurt, Germany

Abstract

The goal of the directed feedback vertex set problem is to, given a directed graph, find the smallest subset of nodes to be removed to render the graph acyclic. In this report, we give a short description of our exact solver found in *BreakingTheCycle*.

2012 ACM Subject Classification Mathematics of Computation → Graph algorithms

Keywords and phrases PACE Challenge 2022, Directed Feedback Vertex Set, Exact

Supplementary Material The project's source code can be found on GitHub `https://github.com/goethe-tcs/breaking-the-cycle` and Zenodo `https://doi.org/10.5281/zenodo.6602946`.

Funding This work was partially supported by the Deutsche Forschungsgemeinschaft (DFG) under grants ME 2088/4-2 (SPP 1736 — Algorithms for BIG DATA) and ME 2088/5-1(FOR 2975 — Algorithms, Dynamics, and Information Flow in Networks).
Johannes Meintrup: Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 379157101.

Acknowledgements We would like to thank Holger Dell and Anselm Haak for valuable discussions, the PACE committees for organizing the challenge and curating the benchmark data, as well as the Center for Scientific Computing at Goethe University Frankfurt for making their high-performance computing facilities available.

1 Introduction

Our *BreakingTheCycle* DFVS solver, follows a traditional algorithm design. We start by exhaustively applying a set of data reduction rules (see Section 2). Since several of these kernelization rules have a super-linear time complexity, we regularly attempt to decompose the input instance into strongly connected components (SCCs) which we then process independently. The remaining kernels are solved using a branch and bound algorithm incorporating several branching strategies (see Section 3).

1.1 Preliminaries

Let $G = (V, E)$ be a directed graph. Denote the open in- and out-neighborhoods of node $u \in V$ as $N_{\text{in}}(u) = \{v \,|\, (v, u) \in E\}$ and $N_{\text{out}}(u) = \{v \,|\, (u, v) \in E\}$, respectively, and let $N_{\text{und}}(u) = N_{\text{in}}(u) \cap N_{\text{out}}(u)$. Denote the closed neighborhoods as $N_{\text{in}}[u] = N_{\text{in}}(u) \cup \{u\}$,

$N_{\text{out}}[u] = N_{\text{out}}(u) \cup \{u\}$, and $N_{\text{und}}[u] = N_{\text{und}}(u) \cup \{u\}$. Let $G \setminus u = G(V \setminus \{u\}, \{(x,y) | (x,y) \in E \colon x \neq u \wedge y \neq u\})$ be the graph after *deleting* node u and $G \times u = G(V \setminus \{u\}, \{(x,y) | (x,y) \in E \colon x \neq u \wedge y \neq u\} \cup N_{\text{in}}(u) \times N_{\text{out}}(u))$ be the graph after *contracting* node u (valid only if $(u,u) \notin E$). The directed feedback vertex set problem (DFVS) asks for a minimum set of nodes $V' \subseteq V$, such that the graph resulting from removing all nodes of V' from G is acyclic.

2 Reduction Rules

The following data reduction rules are used in the exact solver we submitted. Observe that DFVS on undirected instances degenerates into the well studied Vertex Cover problem. Thus, we can adopt rules for the Vertex Cover and Independent Set problems for DFVS.

2.1 Existing Kernelization Rules

We use the following kernelization rules previously described in literature:

- Rules 1 to 6 due to Fleischer et al. [4]
- *PIE Rule* due to Lin and Jou [8]
- *Dominance Rule* due to Fomin et al. [5] and Akiba and Iwata [2]
- *Unconfined vertices* due to Kneis et al. [7]
- *Crown Rule*, which is inspired by Abu-Khzam et al. [1]
- *Vertex folding* due to Chen et al. [3]
- *Twin Rule* due to Xiao and Nagamochi [9] as described by Hespe et al. [6]
- *Desk Rule* and *Funnel Rule* due to Hespe et al. [6]

2.2 Novel Kernelization Rules

Our solver also includes the following novel rules:

DiClique Rule

For every node $u \in G$, the DiClique Rule checks whether $N_{\text{und}}[u]$ is a clique. If additionally u is not part of a cycle in $G \setminus N_{\text{und}}(u)$, we add $N_{\text{und}}(u)$ to the DFVS and remove $N_{\text{und}}[u]$ from G. This rule is a variation of the CORE Rule from Lin and Jou [8].

DOMN Rule

If the contraction $G \times v$ of a node $v \in G$ does not add any new edges to the graph G, we remove node v from G.

Redundant Cycles Rule

This rule searches for small cycles of length two (undirected edges), three, and four. Let $C \subseteq V$ be such a cycle. If there exists an edge outside of C that is not part of any cycle every time any of C's nodes is deleted, we can delete that edge safely since we know that we have to delete at least one of C's nodes eventually.

C4 rule

We search four pairwise disjoint nodes u, v, w, x such that there are undirected edges $\{u, v\}$, $\{u, w\}$, $\{v, x\}$, and $\{w, x\}$. In such a structure, we have to delete at least $A = \{u, x\}$ or $B = \{v, w\}$. If –wlog– the nodes A are not in any cycle in $G \setminus B$, we add B to the DFVS and remove $A \cup B$.

3 Branch and Bound

We employ a branch-and-bound algorithm which maintains lower- and upper-bounds on the solution size to prune the search tree if both bounds agree (returning a minimal DFVS) or if the lower-bound exceeds the upper-bound (signalling an infeasible branch). All branching strategies are build around two base operations: We either add a node into the DFVS and recurse on $G \setminus u$ (delete branch) or exclude u from the DFVS and recurse on $G \times u$. We typically begin with the delete branch since a minimal DFVS of size k' for $G \setminus u$ also implies the lower-bound k' for $G \times u$. If some branch returned a feasible solution, we artificially decrease the upper bound for subsequent branches on that parent such that they either yield a strictly smaller solution or can be easier pruned.

In each step of the branch-and-bound algorithm, we first apply kernelization rules. If the remaining graph is not strongly connected, we process the SCCs independently. We consider the SCCs in increasing size as solution size bounds inherited from the parent can only be forwarded to the last SCCs.

Next, we attempt to forcefully split the instance into smaller subproblems by finding small vertex cuts to branch on. Here, a node u with $N_{\text{in}}(u) = N_{\text{und}}(u)$ or $N_{\text{out}}(u) = N_{\text{und}}(u)$ is especially valuable since we either need to remove u or its neighbors $N_{\text{und}}(u)$ from G which both cuts the graph.

In the next two steps we use by-products of the lower-bound heuristic:

If a maximal clique $C \subseteq V$ with $|C| > 2$ was found, we branch on it. Observe that for any DFVS D we have $|D \cap C| \geq |C| - 1$ leading to $|C| + 1$ branches. In the first branch, we delete all nodes C to obtain an almost tight lower bound. The remaining $|C|$ branches (one for each node in C to be spared) can only improve the total solution size by one, i.e. if one returns a solution, we can prune the remaining branches.

Otherwise, the lower-bound heuristic may return a shortest cycle on which we can branch (using the same machinery we use to branch on cut-vertices).

If all other attempts failed, we branch on a single vertex with high degree.

3.1 Lower Bound

Working on a copy of the graph, we execute the following rules exhaustively in that order:

- Each node with a self-loop is removed, increasing the lower bound by 1
- Cliques of size $k > 2$ are removed , increasing the lower bound by $k - 1$
- Undirected paths of length $k \geq 4$ are removed, increasing the lower bound by $k/2$
- For $k = 2, ..., 10$ apply exhaustively: iteratively search and remove a cycle of length k, increasing the lower bound by 1

References

1 Faisal N. Abu-Khzam, Rebecca L. Collins, Michael R. Fellows, Michael A. Langston, W. Henry Suters, and Christopher T. Symons. Kernelization algorithms for the vertex cover problem: Theory and experiments. In Lars Arge, Giuseppe F. Italiano, and Robert Sedgewick, editors,

Proceedings of the Sixth Workshop on Algorithm Engineering and Experiments and the First Workshop on Analytic Algorithmics and Combinatorics, New Orleans, LA, USA, January 10, 2004, pages 62–69. SIAM, 2004.

2 Takuya Akiba and Yoichi Iwata. Branch-and-reduce exponential/fpt algorithms in practice: A case study of vertex cover. *Theor. Comput. Sci.*, 609:211–225, 2016. doi:10.1016/j.tcs.2015.09.023.

3 Jianer Chen, Iyad A. Kanj, and Weijia Jia. Vertex cover: Further observations and further improvements. *J. Algorithms*, 41(2):280–301, 2001.

4 Rudolf Fleischer, Xi Wu, and Liwei Yuan. Experimental study of FPT algorithms for the directed feedback vertex set problem. In Amos Fiat and Peter Sanders, editors, *Algorithms - ESA 2009, 17th Annual European Symposium, Copenhagen, Denmark, September 7-9, 2009. Proceedings*, volume 5757 of *Lecture Notes in Computer Science*, pages 611–622. Springer, 2009. doi:10.1007/978-3-642-04128-0_55.

5 Fedor V. Fomin, Fabrizio Grandoni, and Dieter Kratsch. A measure & conquer approach for the analysis of exact algorithms. *J. ACM*, 56(5):25:1–25:32, 2009. doi:10.1145/1552285.1552286.

6 Demian Hespe, Sebastian Lamm, Christian Schulz, and Darren Strash. Wegotyoucovered: The winning solver from the PACE 2019 challenge, vertex cover track. In *CSC*, pages 1–11. SIAM, 2020.

7 Joachim Kneis, Alexander Langer, and Peter Rossmanith. A fine-grained analysis of a simple independent set algorithm. In Ravi Kannan and K. Narayan Kumar, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2009, December 15-17, 2009, IIT Kanpur, India*, volume 4 of *LIPIcs*, pages 287–298. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2009. doi:10.4230/LIPIcs.FSTTCS.2009.2326.

8 Hen-Ming Lin and Jing-Yang Jou. Computing minimum feedback vertex sets by contraction operations and its applications on CAD. In *Proceedings of the IEEE International Conference On Computer Design, VLSI in Computers and Processors, ICCD '99, Austin, Texas, USA, October 10-13, 1999*, page 364. IEEE Computer Society, 1999. doi:10.1109/ICCD.1999.808567.

9 Mingyu Xiao and Hiroshi Nagamochi. Confining sets and avoiding bottleneck cases: A simple maximum independent set algorithm in degree-3 graphs. *Theor. Comput. Sci.*, 469:92–104, 2013.