Journal of Electronic Systems

DLINE JOURNALS

Print ISSN: 2278 – 652X
Online ISSN: 2278 – 6538

JES 2025: 15 (2)
https://doi.org/10.6025/jes/2025/15/2/113-117

Effective Solvers for the Directed Feedback Vertex Sets

Rafael Kiesel
TU Wien, Austria

André Schidler
TU Wien, Austria

ABSTRACT

We present the solver DAGer for the Directed Feedback Vertex Set (DFVS) problem, as it was entered in the exact category of the 2022 PACE Challenge. Our method initially employs a variety of preprocessing strategies that incorporate both established data reduction techniques for DFVS and innovative modifications derived from the vertex cover problem. In terms of the actual solving process, we discovered that utilizing a MaxSAT solver with incremental constraints yields impressive results.

Keywords: Effective Solvers, Vertex sets, Data Reduction Techniques

Received:3 November 2024, Revised 24 January 2025, Accepted 1 March 2025

1. Introduction

This paper describes the solver DAGer[1], which we submitted to the exact track of the 2022 PACE Challenge [9]. This year the challenge was to solve the Directed Feedback Vertex Set (DFVS) problem. Informally, the DFVS problem is, given a directed graph $G = (V, A)$ to find a minimum cardinality subset $D \subseteq V$ such that every directed cycle of G uses at least one vertex from D. The problem is one Karp's original 21 NP-complete problems [6] and has a wide range of applications such as for argumentation frameworks [4, 3], deadlock detection, program verification and VLSI chip design [10]. It is known that the problem is fixed-parameter tractable [2] in the size of the DFVS.

The runtime of these parameterized algorithms quickly increases with increasing size of the DFVS, making alternative approaches necessary for solving instances with a large DFVS, like many of the instances in these year's PACE Challenge. We use an alternative approach to developing a dedicated DFVS algorithm: we express the problem in terms of constraints and use a constraint solver for computing the DFVS. Our implementation uses a propositional satisfiability (SAT) solver as a constraint solver. Unfortunately, the size of a direct encoding in propositional logic is, depending on the encoding, cubic or exponential in the size of the input graph, and therefore prohibitively large. We therefore use a lazy approach: we use an adapted SAT solver that creates the necessary constraints when they are violated, keeping the total number of constraints low enough for most of the instances in the public instance set. We thereby exploit the following observation by [1] in a similar context. Namely, while there may be exponentially many cycles in a directed graph, in order to prove that we need at least k vertices to cover every cycle, we may need far fewer.

Data reductions are another crucial component of DAGer, as they transform the given graph into a smaller one that is easier to solve. Notably, apart from the well-known standard data reductions for DFVS of [7] and [8], we also managed to non-trivially generalize many data reductions for the Vertex Cover (VC) problem to DFVS.

2. Preliminaries

We denote an undirected graph as $H = (V, E)$ with vertices V and edges E, and a directed graph, or *digraph*, by $G = (V, A)$ with arcs A. Further, we denote an undirected edge between vertices u and v as $\{u, v\}$ and the arc, or directed edge, from u to v as (u, v). We can now introduce the central problem addressed in this paper.

Definition 1 (Cycle, DFVS). *Given a digraph $G = (V, A)$ a path is a list of vertices $v_1, \ldots, v_n$ such that for $i = 1, \ldots, n - 1$ there exists an arc $(v_i, v_i+1) \in A$. A cycle is a path $v_1, \ldots, v_n$ such that $v_1 = v_n$. Furthermore, a cycle is uncovered, if there is no cycle $v'_i, \ldots, v'_m$ such that $\{v'_i \mid i = 1, \ldots, m\} \subsetneq \{vi \mid i = 1, \ldots, n\}$. A* Directed Feedback Vertex Set (DFVS) *of G is a set $D \subseteq V$ of minimum cardinality such that every cycle of G contains at least one vertex in D.*

We use the notation $C(G)$ to refer to the set of all uncovered cycles in G. Since any self-loop (u, u) can be easily preprocessed, by adding u to the DFVS and removing u from the digraph, we will henceforth assume that the input digraph is self-loop-free.

We can represent the problem of finding a DFVS in terms of another problem.

Definition 2 (Hitting Set). *Given a set V called the universe and a set $C = \{C_1, \ldots, C_n\}$ with $Ci \subseteq V$ for $1 \leq i \leq n$, the minimum hitting set problem asks for a set $D \subseteq V$ of minimum cardinality such that $D \cap C_i \neq \phi$ for all $1 \leq i \leq n$.*

Given a digraph G, a hitting set for V and $C(G)$ is also a DFVS for G. In the special case where each uncovered cycle has length 2, we can also express DFVS as follows:

Definition 3 (Vertex Cover). *Let $H = (V, E)$ be an undirected graph. A minimum vertex cover is a set of vertices $D \subseteq V$, such that D is of minimum cardinality and for each edge $\{u, v\} \in E$ either $u \in D$ or $v \in D$.*

3. Algorithm

Our solver DAGer works via reduction of the DFVS problem to the hitting set problem. Given the set of cycles $C(G)$, we can easily express that each cycle needs to be *hit* by the DFVS in propositional logic. This propositional encoding has one variable per vertex and the corresponding vertex is in the DFVS if and only if the variable is true. We represent each cycle as a disjunction, as at least one vertex per cycle must be in the DFVS, and connect all cycles by a conjunction. We ensure the minimality of the DFVS by using a MaxSAT solver: besides satisfying the clauses representing the cycles, the MaxSAT solver maximizes the number of satisfied *soft clauses*. We add for each variable the negation as a soft clause, causing the MaxSAT solver to minimize the number of variables set to true.

As the graph might have too many cycles to efficiently enumerate them all, we proceed in a lazy fashion: while computing a hitting set for a subset of the cycles we dynamically check for additional cycles that do not intersect the hitting set. More precisely, DAGer works as follows:

1. Preprocess the graph (See Section 3.1)

2. Identify a set $C'(G) \subseteq C(G)$ of uncovered short cycles, we use a maximum length of 8.

3. If $C'(G) = C(G)$, perform additional preprocessing not performed in Step 1 (Section 3.1).

4. Find a minimum hitting set D for $C'(G)$ using a modified MaxSAT solver (See Section 3.2).

5. If an additional cycle is found while solving, add it.

We will briefly discuss our preprocessing and our changes to the MaxSAT solver.

3.1 Preprocessing

From classic DFVS preprocessing we used the data reductions INDICLIQUE, OUTDICLIQUE, DICLIQUE-2 and DICLIQUE-3 from [7], as well as the data reductions PIE and DOME from [8]. Apart from that, we adapted preprocessing techniques from the vertex cover setting to the DFVS problem. Here, we used the data reductions 1, 2, 3, 4, 5, 6 and 7.2 from [11], as well as data reductions 8 and 10.1 from [5]. Their soundness for DFVS is based of the following observation: Given a directed graph G that only has uncovered cycles of length 2, i.e., each cycle has the form u, v, u and let $G' = (V, E')$ such that $E' = \{(u, v) |$ there is a cycle u, v, u in $G\}$. A minimum vertex cover for G' is a minimum DFVS for G.

Clearly, in the above case we can apply all vertex cover data reductions on the undirected graph. Furthermore, if we know all uncovered cycles, we can apply vertex cover preprocessing locally, whenever all vertices involved in the reduction only take part in uncovered cycles of length 2. This case constitutes Step 3. of the algorithm, we described above.

However, it is not always the case that we know all cycles. Therefore, we integrated all the aforementioned vertex cover preprocessing techniques also into the preprocessing in Step 1. It follows from the above reasoning that if all vertices involved in a reduction only have bidirectional edges, then we can apply it. There are additional cases in which we can additionally apply vertex cover preprocessing, however, a detailed analysis of these cases requires a lot of technical details and is therefore left out due to space reasons. The underlying intuition is simple though. We can apply a VC reduction if (1) it would be applicable, in the undirected graph that has an edge $\{u, v\}$ whenever u and v are bidirectionally connected in the original graph and (2) the application of the reduction does not lead to new subset-minimal cycles.

3.2 MaxSAT and Cycle Check

The SAT solver used inside the MaxSAT[2] solver tries to find satisfying assignment (equivalent to a DFVS) by repeating the following steps, until all variable values have been set:

1. Decide on a variable and value, i.e., decide to add a vertex to the DFVS or leave it in the graph.

2. Propagate values that are implied by the decision, e.g., if all but one vertices in a cycle have been left in the graph, add the remaining vertex to the DFVS.

3. Check if any clause cannot be satisfied, i.e., a cycle where all vertices are left in the graph.

4. If yes, learn a clause to avoid making the same decisions and undo the corresponding decisions.

We extended the conflict check in Step 3: whenever the SAT solver decides to leave a vertex in the graph, we check if there is a cycle. If there is a cycle, we add the cycle to our encoding and notify the solver of the conflict. This way we do not have to find all the cycles initially and we only add those cycles that are required for finding a DFVS.

References

[1] Baharev, Ali., Schichl, Hermann., Neumaier, Arnold., Achterberg, Tobias. (2021). An exact method for the minimum feedback arc set problem. *Journal of Experimental Algorithmics (JEA)*, 26, 1–28.

[2] Chen, Jianer., Liu, Yang., Lu, Songjian., O'Sullivan, Barry., Razgon, Igor. (2008). A fixed-parameter algorithm for the directed feedback vertex set problem. In Cynthia Dwork (Ed.), *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008* (pp. 177–186). ACM.

[3] Dvorák, Wolfgang., Hecher, Markus., König, Matthias., Schidler, André., Szeider, Stefan., Woltran, Stefan. (2022). Tractable abstract argumentation via backdoor-treewidth. In *AAAI 2022*.

[4] Dvorák, Wolfgang., Ordyniak, Sebastian., & Szeider, Stefan. (2012). Augmenting tractable fragments of abstract argumentation. *Artificial Intelligence*, 186, 157–173. https://doi.org/10.1016/j.artint.2012.03.002

[5] Fellows, Michael R., Jaffke, Lars., Király, Aliz Izabella., Rosamond, Frances A., & Weller, Mathias. (2018). What is known about vertex cover kernelization? In Hans-Joachim Böckenhauer, Dennis Komm, Walter Unger (Eds.), *Adventures Between Lower Bounds and Higher Altitudes - Essays Dedicated to Juraj Hromkoviè on the Occasion of His 60th Birthday* (Vol. 11011, Lecture Notes in Computer Science, pp. 330–356). Springer. https://doi.org/10.1007/978-3-319-98355-4_19

[6] Karp, Richard M. (1972). Reducibility among combinatorial problems. In Raymond E. Miller & James W. Thatcher (Eds.), *Proceedings of a Symposium on the Complexity of Computer Computations, held March 20-*

[2]We use the solver *EvalMaxSAT* https://github.com/FlorentAvellaneda/EvalMaxSAT

22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA, The IBM Research Symposia Series (pp. 85–103). Plenum Press, New York. https://doi.org/10.1007/978-1-4684-2001-2_9

[7] Lemaic, Mile. (2008). *Markov-chain-based heuristics for the feedback vertex set problem for digraphs* (PhD thesis). Universität zu Köln.

[8] Lin, Hen-Ming., Jou, Jing-Yang. (2000). On computing the minimum feedback vertex set of a directed graph by contraction operations. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 19(3), 295–307.

[9] Schulz, Christian., Großmann, Ernestine., Heuer, Tobias., Strash, Darren. (2022). *Pace 2022*. Retrieved from https://pacechallenge.org/2022/

[10] Silberschatz, Abraham., Gagne, Greg., Galvin, Peter B. (2018). *Operating System Concepts*. JW Wiley.

[11] Stege, Ulrike., Fellows, Michael Ralph. (1999). An improved fixed parameter tractable algorithm for vertex cover. *Technical report/Departement InformAatik, ETH Zürich*, 318.