

Solving Directed Feedback Vertex Set by Iterative Reduction to Vertex Cover

Sebastian Angrick ✉

Hasso Plattner Institute,
Universität Potsdam, Germany

Katrin Casel ✉ 

Humboldt-Universität zu Berlin, Germany

Tobias Friedrich ✉ 

Hasso Plattner Institute,
Universität Potsdam, Germany

Theresa Hradilak ✉

Hasso Plattner Institute,
Universität Potsdam, Germany

Otto Kißig ✉ 

Hasso Plattner Institute,
Universität Potsdam, Germany

Leo Wendt ✉


Hasso Plattner Institute,
Universität Potsdam, Germany

Ben Bals ✉

Hasso Plattner Institute,
Universität Potsdam, Germany

Sarel Cohen ✉ 

The Academic College of Tel Aviv-Yaffo, Israel

Niko Hastrich ✉ 

Hasso Plattner Institute,
Universität Potsdam, Germany

Davis Issac ✉ 

Hasso Plattner Institute,
Universität Potsdam, Germany

Jonas Schmidt ✉

Hasso Plattner Institute,
Universität Potsdam, Germany

Abstract

In the *Directed Feedback Vertex Set* (DFVS) problem, one is given a directed graph $G = (V, E)$ and wants to find a minimum cardinality set $S \subseteq V$ such that $G - S$ is acyclic. DFVS is a fundamental problem in computer science and finds applications in areas such as deadlock detection. The problem was the subject of the 2022 PACE coding challenge. We develop a novel exact algorithm for the problem that is tailored to perform well on instances that are mostly bi-directed. For such instances, we adapt techniques from the well-researched vertex cover problem. Our core idea is an iterative reduction to vertex cover. To this end, we also develop a new reduction rule that reduces the number of not bi-directed edges. With the resulting algorithm, we were able to win third place in the exact track of the PACE challenge. We perform computational experiments and compare the running time to other exact algorithms, in particular to the winning algorithm in PACE. Our experiments show that we outpace the other algorithms on instances that have a low density of uni-directed edges.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms

Keywords and phrases directed feedback vertex set, vertex cover, reduction rules

Digital Object Identifier 10.4230/LIPIcs.SEA.2023.10

Supplementary Material *Software (Source Code)*: <https://doi.org/10.5281/zenodo.6645235>

Software (Source Code): <https://github.com/BenBals/mount-doom/tree/exact>
archived at `swh:1:dir:a8ce8a824241821bdf98f5380594c74d2d6c327`

1 Introduction

In the DIRECTED FEEDBACK VERTEX SET (DFVS) problem, we are given a directed graph $G = (V, E)$ and the objective is to find a minimum cardinality set $S \subseteq V$ such that $G - S$ is acyclic. DFVS is a fundamental computational problem that appeared in Karp's seminal paper [19]. The problem is equivalent to the FEEDBACK ARC SET (FAS) problem where we want to delete edges instead of vertices i.e., there are reductions in both directions that preserve the value of the solution and blow up the graph size only polynomially. Both



© Sebastian Angrick, Ben Bals, Katrin Casel, Sarel Cohen, Tobias Friedrich, Niko Hastrich, Theresa Hradilak, Davis Issac, Otto Kißig, Jonas Schmidt, and Leo Wendt;
licensed under Creative Commons License CC-BY 4.0

21st International Symposium on Experimental Algorithms (SEA 2023).

Editor: Loukas Georgiadis; Article No. 10; pp. 10:1–10:14



Leibniz International Proceedings in Informatics

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

problems have applications in areas such as deadlock detection [13], compiler optimization [34], program verification [31], VLSI chip design [5], Computer Aided Design (CAD) [23], and chemical engineering [2].

From a theoretical point of view, DFVS is NP-hard, thus polynomial exact algorithms are very unlikely. The currently best known theoretical runtime known in terms of input size n for an exact algorithm is the $\mathcal{O}(1.9977^n)$ algorithm by Razgon [28], only barely beating the trivial $\mathcal{O}(2^n)$ time brute-force algorithm. From the perspective of parameterized algorithms (see [8] for an introduction to parameterized complexity), the current best fixed parameter tractable algorithm for DFVS is a $4^k k! n^{\mathcal{O}(1)}$ time algorithm by Chen et al. [7], where the parameter k is the size of the directed feedback vertex set. The best known polynomial time approximation algorithm known for DFVS is a $\mathcal{O}(\log k \log \log k)$ approximation by Even et al. [12].

From a practical point of view, there are only few recent exact algorithms that can be used for large general instances. Historically, Smith and Walford [32] gave the first exact algorithm for DFVS in 1975 but their algorithm can only handle small graphs [21]. There are also a few *branch and bound* based exact algorithms. Chakradhar et al. [5] in 1995 gave a branch and bound exact algorithm using an ILP relaxation for finding DFVS of flip-flop dependency graphs arising from sequential circuits. Orenstein et al. [25], also in 1995, used branch and bound to find optimal DFVS in digital circuit graphs and tested them on ISCAS89 benchmarks. Lin and Jou [23] in 1999 gave an improved *branch and bound* exact algorithm together with reduction rules, and experimentally evaluated them on the ISCAS89 benchmarks coming from CAD applications.

More recent exact algorithms use different techniques. Fleischer et al. [13] experimentally evaluate Chen's algorithm and some reduction rules. Based on their experiments, the algorithm already becomes impractical when the solution size k is as small as 10. Baharev et al. [2] gave an integer-program based exact algorithm for Feedback Arc Set that uses a lazy enumeration of cycles, and can be also applied to DFVS. To apply an FAS algorithm to DFVS, one can split each vertex into an in-vertex and out-vertex with a uni-directed edge between them.

Some heuristic algorithms have been also developed for DFVS e.g, based on greedy randomized adaptive search [26] (1998), markov chains [22] (2008), and local search [15] (2013). We do not describe them in further detail as exact algorithms are the focus of this work.

Perhaps also due to this lack of fast exact algorithms for DFVS, it was selected as the problem for the PACE 2022 coding challenge [30].

A much more extensively studied problem, that was also subject of the PACE challenge in 2019 [10], is VERTEX COVER. It is closely related to other important problems such as MAXIMUM CLIQUE, MAXIMUM INDEPENDENT SET, ODD CYCLE TRANSVERSAL etc. Often, a formulation in terms of vertex cover have been useful in solving these problems [24].

From a theoretical point of view, VERTEX COVER is "easier" than DFVS, since there is an easy reduction from VERTEX COVER to DFVS that just replaces undirected edges with bi-directional arcs. This is also reflected in the much better theoretical exact algorithms. The current best exact algorithm measured with respect to the number of vertices n runs in $\mathcal{O}(1.1996^n)$ [35]. Also, just recently, the parameterized $1.2738^k n^{\mathcal{O}(1)}$ algorithm by Chen, Kanj and Xia [6] was improved to $1.25298^k n^{\mathcal{O}(1)}$ by Harris and Narayanaswamy [17].

Also, and perhaps even more so, there are much more powerful practical solvers for VERTEX COVER than for DFVS. VERTEX COVER has a straight forward ILP formulation that can be then fed into commercial ILP solvers such as Gurobi and CPLEX. Akiba and

Iwata [1] gave a branch and reduce algorithm and showed it to be competitive with the CPLEX solver. Faster solvers were developed as a result of the PACE 2019 challenge [10]. The winning solver is the `WeGotYouCovered` solver by Hesse et al. [18] which uses branch and reduce and branch and bound. Later, Plachetta and van der Grinten [27] developed a branch and reduce algorithm using a SAT solver that was competitive with the PACE winner. Stallmann, Ho and Goodrich [33] enhanced the solver by Akiba and Iwata [1] using targeted reductions depending on the instance profile. Also, vertex cover has extremely fast heuristic solvers, e.g. [4]. Thus, our approach to DFVS is to take advantage of this wealth of results for VERTEX COVER.

In a recent ALENEX 2023 paper [20], Kiesel and Schidler describe their DAGer algorithm for DFVS, which won the first place in the PACE 2022 challenge [29]. In a nutshell, their algorithm first applies an extensive set of preprocessing rules to reduce the size of the instance. The reduced instance is then solved exactly using a modified SAT-solver. As initialization, a set of clauses corresponding to cycles that need to be hit, is given. This set might not be exhaustive. To ensure feasibility, the SAT-solver is modified to maintain a topological ordering on the vertices and dynamically detect cycles and adding the corresponding clauses to the problem. They refer to this approach as Cycle Propagation, which builds on the idea that already a limited number of the constraints in a propositional encoding is usually sufficient for finding an optimal solution, thus their algorithm starts with a small number of constraints and cycle propagation adds additional constraints when necessary. Our approach deviates from DAGer, as we describe in the following section.

Our Contribution

We develop a novel exact algorithm `Mount-Doom` for the DFVS problem. Our approach is different from all the previous exact as well as heuristic approaches. We develop a method to iteratively reduce DFVS to the VERTEX COVER problem and exploit fast existing algorithms for this problem. In order to strategically reduce to VERTEX COVER, we also develop a new reduction rule called `SHORTONE` that reduces the number of uni-directed edges.

With these strategies, `Mount-Doom` was able to secure third place in the exact track of PACE 2022 challenge.

We perform experiments to compare our algorithm to the PACE winning solver DAGer [20, 29] and the state-of-the-art ILP based solver `Sdopt` [2] for DFAS. We use the complete set of 200 instances from the PACE competition as well as our own randomly generated instances, and observe that in a fair share of the instances our running times are significantly better. As characteristic of instances in which our algorithm performs better, we consider the density of uni-directed edges. This measure can be seen as a distance to the DFVS instance being a VERTEX COVER instance. We show experimentally that when the density of uni-directed edges is small then we are considerably faster than other algorithms in the PACE instances.

We also perform another set of experiments to demonstrate the utility of our new reduction rule `SHORTONE`. Our experiments reveal that in many of the PACE instances there is significantly more reduction in the instance size by using this new rule in addition to the existing reduction rules. On our less structured set of generated graphs (Erdős-Rényi graphs), we are more often outperformed by the competitors. This indicates that our algorithm heavily takes advantage of structured input.

2 Preliminaries

We use $G = (V, E)$ to denote directed and undirected graphs. For any $S \subseteq V$ we write $G - S$ for the graph obtained from G by deleting all vertices in S together with their adjacent edges. With these notions, the Directed Feedback Vertex Set problem (DFVS for short) is the task to find a minimum cardinality subset $S \subseteq V$ such that $G - S$ is acyclic.

We call an edge $vw \in E$ *bi-directed* if $wv \in E$. Let $PIE \subseteq E$ be the set of all bi-directed edges and $DIR = E \setminus PIE$. We define $G[PIE]$ to be the *undirected* graph obtained from G by deleting all edges in DIR , and replacing any pair of bi-directed edges by an undirected edge. Further, we define the graph $G[DIR] = (V, DIR)$, the subgraph of G containing only the not bi-directed edges. We call these edges *uni-directed*. Therefore, we define the uni-directed edge density of a graph as the number of uni-directed edges divided by the number of possible edges $\binom{|V|}{2}$. We use $uv \in E$ to denote an uni-directed edge from u to v , and $\{u, v\}$ for a bi-directed edge.

For any $v \in V$, $N^+(v)$ denotes the set of outgoing neighbors of v , i.e. $N^+(v) = \{u \mid uv \in E\}$. Similarly, $N^-(v)$ denotes the set of incoming neighbors. We define the set of bi-directed neighbors $N(v)$ of v as the set of its neighbors in $G[PIE]$, formally $N(v) = \{u \mid uv \in E \wedge vu \in E\}$. Additionally, we call $D \subseteq V$ a *diclique*, if $D \setminus \{u\} \subseteq N(u)$ for each $u \in D$.

For $v \in V$, we define the graph obtained from G by *shortcutting* v as the graph $G' = (V', E')$ with vertex set $V' = V \setminus \{v\}$ and edge set $E' = (E \cap (V' \times V')) \cup (N^-(v) \times N^+(v))$. Note that in the context of the DFVS problem, shortcutting v corresponds to the assumption that v is not part of the solution.

3 Mount Doom Solver

As already mentioned, our overall idea is to profit from the plethora of results obtained for the VERTEX COVER problem. We do this both by adapting known reduction rules for and also by a direct reduction to VERTEX COVER.

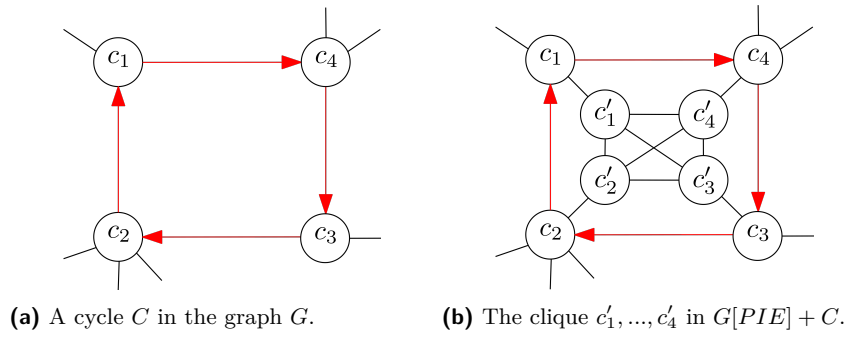
3.1 Reduction to Vertex Cover

Our reduction to VERTEX COVER relies on the simple but surprisingly powerful observation, that any directed feedback vertex set has to in particular cover all bi-directed edges. In the special case of a graph that only contains bi-directed edges, DFVS is equivalent to VERTEX COVER. More generally, one can observe the following relation to the underlying undirected graph; recall that we defined $G[PIE]$ to be the graph obtained from G by replacing bi-directed edges by undirected edges, and then deleting the remaining directed edges.

► **Observation 1.** *If S is a minimum vertex cover for $G[PIE]$ and $G - S$ is acyclic, then S is a minimum feedback vertex set for G .*

This observation implies that if we happen to find a feedback vertex set by solving the VERTEX COVER problem on $G[PIE]$, then we have solved DFVS on G . On the other hand, if such a minimum vertex cover S for $G[PIE]$ is not a feedback vertex set for G , then there exists a cycle in $G[DIR]$ that is not covered by S . For our reduction, we add a *clique gadget* to such cycles.

Formally, let $G = (V, E)$ be an undirected graph and let $C \subseteq V$. Our goal is to create a graph G' such that that any minimum vertex cover for G' is a vertex cover for G and also contains a vertex in C . Let C be a set of r vertices c_1, \dots, c_r . For our clique gadget, we



■ **Figure 1** An example reduction to vertex cover. Red denotes uni-directed and black denotes bi-directed edges.

add r new vertices c'_1, \dots, c'_r to G , turn them into a clique, and add the edges $\{c_i, c'_i\}$ for all $1 \leq i \leq r$. We denote this operation by $G + C$. See Figure 1 for an example application. We first prove an exchange argument for vertex covers for $G + C$.

► **Lemma 2.** *For any cycle C in $G[DIR]$ and vertex cover S for $G[PIE] + C$, a vertex cover S' for $G[PIE] + C$ with $|S'| \leq |S|$ and such that $C \cap S' \neq \emptyset$ can be created in linear time.*

Proof. Assume that S contains no vertices of $C = \{c_1, \dots, c_r\}$, as otherwise we are already done. In order to cover the edges $\{c_i, c'_i\}$ for $1 \leq i \leq r$, S then has to contain all vertices c'_i , $1 \leq i \leq r$ added by the clique gadget. To create S' from S , we replace c'_1 by c_1 . This choice of S' obviously satisfies $|S'| \leq |S|$. Further, S' is also a vertex cover for $G[PIE] + C$, since $N(c'_1) = \{c_1, c'_2, \dots, c'_r\} \subseteq S'$ and thus all edges that were covered by S through c'_1 remain covered by S' . ◀

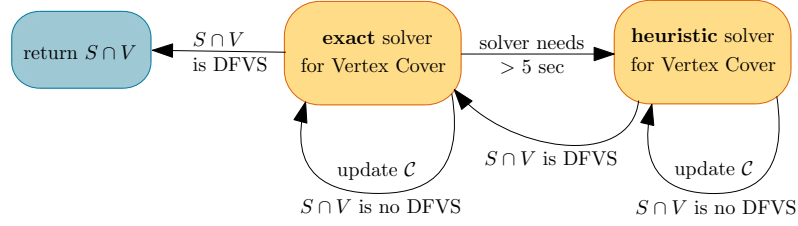
With this exchange argument, we can assume that adding the clique gadget results in a vertex cover for $G[PIE] + C$ that contains at least one vertex from C . The overall idea of our reduction to VERTEX COVER is to iteratively add clique gadgets for so far not covered cycles, until we find a feedback vertex set for the original graph.

Consider any set of cycles $\mathcal{C} = \{C_1, C_2, \dots, C_q\}$ in $G[DIR]$. Denote the graph resulting from iteratively applying the clique gadget for each cycle in \mathcal{C} by $G[PIE] + \mathcal{C}$. Note that with this gadgetry we add multiple *disjoint* sets C'_i of new vertices. Formally, for $C_i = \{c_1^i, \dots, c_{r_i}^i\}$, we add new vertices $C'_i = \{d_1^i, \dots, d_{r_i}^i\}$, add edges to turn C'_i into a clique, and add edges $\{c_j^i, d_j^i\}$ for all $1 \leq j \leq r_i$, for each $1 \leq i \leq q$. For this iterative application of the clique gadget, we observe the following.

► **Lemma 3.** *For any set of $\mathcal{C} = \{C_1, C_2, \dots, C_q\}$ in $G[DIR]$ and minimum vertex cover S for $G[PIE] + \mathcal{C}$, if $G - S$ is acyclic, then $S \cap V$ is a minimum feedback vertex set for G .*

Proof. Let S^* be a minimum feedback vertex set for G . We claim that $|S^*| \geq |S \cap V|$, which proves the lemma. Since S^* is a feedback vertex set for G , it follows that S^* is a vertex cover for $G[PIE]$, and contains at least one vertex from each cycle in \mathcal{C} . (Note that any edge in $G[PIE]$ and any cycle in $G[DIR]$ corresponds to a cycle in G that needs to be covered.) Assume w.l.o.g. that S^* contains vertex c_1^i for each C_i .

Let $S_{\mathcal{C}}^*$ be the set created from S^* by adding $d_2^i, \dots, d_{r_i}^i$ for each i , $1 \leq i \leq q$. Observe that $S_{\mathcal{C}}^*$ then is a vertex cover for $G[PIE] + \mathcal{C}$. The vertices added from the C'_i cover the clique-edges among the C'_i , and all edges $\{c_j^i, d_j^i\}$ with $j \geq 2$. Since S^* contains vertex c_1^i , also the edge $\{c_1^i, d_1^i\}$ is covered for each i .



■ **Figure 2** Overview of the reduction to Vertex Cover.

Since S is a minimal vertex cover for $G[PIE] + \mathcal{C}$, it follows that $|S_{\mathcal{C}}^*| \geq |S|$. At last, note that $S \cap V$ contains at least $r_i - 1$ out of the r_i vertices in C'_i for each i , since these build a clique in $G[PIE] + \mathcal{C}$ that needs to be covered. Also, we added exactly $r_i - 1$ out of the r_i vertices in C'_i for each i to S^* to create $S_{\mathcal{C}}^*$.

$$\text{Thus } |S \cap V| \leq |S| - \sum_{i=1}^q (r_i - 1) \leq |S_{\mathcal{C}}^*| - \sum_{i=1}^q (r_i - 1) = |S^*|. \quad \blacktriangleleft$$

Building from this idea, we iteratively choose new cycles and add their corresponding clique gadgets, growing a set \mathcal{C} . By using the exchange argument from Lemma 2 on the vertex cover for $G[PIE] + \mathcal{C}$, we are certain that this process converges to finding a directed feedback vertex set for the original graph, at the latest when \mathcal{C} contains all cycles in G . We start with $\mathcal{C} = \emptyset$, and iteratively do the following until we find a solution. Compute a minimum vertex cover S for $G[PIE] + \mathcal{C}$ and use the exchange argument from Lemma 2 to create a set that contains at least one vertex for each cycle in \mathcal{C} . If $S \cap V$ is not a feedback vertex set for G , we compute a set of vertex disjoint cycles in $G[DIR] - S$ (simply with a depth first search) and add them to \mathcal{C} .

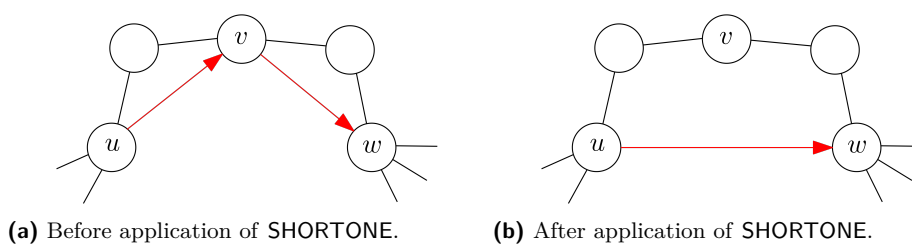
Since this iterative built up of \mathcal{C} can result in multiple calls to a VERTEX COVER solver, we do the following. If the exact VERTEX COVER solver takes more than five seconds, we switch to a heuristic solver to find a set \mathcal{C} that gives a feedback vertex set. (See Figure 2 for an overview of how we switch between solvers.) Then we run the exact VERTEX COVER solver only once on $G[PIE] + \mathcal{C}$ for this set \mathcal{C} to find an optimal solution. In theory, it could happen that the set \mathcal{C} by the heuristic solver gives a feedback vertex set does not work for the exact solver and we would have to iterate further, adding more cycles to \mathcal{C} . To our knowledge, this did not happen in our experiments.

3.2 Reduction Rules

Before reducing to VERTEX COVER, we apply a number of reduction rules to shrink the input. We adapt some reduction rules that were introduced for the VERTEX COVER problem, and also introduce a new rule that is tailored specifically for instances with many bi-directed and few not bi-directed edges.

We can again use that any feedback vertex set in particular is a vertex cover for $G[PIE]$ and inherit the following reduction rules from [14] for VERTEX COVER.

VC-DOME. Let u be isolated in $G[DIR]$ such that $N(u) \setminus \{v\} \subseteq N(v)$ for some $v \in N(u)$ (u is dominated by v in $G[PIE]$). Add v to the solution and delete it from G .



■ **Figure 3** An example application of the SHORTONE reduction rule. Red denotes uni-directed and black denotes bi-directed edges.

Degree 2 Fold. Let $v, u \in V$ be two isolated vertices in $G[DIR]$ (i.e. they are adjacent only to bi-directed edges in G). Also, let v have degree two in $G[PIE]$ with $N(v) = \{u, w\}$. Add a new vertex t to G and connect t in such a way, that $N^+(t) = N^+(w) \cup N(u)$ and $N^-(t) = N^-(w) \cup N(u)$ and remove u, v and w .

To unfold this reduction after a solution S is computed for the reduced graph do the following. If $t \in S$, the solution to the original instance is $(S \setminus \{t\}) \cup \{u, w\}$, and otherwise, the solution to the original instance is $S \cup \{v\}$.

Funnel Fold. Let v be isolated in $G[DIR]$ and $w \in N(v)$ such that $N(v) \setminus w$ is a diclique. Add $C = N(v) \cap N(w)$ to the solution and delete these vertices. Then add edges, such that each $x \in N(v) \setminus C$ is a bi-directed neighbor of every $y \in N(w) \setminus C$. Finally delete v and w .

To unfold this reduction after a solution S is computed for the reduced graph do the following. If $N(v) \setminus \{w\} \subseteq S$, the solution to the original instance is $S \cup \{w\}$, otherwise $S \cup \{v\}$.

Particularly for instances with low degree in $G[DIR]$, we design the following reduction rule to decrease the number of edges that are not bi-directed.

SHORTONE. Let v be a node with exactly one incoming edge uv and one outgoing edge vw in $G[DIR]$ such that $N(v) \subseteq N(u) \cup N(w)$ (any bi-directed neighbor of v is also a bi-directed neighbor of u or w). Remove the edges uv and vw and add uw .

► **Lemma 4.** *Reduction rule SHORTONE is correct.*

Proof. Let G be the original graph, and G' be the graph obtained from G by applying SHORTONE. First consider any directed feedback vertex set S for G . If $v \notin S$, then S is also a solution for G' . If $v \in S$, assume S is not a solution for G' . Then $u, w \notin S$ since any cycle that is present in G' and not in G was introduced by SHORTONE and hence must use uw . Since all bi-directed neighbors of v in G are also bi-directed neighbors of u or w in G , those must all be included in S . Thus we can simply replace v by u (or w) and obtain a solution for G' of the same size as S .

Conversely, consider any solution S' for G' and assume that it is not a solution for G . Thus $G - S'$ still contains a cycle C . Since S' is a solution for G' , C has to contain uv or vw since these are the only arcs deleted to create G' . Further, C cannot contain both uv and vw , since G' contains the added uw that could be used to turn C into a cycle in G' . Thus, C has to contain edge of the form vx or xv for some $x \notin \{u, w\}$. However, all possible choices of x are bi-directed neighbors of v in G' , and are thus contained in S' . ◀

See Figure 3 for an example application of the SHORTONE reduction rule.

Aside from these reduction rules imported or inspired by techniques developed for VERTEX COVER, we use and slightly alter some of those rules for DFVS.

These following two rules can be found in [22], where we do a slight novel alteration of the second one.

PIE. If uv is an edge between different strongly connected components in $G[DIR]$, then delete uv .

Improved CORE. A vertex a is a *core* of a diclique, if the graph induced by a and its neighbors is a diclique. Traditionally, one now deletes $N(a)$ from G since if S' is optimal for $G - N(v)$, $S' \cup N(v)$ is optimal for G . We proceed differently and shortcut the node a if $N^+(a)$ or $N^-(a)$ are dicliques. While this extension is easy to prove, it is, to the best of our knowledge, novel.

As first obvious reduction rules that are always applied, all nodes with self loops are collected into the solution and removed from the graph, and isolated nodes are removed as well. Further we use the classical domination rule, formally.

DOMe. An edge $ab \in DIR$ is called dominated if all outgoing neighbors of b are also outgoing neighbors of a or if all incoming neighbors of a are also incoming neighbors of b . It is well known (see e.g. [9]) that such a dominated edge can safely be deleted.

3.3 Implementation Details

We first only use the reduction rules Improved CORE and PIE, and deletion of self loops and isolated nodes. If we do not completely solve the instance within five seconds, using only these reductions, we proceed with all reduction rules listed above.

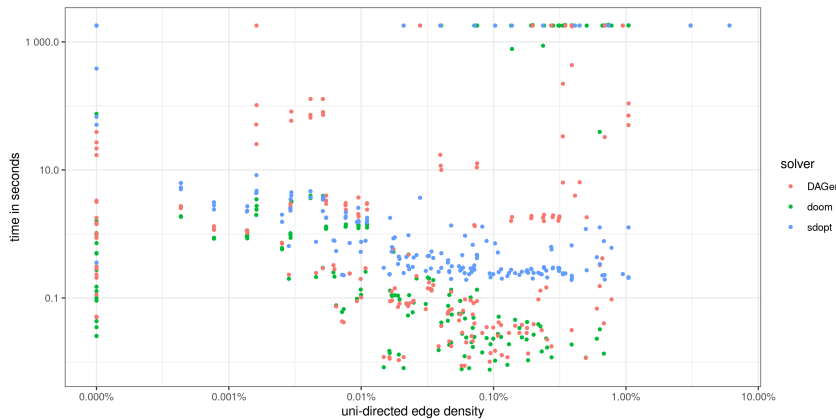
In both cases, if the instance is not solved after exhaustive application of all reduction rules, we proceed with our clique gadget to reduce to VERTEX COVER. To solve the VERTEX COVER instances $G[PIE] + \mathcal{C}$, we initially reduce them using the kernelization procedure, implemented by the winning solver of the 2019 PACE challenge [18]. For solving these reduced instances heuristically, we use the local-search solver NuMVC by Cai et al. [4]. When the heuristic solver has found a successful set of cycles \mathcal{C} , we compute an exact minimal vertex cover for $G[PIE] + \mathcal{C}$. For this exact computation, we use the solver SAT-and-Reduce by Plachetta and van der Grinten [27], which we augment by implementing better upper bounds using the aforementioned local-search solver (SAT-and-Reduce internally uses the SAT solver CaDiCaL by Biere et al. [3].)

4 Experiments

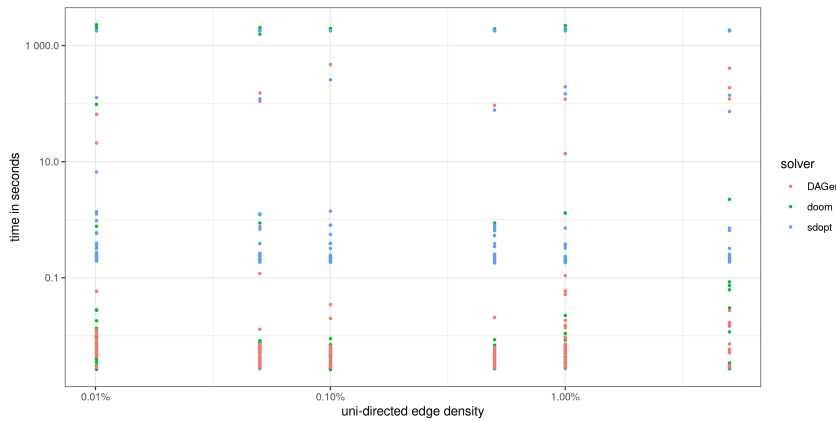
In this section we give the details of the computational experiments conducted to evaluate our solver Mount-Doom.

4.1 Data

We evaluate both the effect of our new reduction rule SHORTONE and the overall solver on two data sets. Firstly, we examine the performance on the public and private instances from the PACE 2022 challenge. Those are designed to provide a wide range of graph types with practical relevance [30].



■ **Figure 4** Running time on PACE graphs by density of uni-directed edges.



■ **Figure 5** Running time on Erdős-Rényi graphs by density of uni-directed edges.

Secondly, we evaluate on randomly generated graphs of varying sizes and densities. In the usual Erdős-Rényi model the number of vertices is fixed and each possible edge is present uniformly and independently at random with some constant probability [11]. Since we are interested in the effect of the uni-directed edge density on the solver performance, we modify this model as follows: For $n \in \mathbb{N}$ and $p, q \in \mathbb{R}^+$ s.t. $p + q \leq 1$ generate a graph $G(n, p, q)$ with n vertices letting each possible edge be a bi-directed edge with probability p , a single edge with probability q and not present with probability $1 - p - q$. In the case of single edges, pick one of the two possible directions with equal probability. We then examine different parameters, with n between 10 and 500, p and q between 0.0001 and 0.05. Overall, we generated 253 such graphs.

These two data sources help us gain a good understanding of two different aspects: The PACE instances provide large scale real-world relevant instance which contain up to a million edges while our random instances provide a more systematic view into the effect of the uni-directed edge density.

4.2 Competing Algorithms

We compare our algorithm with the winner of the PACE 2022 implementation challenge DAGer [20, 29] and a state-of-the-art-solver for the DIRECTED FEEDBACK ARC SET problem (DFAS) Sdopt [2].

The solver DAGer first applies an extensive set of preprocessing rules to reduce the size of the instance. The reduced instance is then solved exactly using a modified SAT-solver. As initialization, a set of clauses corresponding to cycles that need to be hit, is given. This set might not be exhaustive. To ensure feasibility, the SAT-solver is modified to maintain a topological ordering on the vertices and dynamically detect cycles and adding the corresponding clauses to the problem.

To use the solver Sdopt, which is designed to solve the DFAS problem, we first use the canonical reduction from the DFVS problem to the DFAS problem. Initially the solver enumerates a set of cycles and solves the integer programming formulation for hitting all of these cycles. If the obtained solution is a directed feedback arc set, this solution is returned, otherwise the current solution is augmented to a feasible DFAS and the found optimum is updated. Additionally, it searches for cycles that are not hit and adds some of them to the integer program relaxation. This process is repeated until a solution is found.

4.3 Machine Specifications

All experiments were conducted on an AMD EPYC 7742 at 2.25GHz CPU running Fedora 34. Both DAGer and Mount-Doom were compiled with GCC 11.3.1. Sdopt was run with Python 3.9 against Gurobipy 9.5.2 [16]. All experiments were conducted on a single thread.

4.4 Experiment Description

To examine the efficacy of our solver as a whole and the effect of our proposed reduction rule SHORTONE, we conducted two experiments.

For our first experiment, we take the two data sets described above and run the three described solvers (DAGer, Sdopt, Mount-Doom) on them.

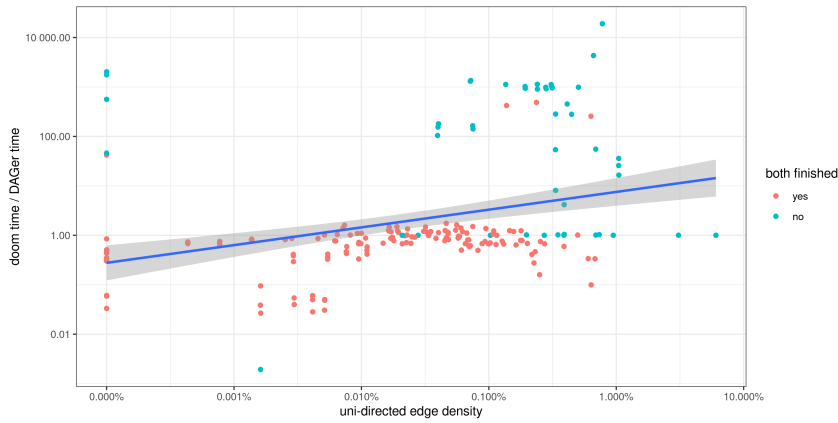
We hypothesize that on instances with a large number of bi-directed edges, our solver outperforms both competing solvers. To support this conclusion, the randomly generated graphs contain instances with a wide range of bi-directed and uni-directed edges and overall densities. The PACE instances also vary widely in these parameters.

In the second experiment, we want to examine the effect of our novel reduction rule SHORTONE. We compare the reduction achieved by our whole set of reduction rules to this set without SHORTONE on both of the data sets.

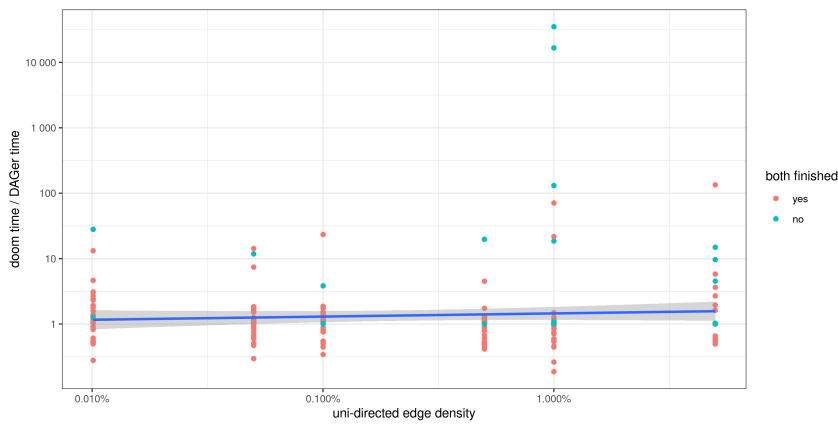
4.5 Results and Discussion

Next, we present the results of the conducted experiments and summarize our findings.

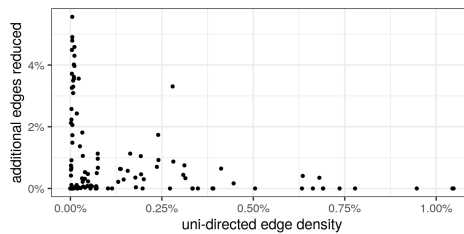
Figure 4 shows the running times of the three algorithms Mount-Doom, DAGer, and Sdopt on the PACE instances and Figure 5 shows the running times on the Erdős-Rényi graphs. It is apparent from the plots that Mount-Doom and DAGer outpace the Sdopt solver. Hence we make further one-to-one comparisons between DAGer and Mount-Doom in Table 1. Although DAGer clearly wins on the PACE instances in the total number of instances solved before the timeout, our Mount-Doom solver wins in the number of instances solved faster. On Erdős-Rényi graphs, DAGer also wins in the number of fast solved instances. This indicates that our solver is more suited for structured and less random instances.



■ **Figure 6** Ratio of Mount-Doom and DAGer running times on the PACE instances sorted by uni-directed edge density. On blue instances at least one of the solvers did not finish within 30 minutes. The unfinished solver’s time is denoted by 30 minutes. The line shows a linear regression model with a confidence interval of 95%.



■ **Figure 7** Ratio of Mount-Doom and DAGer running times on the Erdős-Rényi instances. On blue instances at least one of the solver’s did not finish within 30 minutes. The unfinished solvers time is denoted by 30 minutes. The line shows a linear regression model with a confidence interval of 95%.



■ **Figure 8** Percentage of total edges which are reduced with SHORTONE but not without by density of directed edges. The instances are taken from the set of all PACE instances after removing graphs which were fully reduced without SHORTONE (60 instances) and those that are not reduced at all even after including SHORTONE (2 instances).

■ **Table 1** Comparison of number of instances solver faster by each solver for different graph types.

Data set	Solver	#Solved	#Solved faster	#Solved faster by factor 2
PACE	Mount-Doom	153	109	46
	DAGer	186	78	38
Erdős-Rényi	Mount-Doom	166	75	17
	DAGer	213	102	28

The instances in the plots in Figure 4 and Figure 5 are ordered by the density of the uni-directed edges. We can see that in Figure 4 i.e. the PACE instances, we are in general faster than the other two algorithms in those instances with few uni-directed edges. Figure 6 shows the ratios between running times of Mount-Doom and DAGer with a similar X-axis and a linear regression line with a confidence interval of 95%. The regression line clearly shows the trend that Mount-Doom is better on low uni-directed density instances. But such a trend cannot be seen in Erdős-Rényi graphs from similar plots in Figures Figure 5 and Figure 7, again showing that Mount-Doom is better suited for structured instances.

Our SHORTONE reduction rule relies on the existence of a vertex v with exactly one incoming and one outgoing uni-directed neighbor, such that for every $w \in N(v)$ there is a uni-directed neighbor of v adjacent to w in $G[PIE]$. Therefore, when we have a sparse directed graph with relatively many bi-directed edges, we expect SHORTONE to be well applicable. This can be attested by Figure 8 showing the performance of SHORTONE on PACE instances. When we have only a small percentage of uni-directed edges, we can reduce a considerable percentage of initial edges which we do not reduce without SHORTONE. With up to 5% of the initial edges, we see a clear advantage compared to not using SHORTONE. This advantage decreases with increasing density of uni-directed edges. But even for a uni-directed edge density of about 1%, where $G[DIR]$ becomes quite dense, we are still able to reduce some edges.

However, on the generated Erdős-Rényi graphs, which exhibit a uniform edge distribution, the required edge structure for the rule to be applied is much less likely to occur. This was confirmed by experimental results, where the inclusion of SHORTONE did not reduce the instances any further.

5 Conclusion

We give the description of our new exact algorithm Mount-Doom for DFVS, which won the third place in the PACE coding challenge 2022. We also present the results of extensive experiments we conducted to compare Mount-Doom to the PACE winner DAGer and a state of art solver Sdopt. We demonstrated that we clearly outpace Sdopt in terms of running time. Although DAGer beats our algorithm in many cases, our algorithm is still competitive, and is considerably faster in a big share of the instances. Especially, on structured instances with low uni-directed edge density Mount-Doom has an upper hand.

We also introduced a new reduction rule SHORTONE in our algorithm. We demonstrated that the reduction rule comes to good use in many of the instances, especially the structured instances with small uni-directed edge density.

The efficiency of our solver seems to not just depend on the uni-directed edge density but even more on structure, as can be seen by the different performance between the PACE instances and the Erdős-Rényi graphs. As further work, it would be interesting to study more

closely which structures are beneficial not just for our particular algorithm, but generally for the approach to reduce DFVS to VERTEX COVER. One could, for example, also directly attempt to develop an algorithm that has theoretical efficiency in the low uni-directed density regime.

References

- 1 Takuya Akiba and Yoichi Iwata. Branch-and-reduce exponential/fpt algorithms in practice: A case study of vertex cover. *Theoretical Computer Science*, 609:211–225, 2016.
- 2 Ali Baharev, Hermann Schichl, Arnold Neumaier, and Tobias Achterberg. An exact method for the minimum feedback arc set problem. *ACM Journal of Experimental Algorithmics*, 26:1.4:1–1.4:28, 2021. doi:10.1145/3446429.
- 3 Armin Biere, Katalin Fazekas, Mathias Fleury, and Maximillian Heisinger. Cadical, kissat, paracooba, plingeling and treengeling entering the sat competition 2020. In *Proceedings of the SAT Competition 2020 – Solver and Benchmark Descriptions*, volume B-2020-1, pages 51–53, 2020.
- 4 Shaowei Cai, Kaile Su, Chuan Luo, and Abdul Sattar. NuMVC: An Efficient Local Search Algorithm for Minimum Vertex Cover. *Journal of Artificial Intelligence Research*, 46:687–716, 2013. doi:10.1613/jair.3907.
- 5 Srimat T Chakradhar, Arun Balakrishnan, and Vishwani D Agrawal. An exact algorithm for selecting partial scan flip-flops. *Journal of Electronic Testing*, 7(1):83–93, 1995.
- 6 Jianer Chen, Iyad A Kanj, and Ge Xia. Improved upper bounds for vertex cover. *Theoretical Computer Science*, 411(40-42):3736–3756, 2010.
- 7 Jianer Chen, Yang Liu, Songjian Lu, Barry O’Sullivan, and Igor Razgon. A Fixed-Parameter Algorithm for the Directed Feedback Vertex Set Problem. *Journal of the ACM*, 55:21:1–21:19, 2008.
- 8 Marek Cygan, Fedor V Fomin, Łukasz Kowalik, Daniel Lokshantov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*. Springer, 2015.
- 9 Reinhard Diestel. Graph Theory 3rd ed. *Graduate Texts in Mathematics*, 173, 2005.
- 10 M. Ayaz Dzulfikar, Johannes K. Fichte, and Markus Hecher. The PACE 2019 Parameterized Algorithms and Computational Experiments Challenge: The Fourth Iteration (Invited Paper). In *International Symposium on Parameterized and Exact Computation (IPEC)*, volume 148, pages 25:1–25:23, 2019. doi:10.4230/LIPIcs.IPEC.2019.25.
- 11 Paul Erdős and Alfréd Rényi. On random graphs. *Publicationes Mathematicae*, 6:290–297, 1959.
- 12 Guy Even, Joseph Naor, Baruch Schieber, and Madhu Sudan. Approximating minimum feedback sets and multicuts in directed graphs. *Algorithmica*, 20(2):151–174, 1998.
- 13 Rudolf Fleischer, Xi Wu, and Liwei Yuan. Experimental study of fpt algorithms for the directed feedback vertex set problem. In *European Symposium on Algorithms (ESA)*, pages 611–622. Springer, 2009.
- 14 Fedor V. Fomin, Fabrizio Grandoni, and Dieter Kratsch. A Measure & Conquer Approach for the Analysis of Exact Algorithms. *Journal of the ACM*, 56(5):25:1–25:32, 2009. doi:10.1145/1552285.1552286.
- 15 Philippe Galinier, Eunice Lemamou, and Mohamed Wassim Bouzidi. Applying local search to the feedback vertex set problem. *Journal of Heuristics*, 19(5):797–818, 2013.
- 16 Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2022. URL: <https://www.gurobi.com>.
- 17 David G. Harris and N. S. Narayanaswamy. A faster algorithm for vertex cover parameterized by solution size. *CoRR*, abs/2205.08022, 2022. doi:10.48550/arXiv.2205.08022.
- 18 Demian Hesse, Sebastian Lamm, Christian Schulz, and Darren Strash. WeGotYouCovered: The winning solver from the PACE 2019 challenge, vertex cover track. In *Proceedings of the SIAM Workshop on Combinatorial Scientific Computing (CSC)*, pages 1–11, 2020. doi:10.1137/1.9781611976229.1.

- 19 Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.
- 20 Rafael Kiesel and André Schidler. A Dynamic MaxSAT-based Approach to Directed Feedback Vertex Sets. In Gonzalo Navarro and Julian Shun, editors, *Proceedings of the Symposium on Algorithm Engineering and Experiments (ALENEX)*, pages 39–52. SIAM, 2023. doi: 10.1137/1.9781611977561.ch4.
- 21 D.H. Lee and S.M. Reddy. On determining scan flip-flops in partial-scan designs. In *IEEE/ACM International Conference on Computer-Aided Design, (ICCAD)*, pages 322–325, 1990. doi: 10.1109/ICCAD.1990.129914.
- 22 Mile Lemaic. *Markov-Chain-Based Heuristics for the Feedback Vertex Set Problem for Digraphs*. PhD thesis, Universität zu Köln, 2008. URL: <https://kups.ub.uni-koeln.de/2547/1/Dissertation.pdf>.
- 23 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)*, pages 364–369. IEEE, 1999.
- 24 Daniel Lokshantov, NS Narayanaswamy, Venkatesh Raman, MS Ramanujan, and Saket Saurabh. Faster parameterized algorithms using linear programming. *ACM Transactions on Algorithms (TALG)*, 11(2):1–31, 2014.
- 25 Tatiana Orenstein, Zvi Kohavi, and Irith Pomeranz. An optimal algorithm for cycle breaking in directed graphs. *Journal of Electronic Testing*, 7(1):71–81, 1995.
- 26 Panos M Pardalos, Tianbing Qian, and Mauricio GC Resende. A greedy randomized adaptive search procedure for the feedback vertex set problem. *Journal of Combinatorial Optimization*, 2(4):399–412, 1998.
- 27 Rick Plachetta and Alexander van der Grinten. Sat-and-reduce for vertex cover: Accelerating branch-and-reduce by sat solving. In *Proceedings of the Symposium on Algorithm Engineering and Experiments (ALENEX)*, pages 169–180, 2021. doi:10.1137/1.9781611976472.13.
- 28 Igor Razgon. Computing minimum directed feedback vertex set in $O(1.9977^{11})$. In *Italian Conference on Theoretical Computer Science (ICTCS)*, pages 70–81. World Scientific, 2007.
- 29 Andreas Schindler and Rafael Kiesel. Dager. URL: <https://github.com/ASchidler/dfvs>.
- 30 Christian Schulz, Ernestine Großmann, Tobias Heuer, and Darren Strash. Pace 2022: Directed feedback vertex set. In *17th International Symposium on Parameterized and Exact Computation (IPEC 2022)*, 2022.
- 31 Adi Shamir. A linear time algorithm for finding minimum cutsets in reducible graphs. *SIAM Journal on Computing*, 8(4):645–655, 1979.
- 32 G Smith and R Walford. The identification of a minimal feedback vertex set of a directed graph. *IEEE Transactions on Circuits and Systems*, 22(1):9–15, 1975.
- 33 Matthias F Stallmann, Yang Ho, and Timothy D Goodrich. Graph profiling for vertex cover: Targeted reductions in a branch and reduce solver. *arXiv preprint arXiv:2003.06639*, 2020.
- 34 Ching-Chy Wang, Errol L Lloyd, and Mary Lou Soffa. Feedback vertex sets and cyclically reducible graphs. *Journal of the ACM*, 32(2):296–313, 1985.
- 35 Mingyu Xiao and Hiroshi Nagamochi. Exact algorithms for maximum independent set. *Information and Computation*, 255:126–146, 2017. doi:10.1016/j.ic.2017.06.001.