Domination data reduction rules for the directed feedback vertex set problem

Sylwester Swat[0000−0001−8763−0045], Piotr Łukasiak[0000−0003−4012−9135], and Marta Kasprzak[0000−0002−9863−5412]

Poznan University of Technology, Institute of Computing Science
Piotrowo 2, 61-138 Poznań, Poland

Abstract. In the Directed Feedback Vertex Set problem (DFVS), the goal is to find the smallest possible subset of vertices of a given directed graph whose removal leaves the graph acyclic. The problem has important applications in computational science, and it was a subject of the 7^{th} Parameterized Algorithms and Computational Experiments Challenge, an international algorithmic contest held in 2022 aiming to stimulate and enhance developments in the field. The DFVS is known to be NP-complete and constitutes a natural generalization of the Vertex Cover problem (VC). Compared to the VC, however, the DFVS seems to be vastly more difficult. One of the common approaches to tackle computationally difficult problems is to design data reduction rules. The approach is known to be highly effective (both in theory - kernelization algorithms, and practice - reduction in processed data size) for several combinatorial problems, including the VC. For the DFVS substantially fewer reduction rules are known than for the VC, and they are much less robust in practical applications. In this paper we address this issue and present several new data reduction rules for the problem. Presented algorithms are based on an extended concept of domination. We provide detailed description and analysis of designed techniques and, additionally, provide insights regarding practical applicability of discussed methods. At the end, we present a compact juxtaposition showing the impact of implemented data reduction rules for a collection of real-world graph instances.

Keywords: Induced acyclic graph · Data reduction rules · Domination

1 Introduction

One of the NP-complete problems in the field of graph theory is the Directed Feedback Vertex Set problem (DFVS), where the goal is to find the smallest subset of vertices whose removal makes a given graph acyclic. The problem is sometimes equivalently stated as the Maximum Induced Acyclic Subgraph problem, where one tries to find a largest subset of vertices that induces an acyclic graph. The statement can also be formed as a hitting set problem, where the goal is to find a subset of vertices that hit all cycles in a given graph.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29921-5_35

The DFVS has a wide range of applications in real-world problems, in many scientific areas. Finding a large induced acyclic subgraph plays an important role in the deadlock recovery process [15] in operating systems and databases and is used in argumentation frameworks [12]. The DFVS problem has also significant impact in the field of very large-scale integration circuits and chip design [19], and it is one of the fundamental factors in the partial scan method [8], where it is required to choose a small number of flip-flops to decrease the overhead area of a full scan design. In addition to its applications in practical computational science, the DFVS is also important from the theoretical point of view, especially in the context of parameterized algorithms. Many combinatorial problems admit efficient fixed-parameter tractable (FPT) algorithms when parameterized by a carefully chosen parameter. For example, the DFVS can be solved in time $O(4^k k! k^5 \cdot (n+m))$, when the parameter k corresponds to the solution size, but there also exists a $2^{O(t \cdot \log t)} \cdot n^{O(1)}$ algorithm for general directed graphs and a $2^{O(t)} \cdot n^{O(1)}$ algorithm for directed planar graphs, when parameter t corresponds to treewidth [5]. Due to the relevance of the DFVS, both in theory and in practice, it was the subject of the 7^{th} Parameterized Algorithms and Computational Experiments Challenge (PACE 2022), an international algorithmic competition held annually since 2016.

It is known that the DFVS is NP-hard [16,25] and FPT (for parameter "solution size"). The determination of the tractability of the DFVS was proposed as an open problem in some of the first articles on fixed-parameter tractability [11], and it took a surprisingly long time to finally find an FPT algorithm for the problem. In a breakthrough paper [7] an algorithm running in time $O(4^k k! k^4 n^4)$ was given, thus proving that the DFVS is FPT. Throughout several years the solution was further improved (e.g., in [28] authors presented an $O(4^k k! k^5 (n+m))$ algorithm for the problem), but the results are mainly interesting from the theoretical point of view, as the complexity still makes the algorithms impractical. The DFVS is a generalization of the NP-hard Vertex Cover problem (VC). This is because the VC can be easily transformed to the DFVS by replacing each edge in the vertex cover graph instance with two arcs with opposite directions. Despite the apparent similarity between the problems, the practical difficulty of the two problems is vastly different. There exist many distinct simple algorithms for the Vertex Cover problem with running time complexity that can be easily bounded by $O(2^k \cdot (n+m))$. Contrary to the VC, the DFVS does not seem to have a simple FPT algorithm. The first such algorithm found for the problem has running time $O(4^k k! k^4 n^4)$, and no significant improvement has been made since then to notably reduce the parametric dependence. Additionally, the algorithm uses a rather complex approach based on an iterative compression framework [9] and nontrivial observations regarding topological orderings of graphs [7]. After the positive answer for the fixed-parameter tractability of the DFVS was provided, a natural question for the existence of a polynomial kernel for the problem occurred. This question remains one of the largest open problems in the field of parameterized algorithms. Although the existence of a polynomial kernel has not been proven for the parameter "solution size", there exists a kernel for the prob-

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29921-5_35

lem with $O(k^4)$ vertices for general graphs, where k is the size of the feedback vertex set of the underlying undirected structure of the graph [4].

A number of exact algorithms have been designed to solve the problem in practice (see, e.g., [1,3,22,29,30]). The most efficient approaches in this context are based on the hitting-set approach and aim to find a smallest hitting set of a dynamically extended set of cycles in the graph. Distinct heuristics have also been developed to find suboptimal solutions for large graphs and graphs with complex structures. To the most common approaches one can include GRASP heuristics with distinct node selection objectives (see, e.g., [24,32]) or local search approaches, particularly those based on the simulated annealing technique [14,31,32,33,34]. Alongside heuristic and exact algorithms for the DFVS, a number of preprocessing techniques can be used to speed up the process of finding solutions. Several data reduction rules have already been known for years [13,23,24,26,27], several new ones have been developed in recent years, many of which are a "byproduct" of prepared submissions to the PACE 2022 contest [1,2,10,22,29,34]. In spite of the noticeable increase in the number of data reduction rules for the DFVS, the set of those preprocessing techniques is meagre when compared to the family of known kernelization algorithms and reduction rules for the VC. In this work we address this issue and present new concepts, approaches and data reduction rules for the Directed Feedback Vertex Set problem.

2 Notions and definitions

Before we proceed to the description of the algorithms, let us fix the following notions and notations. Let $G = (V, A)$ be a directed graph, where V is a set of vertices and A is a set of arcs. Let $u, v \in V$. We call an arc $(u, v) \in A$ a *pi-arc* if $(v, u) \in A$. Otherwise (u, v) is called a *nonpi-arc*. A *pi-graph* G_{pi} of a graph G is a graph $G[A_{pi}]$ induced by a set A_{pi} of all pi-arcs in A. Similarly, a *nonpi-graph* G_{npi} is a graph G induced by a set of all nonpi-arcs A_{npi}. A node $v \in V$ is called a *pi-node* if all arcs incident to v are pi-arcs. Otherwise node v is called a *nonpi-node*. By $N^+(v)$ we denote the out-neighborhood of node v, by $N^-(v)$ we denote the in-neighborhood of node v, and by $N^{pi}(v)$ we denote the intersection $N^-(v) \cap N^+(v)$. By $d^-(v)$ and $d^+(v)$ we denote the in-degree and out-degree of node v, respectively. By $N_H^-(v)$ and $N_H^+(v)$ we denote the in-neighborhood and out-neighborhood of node v in a given graph H. We denote $N_{npi}^-(v) = N_{G_{npi}}^-(v)$ and $N_{npi}^+(v) = N_{G_{npi}}^+(v)$. We also denote closed neighborhood $N[u] = N(u) \cup \{u\}$, and similarly for N^-, N^+, N^{pi}. A *chordless cycle* (or *induced cycle*) is a cycle $C = (u_0, u_1, \ldots, u_{t-1})$ for which, for each $0 \leq i, j < t$, the condition $(u_i, u_j) \in A$ implies that $j - i \in \{1, 1 - t\}$.

Let us now define the contraction operation performed on the graph that will be used in the paper. By *contraction* of node $v \in V$ we mean the procedure of adding to the graph all arcs from the set $(N^-(v) \times N^+(v)) \setminus A$ and removing from the graph node v with all incident arcs. The contraction operation is sometimes also called as *bypass*, *shortcut* or *merge* operation.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29921-5_35

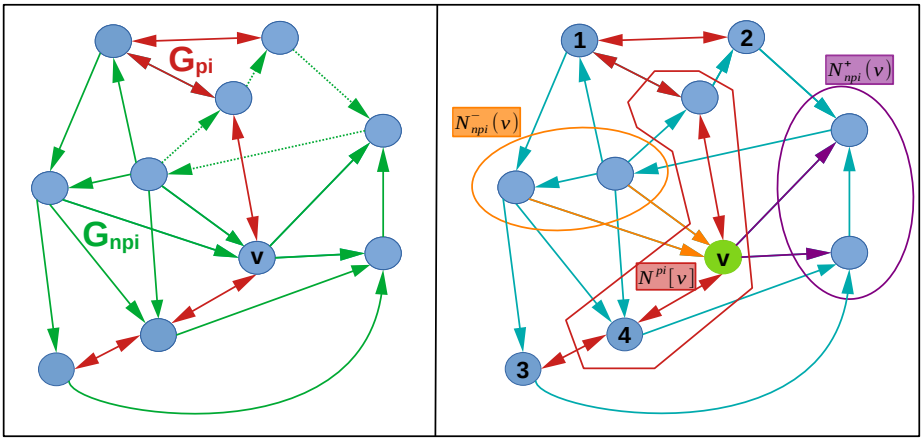

Fig. 1: The figure shows an exemplary graph G. On the left side all pi-arcs are marked with red (they induce a pi-graph G_{pi}, all nodes incident to a red arc belong to G_{pi}), and all nonpi-arcs are marked with green (they induce a nonpi-graph G_{npi}, all nodes incident to a green arc belong to G_{npi}). Additionally, an exemplary chordless cycle is marked with dotted arcs. On the right side a node v and neighborhoods $N^-_{npi}(v)$, $N^+_{npi}(v)$, and $N^{pi}[v]$ are highlighted. Nodes numbered 1-4 are relevant for reduction rules described in the paper.

3 Data reduction rules

The general aim in the process of designing data reduction rules for a combinatorial problem is to reduce the size of processed data or simplify its structure in some specific way. In many graph problems, including the VC and DFVS, this often boils down to identifying a set of nodes X for which it can be proved that there exists an optimum solution S_{opt} containing X. A node $v \in V$ is called *reducible* if there exists such a set X containing v. When a reducible node is detected, it can be added to the created solution and removed from the graph, thus reducing the graph size and size of any optimum solution to the remaining graph. If a reduction rule enables us to identify a reducible node, then we also call the rule *reducible*. Although it is highly profitable to identify reducible nodes in the graph, it might not be possible to do so. It might still be often possible, however, to apply some modifications to the graph structure without immediate detection of reducible nodes. If a reduction rule changes the graph structure during execution and the structure needs to be correspondingly modified later to find the solution, or the solution found for the remaining graph needs to be adequately modified in other way than just adding a node to it, then we call the rule *liftable*. If a rule enables us to make some changes to the graph that might be considered as irrelevant and neither the graph nor the solution need to be modified later, then we call the reduction *pure*. A reduction rule is called *safe* if

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29921-5_35

Table 1: Most commonly used reduction rules for the DFVS

Id	Rule	Type	Description
1	loop	reducible	if there is a loop $(v, v) \in A$, then v is reducible
2	parallel arcs	pure	if there is more than one arc $(u, v) \in A$, remove all but one
3	in-out-1	pure	a node v with $d^-(v) = 1$ or $d^+(v) = 1$ can be bypassed
4	core	reducible	if each neighbor of a pi-node v is a pi-node and an induced graph $G[N[v]]$ is a clique, then nodes $N(v)$ are reducible
5	pie	pure	remove an arc with ends in distinct strongly connected components of G_{npi}
6	dome	pure	remove an arc $(u, v) \in A_{npi}$ for which $N^-_{npi}(u) \subseteq N^-(v)$ or $N^+_{npi}(v) \subseteq N^+(u)$
7	hypergraph domination	pure	if node u belongs to each cycle containing v, then bypass node v
8	flower (node-disjoint cycle)	pure	contract node u for which there do not exist two node-disjoint (except for u) cycles containing u
9	chordless cycle	pure	remove arcs that do not belong to a chordless cycle
10	shortone	pure	let $(u, v), (v, w) \in A_{npi}$; if $\lvert N^+_{npi}(v)\rvert = \lvert N^-_{npi}(v)\rvert = 1$ and $N^{pi}(v) \subseteq N^{pi}(u) \cup N^{pi}(w)$, then remove arcs $(u, v), (v, w)$ and add arc (u, w)
11	in-out-clique	pure	if $N^-(v)$ or $N^+(v)$ induces a clique in G_{pi}, then node v can be contracted
VC1	degree-2 folding	liftable	fold a node with degree 2: replace nodes a, b, c, where $N(b) = \{a, c\}$, with a new node x with $N(x) = N(a) \cup N(c)$.

the optimal solution for the reduced graph can be transformed into an optimal solution of the original graph.

3.1 Known data reduction rules for the DFVS

In this section we consider a set of reduction rules known for the DFVS. This set is by no means complete, it does not contain all known reduction rules but most commonly used reduction rules and rules known to be applicable in practice. In Table 1 those rules are listed together with their short description. More details about the rules can be found in works [1,2,10,13,22,23,24,26,27,29,34]. In the table a single reduction rule known for the VC is additionally listed, the degree-2 folding, it can also be applied to instances of the DFVS whenever local parts of the graph do not contain nonpi-arcs. This one is used to tackle one of commonly occurring structures in many graphs, chains of degree-2 pi-nodes, which can otherwise add a lot of redundancy to a graph structure.

Several more data reduction rules were mentioned in the literature, but most of them are just special cases of a rule listed in the table.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29921-5_35

3.2 Domination data reduction rules

Let us now proceed to the description of new data reduction rules. First let us mention the following easy observation:

Lemma 1 (hitting chordless cycles). *A set $S \subset V$ is a feedback vertex set of a directed graph G if and only if it hits the set of all chordless cycles in G.*

Lemma 1 is a basis for many reduction rules (e.g., the dome rule (6) from Table 1 and rules 4 and 5 from this section). In what follows we present a series of domination-based data reduction rules together with their analysis.

Reduction 1 (domination-1). If there exists a pi-arc (u, v) with $N^+(v) \subseteq N^+[u]$ and $N^-(v) \subseteq N^-[u]$, then node u is reducible and can be removed from the graph and added to the solution.

Theorem 1. *Reduction rule 1 is safe. Checking all pi-arcs in the graph for fulfilling requirements needed to apply the rule can be done in total time $O(|A|^{\frac{3}{2}})$.*

Proof. First we prove that the rule is safe. Suppose now that the rule is not safe and let S be an optimal solution to G. Since the rule is not safe, there does not exist an optimal solution that contains u. Because (u, v) is a pi-arc, we conclude that $v \in S$. Consider now any cycle in G, except for (u, v), that contains v and denote the cycle $C = (v_0, v_1, \ldots, v_t)$. Due to the conditions $N^+(v) \subseteq N^+[u]$ and $N^-(v) \subseteq N^-[u]$, we observe that $C' = (u, v_1, \ldots, v_t)$ is also a cycle in G. Since S is a feedback vertex set in G, then cycle C' is hit by S, from what follows that C is hit by $S \setminus \{v\}$. But now, $S' = S \setminus \{v\} \cup \{u\}$ also hits all cycles in G, which means that it is a valid solution. A contradiction, hence the rule is safe.

Now let us show that the rule can be checked for all pi-arcs in A in total time $O(|A|^{\frac{3}{2}})$. First we observe that the conditions $N^+(v) \subseteq N^+[u]$ and $N^-(v) \subseteq N^-[u]$ imply that $|N(v)| \leq |N(u)|$. We now consider all nodes in order of nonincreasing sum of degrees $d^-(v) + d^+(v)$. For a considered node u we mark the sets $N^-(v)$ and $N^+(v)$ using auxiliary bitarrays. Then we enumerate over all pi-arcs (u, v) incident to the node and over sets $N^-(v)$ and $N^+(v)$ to check for inclusion conditions required by the rule. The algorithm works in a very similar way to the algorithm for listing all triangles in a graph [20]. By partitioning nodes from V into two sets $X = \{v : d(v) \geq K\}$ and $Y = \{v : d(v) < K\}$ and noticing that $|X| \leq \frac{|A|}{K}$ and $d(v) \leq d(u)$ whenever a pi-arc (u, v) is considered, we can bound the number of operations performed by the algorithm by $|X| \cdot |A| + \sum_{v \in Y} d^2(v) \leq \frac{|A|^2}{K} + \frac{|A|}{K} \cdot K^2$. By setting $K = |A|^{\frac{1}{2}}$ we obtain the desired bound on the running time complexity.

In the necessary conditions to apply rule 1, one needs to have both in- and out-neighborhood of node v "dominated" by respective neighborhoods of node u. In the following reduction rule the inclusion condition for only one of the neighborhoods need to hold, but it needs to be a subset of $N^{pi}[u]$ instead. An exemplary domination property required to apply rule 1 is shown on the right side of Fig. 1. By applying the rule for pi-arc $(4, 3)$, we mark node 4 as reducible.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29921-5_35

Reduction 2 (domination-2). If there exists a pi-arc (u, v) with $N^-(v) \subseteq N^{pi}[u]$ or $N^+(v) \subseteq N^{pi}[u]$, then node u is reducible and can be removed from the graph and added to the solution.

Theorem 2. *Reduction rule 2 is safe. Checking all pi-arcs in the graph for fulfilling requirements needed to apply the rule can be done in total time* $O(|A|^{\frac{3}{2}})$.

Proof. Suppose that the rule is not safe. Let S be an optimal solution for graph G. We know that $u \notin S$ (since the rule is not safe), hence $N^{pi}(u) \in S$. Since either $N^-(v) \subseteq N^{pi}[u]$ or $N^+(v) \subseteq N^{pi}[u]$, each cycle containing node v must also contain a node from $N^{pi}[u] \setminus \{v\}$. From the previous observation it follows that each cycle in G is hit by a set $S^* = S \setminus \{v\} \cup \{u\}$. But $|S^*| = |S|$, hence it is an optimal feedback vertex set of G. A contradiction, hence the rule is safe. Proof of the running time complexity is analogous to the proof of Theorem 1.

An exemplary domination property required to apply rule 2 is shown on the right side of Fig. 1. By applying the rule for pi-arc $(1, 2)$ we can mark node 1 as reducible. Like most reduction rules for the DFVS and VC, both rules 1 and 2 make use only of the local properties - neighborhoods of ends of a pi-arc. The following rule relaxes this 'requirement' and is a generalization of the previous rule. This, however, comes at a cost of deteriorated time complexity.

Reduction 3 (domination-3). If there exists a pi-arc (u, v) such that in the graph $G[V \setminus N^{pi}[u]]$ there does not exist a path from $N^+(v)$ to $N^-(v)$, then node u is reducible and can be removed from the graph and added to the solution.

Theorem 3. *Reduction rule 3 is safe. Checking all pi-arcs in the graph for fulfilling requirements needed to apply the rule can be done in time* $O(|A_{pi}||A|)$.

Proof. Assume that the rule is not safe. Let $G' = G[V \setminus N^{pi}[u]]$, and let S be an optimal solution for G. Since the rule is not safe, $u \notin S$. We know that in G' there is no path from $N^+(v)$ to $N^-(v)$, hence there is no cycle containing v in G'. Therefore, there is also no cycle containing v in $G[V \setminus (S \setminus \{v\}]$, except for a pi-arc (u, v). But from this we can see that the set $S^* = S \setminus \{v\} \cup \{u\}$ is a feedback vertex set of G and $|S^*| = |S|$. A contradiction, hence the rule is safe.

The complexity of the algorithm is bounded by the number of times we need to check for an existence of a path, which is bounded by $|A_{pi}|$. Finding the path (or deciding that it does not exist) take $O(|A|)$ time using a standard DFS/BFS search. The final complexity is therefore bounded by $O(|A_{pi}||A|)$.

It is worth to mention now that the algorithm can be implemented to be much faster in practice for most graph instances, due to the fact that the number of reducible nodes is usually much smaller than the number of times we search for a path. This means that almost always a search for an existence of a path concludes in a positive answer. By replacing the standard DFS/BFS search with a bidirectional BFS, the algorithm performance in practice can be significantly improved, as the whole graph will need to be looked through only when a path does not exist [6]. For some graph classes it can be proved that the algorithm is

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29921-5_35

asymptotically faster. For random graphs, for example, the algorithm works in time $O(|A_{pi}| \cdot |A|^{\frac{1}{2}} + |P| \cdot |A|)$, where P is the set of found reducible nodes.

The fourth presented reduction rule is a slight variation of the third, where, by Lemma 1, instead of looking for a path we look for an induced path.

Reduction 4 (domination-4). If there exists a pi-arc (u, v), such that in the graph $G[V \setminus (N^{pi}[u] \setminus \{v\})]$ there does not exist a chordless cycle containing node v, then node u is reducible.

Theorem 4. *Reduction rule 4 is safe.*

Proof. The proof of the safeness of the rule is the same as for rule 3, but taking into account that only chordless cycles need to be hit.

Let us note that the running time of rule 4 depends highly on the algorithm used for the chordless cycle detection. It is in general NP-hard to check whether a given node belongs to a chordless cycle [18]. This does not mean, however, that the problem cannot be solved efficiently in practice. There exist enumeration algorithms that can list all chordless cycles in total time $O((n + m)(c + 1))$, where n, m, c denote the number of nodes, arcs and elementary cycles in the graph, respectively [21]. The number c of cycles can be, however, exponentially dependent on the graph size. But even in that case there may be many vertices in the graph for which the number of induced cycles containing them is acceptable, and for those nodes rule 4 can be applied (for nodes that are contained in a high number of cycles, the enumeration algorithm must be terminated after some fixed number of cycles to avoid getting stuck in a prolonged computation process). In practice such case occurs most often for graphs with a high fraction of pi-arcs. Let us now also add that rule 4 is very useful in the context of lossy data reduction rules. By relaxing the condition of hitting all chordless cycles to hitting only chordless cycles of some small fixed length L (e.g., 12), the applicability of the rule grows significantly whenever we do not need to preserve optimality of the result, as finding short chordless cycles can be done efficiently.

Let us now present fifth of the domination-based reduction rules. It is a generalization of rules 1 - 4, but its running time complexity is correspondingly worse.

Reduction 5 (domination-5). Let $u \in V$ be a node and let $X \subseteq V \setminus \{u\}$ be a set of nodes such that $G[X \cup \{u\}]$ is a chordless cycle and for each $v \in X$ at least one of the following two conditions hold:

1. at least one of the reductions 1, 2, 3, 4 is applicable for pi-arc (u, v) in the graph $G' = (V \setminus (X \setminus \{v\}), A \cup \{(u, v), (v, u)\})$.
2. for every chordless cycle $C : v \to a_1 \to a_2 \to \ldots \to a_k \to v$ for which $u \notin C$, we have $C \cap N^-(u) \neq \emptyset$ and for every $a_j \in C \cap (N^-(u)) \setminus X$ there exists $a_i \in C \cap (N^+(u)) \setminus X$ with $i \leq j$.

Node u is reducible.

Theorem 5. *Reduction rule 5 is safe.*

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29921-5_35

Proof. Let $u \in V$ be a fixed node and consider any node $v \in X$. Let S be an optimal solution to G and assume that the rule is not safe. Hence, $u \notin S$.

Suppose now that the first condition of the rule is met. By assumption, at least one of the reduction rules 1, 2, 3, 4 holds for the pi-arc (u, v) in the graph $G' = (V, A \cup (u, v) \cup (v, u))$. From proofs of those rules it follows that $S \setminus \{v\}$ hits all cycles containing v, except for (u, v) in G'. Hence, it hits all cycles that contain node v in $G[V \setminus \{u\}]$. Since S is a feedback vertex set of G, all cycles containing node u are hit by S. From the condition that $G[X \cup \{u\}]$ contains a cycle containing u, it follows that $X \cap S \neq \emptyset$. But we have proved that for any node $v \in X$ all the cycles containing v in $G[V \setminus \{u\}]$ are hit by $S \setminus \{v\}$. Taking any node $v' \in X$ and considering $S^* = S \setminus \{v'\} \cup \{u\}$, we observe that all the cycles in G are hit by S^*. This is a contradiction to the assumption that the rule is not safe.

Now assume that the second condition of the rule is met and let C be a chordless cycle for which $u \notin C$ and $v \in C$. Let $a_j \in C \cap (N^-(u) \setminus X)$. From the condition it follows that such a node exists and there exists also $a_i \in C \cap (N^+(u) \setminus X)$ with $i \leq j$. We know that S hits all the cycles in G, thus it hits all the cycles containing u. Since every cycle containing u has the form $u \to a \to \ldots b \to u$, where $a \in N^+(u)$ and $b \in N^-(u)$, and $u \notin S$, we have $\{a_i, a_{i+1}, \ldots, a_j\} \cap S \neq \emptyset$. Therefore, the cycle C is hit by $S \setminus \{v\}$. Thus, S hits all cycles that contain v, except for cycle $G[X \cup \{u\}]$. But this means that $S \setminus \{v\} \cup \{u\}$ is a feedback vertex set of G, which ends the proof.

The complexity of determining whether the first condition of rule 5 is met is in theory dominated by the complexity of rule 4. Without additional restrictions, for a given vertex u all other nodes $v \in V$ must be considered and the existence of a cycle containing v must be checked, resulting in a complexity of $O(|V|^2|A|)$. As mentioned earlier, the total number of operations performed by the algorithm in practice can be reduced by using bidirectional BFS instead of a standard cycle detection algorithm. Determining whether the necessary requirements for the second condition of rule 5 are met can be done in $O(|A|)$ time by checking whether there is a path from X to $N^-_{npi}(u) \setminus X$ in the graph $G[V \setminus \{u\}]$. Similarly to replacing standard cycle-search with a bidirectional BFS, we can make adjustments to make the search faster in practice. Finally, let us note that the reduction rule 5 does not depend on the existence of pi-arcs in the graph, and might be applicable even if $A_{pi} = \emptyset$. An exemplary subgraph for the rule 5 can be applied is shown in Fig. 2.

4 Experiment

In this section we provide results of a computational experiment conducted to show applicability of designed rules. The experiment was conducted on a machine with Windows operating system in the WSL2 subsystem, with a Ryzen 7 5700X3D processor and 64GB of DDR4 3200 MT/s RAM memory.

To evaluate the effect of the domination-based data reduction rules, we used all 400 graph instances from the PACE 2022 contest [17]. This set contains,

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29921-5_35

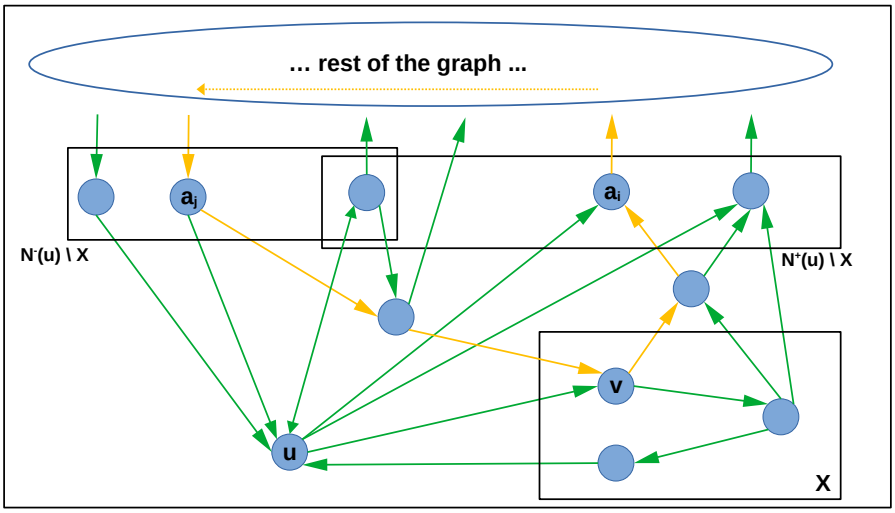

Fig. 2: The figure shows an exemplary graph for which the reduction rule 5 can be applied. A fragment of an exemplary cycle C from the second condition of the rule 5 is highlighted in orange.

among others, real-world instances representing web graphs, social networks, Wikipedia graphs, and autonomous system graphs. Many random graph instances were used as well, generated according to various models (e.g., Erdős-Rényi graphs, geometric graphs, hyperbolic graphs, and Delaunay graphs).

We implemented three preprocessing algorithms using three different sets of data reduction rules. The first set contained only basic data reduction rules: loop, parallel arc removal, and in-out-1 node contraction. A graph obtained after applying these rules constitutes the core structure. The second set also contained several of the known rules: the core, dome, pie, in-out-clique rule, and the degree-2 folding (see Table 1). This set of rules was used to obtain graphs that can be used to examine the influence of the new domination rules on already preprocessed instances. Since these graphs are already subjected to fair preprocessing, they should be much less susceptible to reductions than bare structures. The third set also contained all the domination rules designed and described in this work. Additionally, in order to measure the impact of particular reduction rules, we also considered as sets of reduction rules the second set extended by a single reduction, for which we wanted to measure the impact.

For each of these three sets of reduction rules, we calculated the following values: the number of instances fully solved by the set (that is, instances for which the optimal feedback vertex set was found using the data reduction rules only) and metrics corresponding to the reduction ratios, calculated as the fractions $\frac{|V_G|-|V_{G'}|}{|V_G|}$ - for the node reduction ratio - and $\frac{|A_G|-|A_{G'}|}{|A_G|}$ - for the arc reduction ratio. Here, by G' we denote the graph obtained after applying data reduction

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29921-5_35

Table 2: Comparison of results obtained by application of reduction rules in given sets. Set 1 contains only basic rules, set 2 contains also other known rules, and set 3 contains additionally the proposed domination rules 1–5.

	Set 1 (b)	Set 2 (b+k)	Set 3 (b+k+d)
Number of fully solved instances	2	27	63
Average reduction ratio	0.275 (nodes) 0.254 (arcs)	0.515 (nodes) 0.529 (arcs)	0.601 (nodes) 0.621 (arcs)
Average reduction time	0.245 sec	1.378 sec	9.89 sec

rules to graph G. A summary of the results obtained is shown in Table 2. It can be seen that, on average, the node reduction ratios obtained by the second and third sets are better by 87.2% and 118.5%, respectively, compared to those obtained by the first set. Comparing the effect of using the domination rules described in this work in addition to the known rules (the second set), we were able to obtain the average node reduction ratio greater by 16.7% (0.601 vs. 0.515). The values of the reduction ratios obtained for the arcs are comparable to those obtained for the nodes. However, if we look at the number of fully solved instances, we see a significant increase when comparing the results of the third set vs. the second set, more than twice as much.

Let us now examine the impact of particular domination rules, shown in Figs. 3 and 4. The highest impact, in the sense of graph sizes after the reduction, we can observe for rule 5. It gives the average number of nodes in a graph roughly 14621 and the node reduction ratio 0.601. Two other rules give similar values, rule 2 (14711 and 0.598, respectively) and rule 1 (14876 and 0.594). A larger gap can be observed for the remaining rules 3 and 4, which resulted in graphs of almost equivalent size (on average). Comparing the time efficiency of sets of reduction rules extended by rule 1, 2, 3, 4, or 5, they required on average,

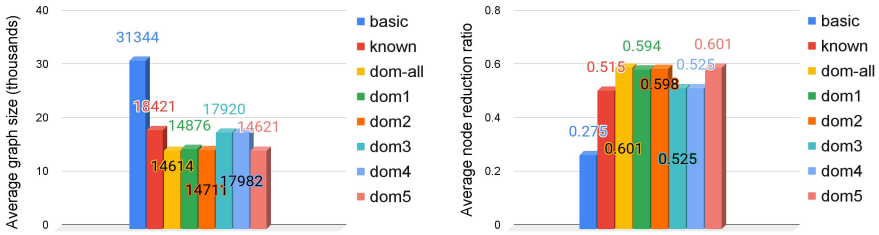

Fig. 3: The figure shows the impact of particular domination rules on graph sizes after the reduction. "domX" denotes the set of known rules (set 2) extended by a single domination rule X. On the left we see the average number of nodes in the graphs after the reduction, on the right there are values of the node reduction ratio.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29921-5_35

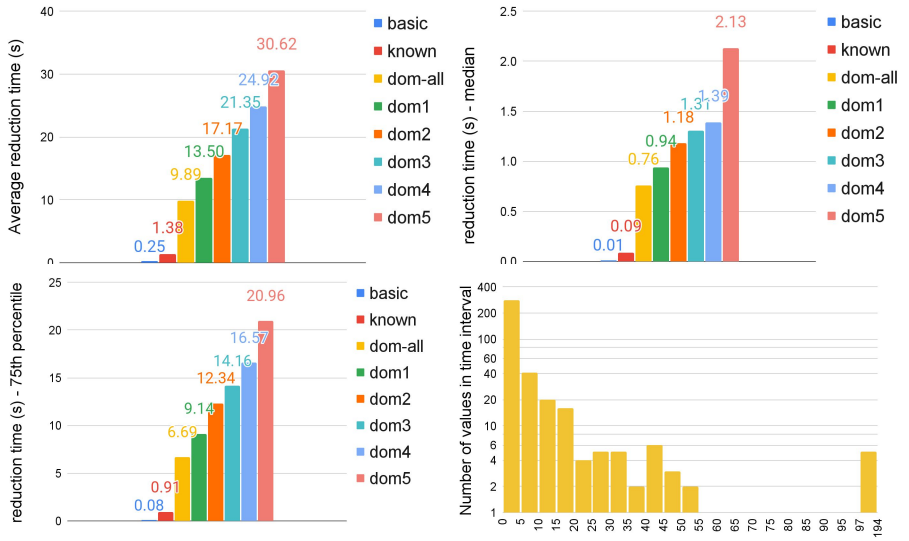

Fig. 4: The figure shows the impact of particular domination rules on the preprocessing in the sense of time. "domX" denotes the set of known rules (set 2) extended by a single domination rule X. The top two charts and the bottom left one show the average reduction time, the median and the 75th percentile, while the bottom right chart shows the distribution of the values, packed in 5-second intervals, with the y-axis in the logarithmic scale. In order to improve readability, the histogram is created for the set of values with removed 2% of outliers.

respectively, 13.5, 17.2, 21.4, 24.9, or 30.6 seconds to process the graph instances. This is a noticeable but still reasonable overhead, compared to the set of "known" rules, which required only 1.4 seconds. Surprisingly, the average time required to process the graphs using the third set of rules was shorter (9.9 seconds) than that for the second set extended by any single domination rule. This shows that using a rule (or even several ones) that is relatively slow may improve the overall performance. This phenomenon is most probably caused by the fact that using approaches that might affect different subgraphs we can reduce the total number of passes required by the algorithm to determine that no rule can be further applied.

5 Conclusions

In the paper we have presented new data reduction rules for the Directed Feedback Vertex Set problem, a problem relevant in computational science with applications in several scientific areas. These rules are based on a concept of domination, similar to the one known for the Vertex Cover problem. However, for the DFVS domination need not be a property as local as for the VC, and the designed rules exploit this observation. We have provided detailed descrip-

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29921-5_35

tions of designed rules, together with their proofs of soundness and comments on variations, whenever a practical need for an algorithm with better worst-case time complexity arose. Finally, based on a computational experiment including a variety of graph classes and real-world instances, we have provided an analysis and evaluation of the designed reduction rules, showing their practicality.

Funding

This research received funding from the National Science Centre, Poland (grant No. 2023/49/N/ST6/02506, PI: Sylwester Swat).

References

1. Angrick, S., Bals, B., Casel, K., Cohen, S., Friedrich, T., Hastrich, N., Hradilak, T., Issac, D., Kißig, O., Schmidt, J., Wendt, L.: Solving directed feedback vertex set by iterative reduction to vertex cover. In: 21st International Symposium on Experimental Algorithms (SEA). pp. 10:1–10:14 (2023). https://doi.org/10.4230/LIPIcs.SEA.2023.10
2. Behr, T., Storandt, S.: Lossy reduction rules for the directed feedback vertex set problem. In: 2023 Symposium on Algorithm Engineering and Experiments (ALENEX). pp. 53–64 (2023). https://doi.org/10.1137/1.9781611977561.ch5
3. Bergenthal, M., Dirks, J., Freese, T., Gahde, J., Gerhard, E., Grobler, M., Siebertz, S.: PACE Solver description: GraPA-JAVA. In: 17th International Symposium on Parameterized and Exact Computation (IPEC). pp. 30:1–30:4 (2022). https://doi.org/10.4230/LIPIcs.IPEC.2022.30
4. Bergougnoux, B., Eiben, E., Ganian, R., Ordyniak, S., Ramanujan, M.: Towards a polynomial kernel for directed feedback vertex set. In: 42nd International Symposium on Mathematical Foundations of Computer Science (MFCS). pp. 36:1–36:15 (2017). https://doi.org/10.4230/LIPIcs.MFCS.2017.36
5. Bonamy, M., Kowalik, Ł., Nederlof, J., Pilipczuk, M., Socała, A., Wrochna, M.: On directed feedback vertex set parameterized by treewidth. In: 2018 Graph-Theoretic Concepts in Computer Science (WG). pp. 65–78 (2018). https://doi.org/10.1007/978-3-030-00256-5_6
6. Cerf, S., Dayan, B., Ambroggio, U.D., Kaufmann, M., Lengler, J., Schaller, U.: Balanced bidirectional breadth-first search on scale-free networks (2024). https://doi.org/10.48550/arXiv.2410.22186
7. Chen, J., Liu, Y., Lu, S., O'Sullivan, B., Razgon, I.: A fixed-parameter algorithm for the directed feedback vertex set problem. J. ACM **55**, 21:1–21:19 (2008). https://doi.org/10.1145/1411509.1411511
8. Cheng, K.T., Agrawal, V.: A partial scan method for sequential circuits with feedback. IEEE Trans. Comput. **39**, 544–548 (1990). https://doi.org/10.1109/12.54847
9. Cygan, M., Fomin, F., Kowalik, Ł., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., Saurabh, S.: Bounded Search Trees. In: Parameterized Algorithms, pp. 51–76. Springer International Publishing, Cham (2015). https://doi.org/10.1007/978-3-319-21275-3_3

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29921-5_35

10. Dirks, J., Gerhard, E., Grobler, M., Mouawad, A., Siebertz, S.: Data reduction for directed feedback vertex set on graphs without long induced cycles. In: SOFSEM 2024: Theory and Practice of Computer Science. pp. 183–197 (2024). https://doi.org/10.1007/978-3-031-52113-3_13
11. Downey, R., Fellows, M.: Fixed-parameter intractability. In: 7th Annual Conference on Structure in Complexity Theory. pp. 36–49 (1992). https://doi.org/10.1109/SCT.1992.215379
12. Dvořák, W., Hecher, M., König, M., Schidler, A., Szeider, S., Woltran, S.: Tractable abstract argumentation via backdoor-treewidth. In: 36th AAAI Conference on Artificial Intelligence. pp. 5608–5615 (2022). https://doi.org/10.1609/aaai.v36i5.20501
13. Fleischer, R., Wu, X., Yuan, L.: Experimental study of FPT algorithms for the directed feedback vertex set problem. In: 17th Annual European Symposium on Algorithms (ESA). pp. 611–622 (2009). https://doi.org/10.1007/978-3-642-04128-0_55
14. Galinier, P., Lemamou, E., Bouzidi, M.: Applying local search to the feedback vertex set problem. J. Heuristics **19**, 797–818 (2013). https://doi.org/10.1007/s10732-013-9224-z
15. Gardarin, G., Spaccapietra, S.: Integrity of data bases: A general lockout algorithm with deadlock avoidance. In: IFIP Working Conference on Modelling in Data Base Management Systems. pp. 395–412 (1976)
16. Garey, M., Johnson, D.: Computers and Intractability. A Guide to the Theory of NP-Completeness. W.H. Freeman and Company, San Francisco (1979)
17. Großmann, E., Heuer, T., Schulz, C., Strash, D.: The PACE 2022 Parameterized Algorithms and Computational Experiments Challenge: Directed Feedback Vertex Set. In: 17th International Symposium on Parameterized and Exact Computation (IPEC). pp. 26:1–26:18 (2022). https://doi.org/10.4230/LIPIcs.IPEC.2022.26
18. Haas, R., Hoffmann, M.: Chordless paths through three vertices. Theor. Comput. Sci. **351**, 360–371 (2006). https://doi.org/10.1016/j.tcs.2005.10.021
19. Hudli, A., Hudli, R.: Finding small feedback vertex sets for VLSI circuits. Microprocess. Microsyst. **18**, 393–400 (1994). https://doi.org/10.1016/0141-9331(94)90067-1
20. Itai, A., Rodeh, M.: Finding a minimum circuit in a graph. SIAM J. Comput. **7**, 413–423 (1978). https://doi.org/10.1145/800105.803390
21. Johnson, D.: Finding all the elementary circuits of a directed graph. SIAM J. Comput. **4**, 77–84 (1975). https://doi.org/10.1137/0204007
22. Kiesel, R., Schidler, A.: A dynamic MaxSAT-based approach to directed feedback vertex sets. In: 2023 Symposium on Algorithm Engineering and Experiments (ALENEX). pp. 39–52 (2023). https://doi.org/10.1137/1.9781611977561.ch4
23. Köhler, H.: A contraction algorithm for finding minimal feedback sets. In: 28th Australasian Conference on Computer Science (ACSC). pp. 165–173 (2005), https://dl.acm.org/doi/10.5555/1082161.1082180
24. Lemaic, M.: Markov-Chain-Based Heuristics for the Feedback Vertex Set Problem for Digraphs. PhD thesis, Universität zu Köln (2008), https://kups.ub.uni-koeln.de/2547/
25. Lempel, A., Cederbaum, I.: Minimum feedback arc and vertex sets of a directed graph. IEEE Trans. Circuit Theory **13**, 399–403 (1966). https://doi.org/10.1109/TCT.1966.1082620
26. Levy, H., Low, D.: A contraction algorithm for finding small cycle cutsets. J. Algorithms **9**, 470–493 (1988). https://doi.org/https://doi.org/10.1016/0196-6774(88)90013-2

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29921-5_35

27. Lin, H., Jou, J.: On computing the minimum feedback vertex set of a directed graph by contraction operations. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **19**, 295–307 (2000). https://doi.org/10.1109/43.833199
28. Lokshtanov, D., Ramanujan, M., Saurabh, S.: When recursion is better than iteration: A linear-time algorithm for acyclicity with few error vertices. In: 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA). pp. 1916–1933 (2018). https://doi.org/10.1137/1.9781611975031.125
29. Meiburg, A.: Reduction rules and ILP are all you need: Minimal directed feedback vertex set (2022). https://doi.org/10.48550/arXiv.2208.01119
30. Razgon, I.: Computing minimum directed feedback vertex set in O(1.9977n). In: 10th Italian Conference on Theoretical Computer Science (ICTCS). pp. 70–81 (2007). https://doi.org/10.1142/9789812770998_0010
31. Russo, L., Castro, D., Ilic, A., Romano, P., Correia, A.: Stochastic simulated annealing for directed feedback vertex set. Appl. Soft Comput. **129**, 109607 (2022). https://doi.org/10.1016/j.asoc.2022.109607
32. Swat, S.: PACE Solver description: DiVerSeS - a heuristic solver for the directed feedback vertex set problem. In: 17th International Symposium on Parameterized and Exact Computation (IPEC). pp. 27:1–27:3 (2022). https://doi.org/10.4230/LIPIcs.IPEC.2022.27
33. Tang, Z., Feng, Q., Zhong, P.: Nonuniform neighborhood sampling based simulated annealing for the directed feedback vertex set problem. IEEE Access **5**, 12353–12363 (2017). https://doi.org/10.1109/ACCESS.2017.2724065
34. Zhang, Q., Du, Y., Su, Z., Li, C.M., Xu, J., Chen, Z., Lü, Z.: Threshold-based responsive simulated annealing for directed feedback vertex set problem. In: 38th AAAI Conference on Artificial Intelligence. pp. 20856–20864 (2024). https://doi.org/10.1609/aaai.v38i18.30075

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29921-5_35