A new concept of partial domination and algorithms for the directed feedback vertex set problem

Sylwester Swat[0000−0001−8763−0045] 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) one seeks a minimum-size set of vertices whose removal makes a given directed graph acyclic. The problem is known to be NP-complete. It is therefore natural to address it using heuristic methods and preprocessing techniques. In contrast to many other combinatorial optimization problems, however, a limited number of data reduction rules are known for the DFVS. Here we introduce a new concept of partial domination and develop data reduction rules derived from this notion. Moreover, we present a linear-time algorithm for identifying cycle-dominators, a problem of considerable importance in various areas of computational science, most notably in control-flow graph analysis. We provide a thorough analysis of the proposed algorithms, along with results from computational experiments that demonstrate their practical effectiveness.

Keywords: Induced acyclic graph · Data reduction rules · Domination

1 Introduction

One of the classical NP-complete problems in graph theory is the Directed Feedback Vertex Set problem (DFVS). Given a directed graph, the goal is to determine a minimum-size set of vertices whose removal renders the graph acyclic. The problem arises in numerous practical applications. It plays an important role, e.g., in argumentation frameworks [6] and is relevant in the design of very-large-scale integration (VLSI) circuits [11]. Due to its theoretical and practical relevance, the DFVS was selected as the focus problem of the 7^{th} Parameterized Algorithms and Computational Experiments Challenge (PACE 2022), an international algorithm engineering competition organized annually since 2016.

It is well known that the DFVS is NP-hard [7] and fixed-parameter tractable (FPT) with respect to the solution size. Determining its parameterized tractability was posed as an open question in early foundational works on fixed-parameter algorithms [5], and it took considerable time before an FPT algorithm was finally discovered. In a breakthrough result [3], an algorithm with running time $O(4^k k! k^4 n^4)$ was presented, establishing the FPT status of the DFVS. Although subsequent improvements refined the polynomial factors (e.g., [15] gives an

$O(4^k k! k^5 (n+m))$ algorithm) and an improvement of the algorithm, with a running time $O(k! k^5 2^{o(k)}(n+m))$, has been presented [18], the dependence on k remains prohibitive for practical purposes. Despite the close relationship to the Vertex Cover problem, the two problems differ significantly in practical difficulty.

Data reduction techniques play a crucial role in accelerating solution methods. Several reduction rules have been known for years [13,14], and many new rules have been introduced recently, often as byproducts of submissions to the PACE 2022 challenge [1,2,4,12,16]. Nevertheless, compared to the extensive body of kernelization results and reduction techniques for the Vertex Cover, the repertoire of preprocessing methods for the DFVS remains relatively limited.

In this work we propose data reduction rules for the DFVS based on a new concept of partial domination. We also present a surprisingly simple linear-time algorithm for finding cycle-dominators in directed graphs and incorporate it to improve performance of designed rules. Although several algorithms for finding dominators in directed graphs exist (which can be used to find cycle-dominators) [8,10,17], the only known linear-time algorithm for the problem is highly complex. In the end, we provide results and analysis of a conducted computational experiment and evaluate the impact of designed algorithms on preprocessing.

2 Notions and definitions

We begin by introducing terminology and notation used throughout the paper. Let $G = (V, A)$ be a directed graph. An arc $(u, v) \in A$ is called a *pi-arc* if $(v, u) \in A$, otherwise (u, v) is referred to as a *nonpi-arc*. Let $A_{pi} \subseteq A$ be the set of all pi-arcs. The *pi-graph* G_{pi} is the subgraph of G induced by A_{pi}, that is, $G_{pi} = G[A_{pi}]$. Analogously, let A_{npi} denote the set of all nonpi-arcs, and define the *nonpi-graph* $G_{npi} = G[A_{npi}]$. A vertex $v \in V$ is called a *pi-node* if every arc incident to v is a pi-arc. Otherwise, v is called a *nonpi-node*. For a vertex $v \in V$, we denote by $N^+(v)$ its out-neighborhood and by $N^-(v)$ its in-neighborhood. We define $N^{pi}(v) = N^-(v) \cap N^+(v)$. Furthermore, we write $N^-_{npi}(v) = N^-_{G_{npi}}(v)$ and $N^+_{npi}(v) = N^+_{G_{npi}}(v)$ for the in- and out-neighborhoods of v in the nonpi-graph G_{npi}. The closed neighborhood of v is denoted by $N[v] = N(v) \cup \{v\}$, and analogously for N^-, N^+, and N^{pi}. A *chordless cycle* is a cycle C such that the induced graph $G[C]$ is a cycle. The *contraction* of a vertex $v \in V$ consists of adding all missing arcs from $(N^-(v) \times N^+(v)) \setminus A$ to the graph, and subsequently removing the vertex v together with all arcs incident to it. For a given pair of distinct nodes $s, t \in V$ and node $u \in V$, we say that node u dominates node s with respect to node t if every path from s to t contains node u. We also say that a vertex u *cycle-dominates* a vertex v if u belongs to every directed cycle that contains v.

3 Partial domination concept and new reduction rules

Before we proceed to the description of new data reduction rules, let us mention the following easy observation, which is a basis for many data reduction rules:

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

Let us now describe the new concept of domination, we called *partial domination*. By *partial contraction* of node $v \in V$ we mean the procedure of adding to the graph all arcs from the set $(N^-_{npi}(v) \times N^+_{npi}(v)) \setminus A$ and removing from the graph all nonpi-arcs incident to v. By partially contracting a node, it is effectively removed from any oriented cycle of length greater than two, and this significantly reduces the graph's structure and increases chances of applying other rules.

Reduction 1 (partial-domination-1). Let $v \in V$ and let $U \subseteq V$ be such that $G_{npi}[U]$ is a path or a cycle that contains v. Denote the path or cycle by $P = (u_0, u_1, \ldots, u_p)$ and let k be such an index that $u_k = v$. If $|N^-_{npi}(u_i)| = 1$ for all $0 < i \leq k$, $|N^+_{npi}(u_i)| = 1$ for all $k \leq i < p$, and $N^{pi}(v) \subseteq \bigcup_{u \in U} N^{pi}(u)$, then node v can be partially contracted.

Theorem 1. *Reduction rule 1 is safe. Existence of a partially contractable node $v \in V$ can be checked in total time $O(|A|)$.*

Proof. Let S be an optimal solution for G and let S' be an optimal solution for G after partially contracting v (G'). From a previous remark if follows that a solution for G' is also a solution for G, hence we have $|S'| \geq |S|$. We now prove that $|S'| \leq |S|$. If S is a solution for G', then the condition holds, and we are done. Assume then that S is not a feedback vertex set of G'. This can only be the case if $v \in S$, as otherwise all cycles in G' are hit by S. If $N^{pi}(v) \subseteq S$, then the set $S^* = S \setminus \{v\} \cup \{u_0\}$ is a solution for G' and we have $|S'| \leq |S^*| = |S|$, and we are done. Suppose now that there exists $w \in N^{pi}(v)$ such that $w \notin S$. From condition $N^{pi}(v) \subseteq \bigcup_{u \in U} N^{pi}(u)$ it follows that there exists $u \in U$ such that $u \in S$. Hence S is also a feedback vertex set of G', because the in-degree and out-degree conditions imply that all chordless cycles of length larger than three in G that contain v are hit by u. A contradiction to the assumption that S is not a feedback vertex set of G'. This completes the proof of safeness.

To determine the existence of a partially contractable node in linear time, it is enough to see that all nodes v and corresponding best candidate paths (or cycles) P can be considered by starting a DFS traversal only from nodes u_1 for which $N^-(u_1) = \{u_0\}$ and $deg^+(u_0) > 1$ (or any node in P if it is an isolated cycle in G_{npi}) and counting, by dynamically updating array entries, for each node $w \in V$ the number $|N^{pi}(w) \cap P|$. Traversing over P a second time, we check for each $v \in P \setminus \{u_0, u_p\}$ whether $|N^{pi}(w) \cap P| > 1$ for each $w \in N^{pi}(v)$, which is equivalent to the condition $N^{pi}(v) \subseteq \bigcup_{u \in U} N^{pi}(u)$.

The first partial domination rule is particularly useful for graphs for which G_{npi} contains long, trivially induced paths, as it significantly simplifies the graph's structure by bringing it somewhat "closer" to the vertex cover problem. Even if the structure of G_{npi} does not contain long induced paths, it might still be possible to apply a similar approach. Instead of considering paths ending in a node v, we consider a set of all cycle-dominators of node v in G_{npi}.

Reduction 2 (partial-domination-2). Let U be a set of all cycle-dominators of a node v in G_{npi}. If $N^{pi}(v) \subseteq \bigcup_{u \in U} N^{pi}(u)$, then v can be partially contracted.

Theorem 2. *Reduction rule 2 is safe. Checking a single node $v \in V$ can be done in time $O(|A|)$.*

Proof. The proof of safeness of this reduction rule is analogous to the proof of rule 1. The difference is that, instead of finding a set U corresponding to a path P in G_{npi}, we need to find a set of all cycle-dominators in G_{npi}.

Algorithm 1 Finding s-t dominators

```
1: P ← (s, u_1, ..., u_{p-1}, t)                          ▷ any path from s to t
2: Create 0-filled bitvector was and vector largest_j filled with values –1
3:
4: function DFS(ind, v)
5:     was[v] ← True
6:     for all w ∈ N^+_npi(v) do
7:         if w lies on path P then
8:             j ← index of w on path P                              ▷ u_j = w
9:             largest_j[v] ← max(largest_j[v], j)
10:        else if was[w] then                          ▷ w was considered earlier
11:            largest_j[v] ← max(largest_j[v], largest_j[w])
12:    for all w ∈ N^+_npi(v) do
13:        if not was[w] and w does not lie on path P then
14:            DFS(ind, w)
15:            largest_j[v] ← max(largest_j[v], largest_j[w])
16:    return                                                  ▷ backtrack
17:
18: dominators ← ∅
19: for all 1 ≤ i < p do
20:    Run search by calling DFS(i, u_{i-1})
21:    if largest_j[u_{i-1}] ≤ i then
22:        add u_i to dominators
23:    largest_j[u_i] ← max(i, largest_j[u_{i-1}])
24: return dominators
```

To find a set of cycle-dominators of node v, let us consider a more general algorithm given by Algorithm 1, which finds, for given $s, t \in V$, all nodes that dominate s with respect to t. By adding to G_{npi} two new nodes v_+ and v_-, where $N^+(v_+) = N^+_{npi}(v)$ and $N^-(v_-) = N^-_{npi}(v)$, and running the algorithm for $s = v_+$ and $t = v_-$, we find all cycle-dominators of v. At the beginning, an arbitrary path $P \leftarrow (u_0, \ldots, u_p,)$ from $u_0 = s$ to $u_p = t$ is found, then we start iterating over nodes on the path. We want to find for each node $u_i \in P$ the largest index j such that u_j is reachable from nodes $\{u_0, \ldots, u_i\}$ in graph $G[A \setminus (A' \cup \{(u_i, u_{i+1})\})]$, where $A' = \{(x, y) : x \in \{u_{i+1}, \ldots, u_p\}, y \in N^+(x)\}$. The crucial point is to observe that this can be done by running the DFS using only nodes that were not visited in previous DFS calls. To achieve that, we store for each node $u \in V$ the value $largest\_j[u]$ (written briefly $lj[u]$ from now on)

equal to the largest j as described earlier. When considering node v in the DFS call, we find the value $lj[v] = \max(X, Y)$, where X is the largest of values $lj[w]$ among visited earlier neighbors w of node v, and Y is the largest index j such that $u_j \in N^+(v)$ (or –1 if such index does not exist). It remains to be seen that each node $u_i \in P$ for which $lj[u_{i-1}] \leq i$ dominates node s with respect to t, that is every path from s to t contains u_i. If u_i would not be a dominator, then there would exist a path P' from s to t that did not contain u_i. But then, for some node u_k with $k < i$ (as an index k we can take, e.g., the largest integer smaller than i for which $u_k \in P'$) we would have to get $lj[u_k] > i$. This is impossible, as $lj[u_i] \geq lj[u_j]$ for any $j < i$. Since every node in the graph is visited only once during all calls to DFS for subsequent nodes on path P, and we need to iterate only once over neighbors of each node, the complexity of the algorithm is $O(\sum_{v \in V} |N^+(v)|) = O(|A|)$.

Reduction rule 2 is a generalization of rule 1, but cannot be checked as efficiently. The linear-time algorithm for finding all cycle-dominators constitutes also an improvement of an algorithm proposed in work [4]. Using our approach, the complexity of the rule "Rule DFVS 2" from that paper can be improved from $O(n^2(n+m))$ to $O(nm)$, which makes it feasible for much larger graphs.

Rules 1 and 2 required to find a subset or the full set of cycle-dominators of a given node. Both rules would remain valid if we considered as dominators of node u not nodes that belong to every cycle containing u, but nodes that belong to every chordless cycle containing u. Finding chordless cycles, however, is computationally difficult. We therefore consider the following algorithm:

Reduction 3 (partial-domination-3). Let L be a fixed integer. For a given node $u \in V$ run a depth-first search with backtracking to find the intersection U of all cycles that do not contain a chord among its first L (at most) nodes. Apply rule 2 to the set U (in rule 2 set U contains all cycle-dominators for a given node, here it is a subset of all chordless-cycles-dominators).

We omit the proof of safeness of this reduction, as it is very easy, for the reduction is conceptually a straightforward realization of the brute-force search for all chordless cycles containing a given node, restricted to at most L initial nodes on a cycle. By applying several software-engineering optimizations we are able to run the rule for $L = 12$ on most graphs, as for graphs that do not contain chordless-cycle-dominators, it can be usually quickly determined that the intersection U is empty, and the search can be terminated. There are graphs, however, where the search for such a large value of L takes too long and we terminate the search after a fixed time period.

4 Experiment

In this section we provide results of the conducted computational experiment. For evaluation we used all 400 graph instances from the PACE 2022 contest [9]. This set contains, among others, real-world instances representing web graphs,

social networks, and autonomous system graphs, as well as many random graphs generated according to various models. Used graphs contained up to $5 \cdot 10^6$ arcs.

We implemented three preprocessing procedures, called Set 1, Set 2, and Set 3, each based on a different collection of data reduction rules. Set 1 comprised only basic reduction rules: loop removal, parallel arc elimination, and the single-node neighborhood rule. Set 2 additionally incorporated several previously known reduction rules: core, dome, pie, and in-out-clique, as well as the folding, twin, desk and domination rule, the last four restricted to subgraphs in which the required vertices and their neighbors are pi-nodes (see [1,2,4,12,13,14,16] for more details). This configuration was used to produce instances suitable for evaluating the impact of partial-domination-based reductions on already preprocessed graphs. Since such graphs underwent substantial initial simplification, they are expected to be considerably less susceptible to further reductions than the unprocessed structures. Set 3 extended Set 2 by including partial-domination-based reduction rules introduced and described in this paper. It is necessary to mention that rule 2 was used only when the number of arcs in a graph did not exceed $5{\cdot}10^4$, as its complexity $O(|V|{\cdot}|A|)$ makes it inefficient for larger graphs. To assess effectiveness of the sets of reduction rules we use the reduction ratio measure, defined as the value $1 - \frac{|V_{G'}|}{|V_G|}$, where G' is the graph obtained by preprocessing G. A summary of obtained results is shown in Table 1.

Table 1: Comparison of results obtained by application of three sets of reduction rules to all 400 used graph instances.

	Set 1	Set 2	Set 3
Fully solved instances	2	84	94
Instances improved	356	385	385
Average reduction ratio	0.271	0.589	0.597
Median reduction ratio	0.092	0.700	0.724
Average reduction time	0.18 sec	5.14 sec	66.86 sec
Median reduction time	0.01 sec	0.26 sec	3.41 sec

By applying rules from the three sets, we were able to improve 356, 385, and 385 instances, respectively, out of the 400 test instances used. Among these, 2, 84, and 94 instances were fully solved, meaning that an optimal solution was obtained using only data reduction rules. Thus, for Set 3 we notice a significant 12% increase in the number of fully solved instances over Set 2. This comes at a cost of a longer average computational time, 66.86 vs. 5.14 seconds. We observe, however, that such a large difference is mainly caused by a few outliers, and the median time remains feasible for both sets, 3.41 and 0.26 seconds. The average reduction ratio values are almost identical (0.597 vs. 0.589). This might be, at first glance, misleading. If, e.g., Set 2 reduced a graph with million nodes to a graph with 1000 nodes, and Set 3 reduced it further to 500 nodes - thus achieving a huge 50% improvement over Set 2 - the reduction ratios for the two sets would equal 0.999 and 0.9995, respectively. If the impact of the sets of rules is measured only for the 316 graphs for which Set 2 did not fully solve the instance and compared relative to graph sizes obtained for Set 2, we observe that the

reduction rules proposed in this paper yielded ca. 8.1% increase in the reduction ratio - a fairly reasonable result, since the improved graphs have already been subjected to a strong preprocessing.

To measure the sole influence of each rule 1–3, we preprocessed graphs using Set 2 extended by a single rule, for which the impact we wanted to measure. For this comparison we considered only graphs where Set 2 yielded better results than Set 1. Such a procedure allows for a clearer comparison, as it effectively removes graphs with highly complex structures, for which one could expect that no reduction rule would be applicable. The largest improvement was obtained for rule 2 with the average reduction ratio approximately 0.220, over two times the value than for the other two rules (0.099 for rule 3 and 0.092 for rule 1). The smallest impact on the average computation time can be observed for rule 1, roughly 92% larger for rule 2, and 2.15 times larger for rule 3. Median values are roughly 7x smaller than the average times, which indicates that there are outliers (notably, the two largest average times equalled 1497.02 and 669.28 seconds), for which application of our rules remains very time-consuming.

Table 2: Comparison of the impact of particular reduction rules on the graph reduction ratio. Only 255 graphs that were not fully solved by Set 2 were considered and subjected to further preprocessing. Obtained reduction ratio values are relative to sizes of the graphs preprocessed using Set 2. Median reduction ratios are not presented, as they were very small and did not exceed 0.005.

	Set 2 + rule 1	Set 2 + rule 2	Set 2 + rule 3
Instances improved	142	155	145
Average reduction ratio	0.092	0.220	0.099
Average reduction time	78.06 sec	150.40 sec	168.73 sec
Median reduction time	12.40 sec	23.69 sec	22.37 sec

5 Conclusions

In the paper we introduced a new concept related to domination, called partial domination. Based on this concept, we provided data reduction rules for the Directed Feedback Vertex Set problem, which can significantly simplify a graph's structure. They also require less locality to be applicable than typical domination-based rules. We also provided an efficient linear-time algorithm for finding, for a given pair of nodes s, t, all dominators of s with respect to t. This deceptively simple algorithm constitutes an improvement of an algorithm for finding cycle-dominators, used in other works, reducing the complexity by an order of magnitude, from $O(nm)$ to $O(m)$. Finally, we provided results of a computational experiment for evaluating the designed reduction rules on a wide variety of graph classes, demonstrating 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., et al.: Solving directed feedback vertex set by iterative reduction to vertex cover. In: Proceedings of SEA 2023. 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: Proceedings of ALENEX 2023. pp. 53–64 (2023). https://doi.org/10.1137/1.9781611977561.ch5
3. 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)
4. Dirks, J., Gerhard, E., Grobler, M., Mouawad, A., Siebertz, S.: Data reduction for directed feedback vertex set on graphs without long induced cycles. In: Proceedings of SOFSEM 2024. pp. 183–197 (2024). https://doi.org/10.1007/978-3-031-52113-3_13
5. Downey, R., Fellows, M.: Fixed-parameter intractability. In: Proceedings of SCT 1992. pp. 36–49 (1992). https://doi.org/10.1109/SCT.1992.215379
6. Dvořák, W., Ordyniak, S., Szeider, S.: Augmenting tractable fragments of abstract argumentation. Artificial Intell. **186**, 157–173 (2012)
7. Garey, M., Johnson, D.: Computers and Intractability. A Guide to the Theory of NP-Completeness. W.H. Freeman and Company, San Francisco (1979)
8. Georgiadis, L., Tarjan, R., Werneck, R.: Finding dominators in practice. J. Graph Algorithms Appl. **10**, 69–94 (2006)
9. Großmann, E., Heuer, T., Schulz, C., Strash, D.: The PACE 2022 Parameterized Algorithms and Computational Experiments Challenge: Directed Feedback Vertex Set. In: Proceedings of IPEC 2022. pp. 26:1–26:18 (2022). https://doi.org/10.4230/LIPIcs.IPEC.2022.26
10. Harel, D.: A linear algorithm for finding dominators in flow graphs and related problems. In: Proceedings of STOC 1985. pp. 185–194 (1985). https://doi.org/10.1145/22145.22166
11. Hudli, A., Hudli, R.: Finding small feedback vertex sets for VLSI circuits. Microprocess. Microsyst. **18**, 393–400 (1994)
12. Kiesel, R., Schidler, A.: A dynamic MaxSAT-based approach to directed feedback vertex sets. In: Proceedings of ALENEX 2023. pp. 39–52 (2023). https://doi.org/10.1137/1.9781611977561.ch4
13. Levy, H., Low, D.: A contraction algorithm for finding small cycle cutsets. J. Algorithms **9**, 470–493 (1988)
14. 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)
15. Lokshtanov, D., Ramanujan, M., Saurabh, S.: When recursion is better than iteration: A linear-time algorithm for acyclicity with few error vertices. In: Proceedings of SODA 2018. pp. 1916–1933 (2018). https://doi.org/10.1137/1.9781611975031.125
16. Meiburg, A.: Reduction rules and ILP are all you need: Minimal directed feedback vertex set (2022). https://doi.org/10.48550/arXiv.2208.01119
17. Parotsidis, N., Georgiadis, L.: Dominators in directed graphs: A survey of recent results, applications, and open problems. In: Proceedings of ISCIM 2013. pp. 15–20 (2013), https://dspace.epoka.edu.al/handle/1/836
18. Xiong, Z., Xiao, M.: A simplified parameterized algorithm for directed feedback vertex set. In: Proceedings of SOSA 2025. pp. 378–384 (2025). https://doi.org/10.1137/1.9781611978315.29