# Description of the G$^2$OAT Solver for PACE 2022 Heuristic Track


**Radovan Červený**
Faculty of Information Technology, Czech Technical University in Prague

**Michal Dvořák**
Faculty of Information Technology, Czech Technical University in Prague

**Xuan Thang Nguyen**
Faculty of Information Technology, Czech Technical University in Prague

**Jan Pokorný**
Faculty of Information Technology, Czech Technical University in Prague

**Lucie Procházková**
Faculty of Information Technology, Czech Technical University in Prague

**Jaroslav Urban**
Faculty of Information Technology, Czech Technical University in Prague

**Václav Blažej**
Faculty of Information Technology, Czech Technical University in Prague

**Dušan Knop**
Faculty of Information Technology, Czech Technical University in Prague

**Šimon Schierreich**
Faculty of Information Technology, Czech Technical University in Prague

**Ondřej Suchý**
Faculty of Information Technology, Czech Technical University in Prague



## Abstract

In this paper, we give a brief description of our heuristic solver for the Directed Feedback Vertex Set (DFVS) problem. This solver was written for the heuristic track of the 2022 PACE Challenge. The solver consists of two phases. First, we compute an initial solution using the highest-degree heuristic together with some reduction rules. In the second phase, we try to improve the solution constructed in the first phase by applying simple local operations to the solution. In order to fit the time limit, during computation we dynamically adjust the order of reduction rules, the rate of progress of the highest-degree heuristic, and the effort expended during solution improvement.



**2012 ACM Subject Classification** Theory of computation → Graph algorithms analysis

**Keywords and phrases** Directed Feedback Vertex Set, Vertex Cover, heuristic solver, kernelization

**Digital Object Identifier** 10.4230/LIPIcs.CVIT.2016.23

**Supplementary Material** Source code hosted on Zenodo (`10.5281/zenodo.6637495`) and public repository (`gitlab.fit.cvut.cz/pace-challenge/2022/goat/heuristic`).


## Preliminaries

Let $D = (V, E)$ be a directed graph (digraph). We call an edge $(u, v) \in E$ *two-way* if $(v, u) \in E$, otherwise we call it *one-way*. Let $E^{\Updownarrow}$ be the set of two-way edges, $E^{\uparrow}$ be the set of one-way edges, and $D^{\uparrow} = (V, E^{\uparrow})$. If $E = E^{\Updownarrow}$, then we have an instance of Vertex Cover (VC). For a vertex $v \in V$ we call $N^{\Updownarrow}(v) = \{u : (u, v) \in E^{\Updownarrow}\}$ the *two-way neighborhood* of $v$ and $N^{\uparrow}(v) = \{u : |\{(u, v), (v, u)\} \cap E| = 1\}$ the *one-way neighborhood* of $v$. *Bridging a vertex* $v$ means that we add an edge from each predecessor of $v$ to each successor of $v$ and remove $v$ from the digraph.

## Solver Overview

The solver is split into two phases. In the first phase, we find some good enough solution through controlled use of reduction rules and a max-degree heuristic. In the second phase, we try to improve the solution by simple local changes to the solution.

**Details of the First Phase.** The algorithm iterates the following two steps until the digraph is a DAG. The first step is to exhaustively apply some reduction rules and the second step is to remove some number of the highest degree vertices from the digraph and put them into the solution.

The reduction rules we use are described below. According to our measurements the order in which the reduction rules are applied has a significant impact on the total running time. Therefore, during the computation, we dynamically rearrange their order such that the most used reduction rules are applied first.

In the second step, we remove some number $q$ of the highest degree vertices which we insert into the solution. Again, during the computation, we control the number $q$ in order to fit the time limit. In each iteration, we measure the progress we made (the number of vertices that were deleted) and the time it took to compute the iteration. We use this information to decide whether can afford to decrease $q$ or we need to increase $q$.

**Details of the Second Phase.** We take the solution $S$ constructed for the digraph $D$ in the first phase and perform the following two operations until the solution can no longer be improved or we run out of time.

1. For each vertex $v \in S$ (in the order as they were added to $S$), we take $S' = S \setminus \{v\}$ and check whether $S'$ is still a solution. If it is the case, we continue with $S'$ instead of $S$. We repeat this operation until no vertex of $S$ satisfies the conditions for removal.
2. For each vertex $v \in S$, we identify a set of vertices $X_v \subseteq V(G) \setminus S$ such that $(S \setminus \{v\}) \cup \{x\}$ is a solution for $D$ for each $x \in X_v$. We then take $S^* = S \cup \bigcup_{v \in S} X_v$ and run the first operation with $S^*$. If this leads to a larger solution than before, we terminate the solver.

## Reduction rules

We use the following reduction rules in our solver.

**Known DFVS reduction rules.** Following Levy and Low [1], we remove sources and sinks, if a vertex has exactly one incoming edge or exactly one outgoing edge, then we contract the edge, and we add self-loop vertices to solution. As used by Lin and Jou [2], we remove edges that are not contained in any directed cycles of $D^{\uparrow}$ and use their CORE rule.

**Known VC reduction rule adjusted for DFVS.** The following is a modification of a rule by Robson et al. [3].

▶ **Reduction Rule 1** (Vertex Domination). *Let $u, v$ be two vertices connected by a two-way edge. If every successor of $u$ is also successor of $v$ and every predecessor of $u$ is also predecessor of $v$, add $v$ to solution.*

**Additional rules.** We are not aware of the following rules being described in the literature.

▶ **Reduction Rule 2** (One-way neighborhood is a two-way clique). *If there is a vertex $v \in V$ such that $N^{\uparrow}(v)$ is a clique on two-way edges, then remove all one-way edges incident to $v$.*

▶ **Reduction Rule 3** (Useless Vertex). *If $v \in V$ is a vertex such that its bridging would not introduce any new edges, then remove $v$.*

## References


1 Hanoch Levy and David W. Low. A contraction algorithm for finding small cycle cutsets. *Journal of Algorithms*, 9(4):470–493, December 1988. doi:10.1016/0196-6774(88)90013-2.

2 Hen-Ming Lin and Jing-Yang Jou. On computing the minimum feedback vertex set of a directed graph bycontraction operations. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 19(3):295–307, March 2000. doi:10.1109/43.833199.

3 John M. Robson. Algorithms for maximum independent sets. *Journal of Algorithms*, 7(3):425–440, September 1986. doi:10.1016/0196-6774(86)90032-5.