Orodruin: A Heuristic Solver for Directed Feedback Vertex Set

Sebastian Angrick
Hasso Plattner Institute, University of Potsdam

Ben Bals
Hasso Plattner Institute, University of Potsdam

Katrin Casel
Hasso Plattner Institute, University of Potsdam

Sarel Cohen
The Academic College of Tel Aviv-Yaffo, Israel

Tobias Friedrich
Hasso Plattner Institute, University of Potsdam

Niko Hastrich
Hasso Plattner Institute, University of Potsdam

Theresa Hradilak
Hasso Plattner Institute, University of Potsdam

Davis Issac
Hasso Plattner Institute, University of Potsdam

Otto Kißig
Hasso Plattner Institute, University of Potsdam

Jonas Schmidt
Hasso Plattner Institute, University of Potsdam

Leo Wendt
Hasso Plattner Institute, University of Potsdam

Abstract

This document describes the techniques we used and implemented for our submission the Parameterized Algorithms and Computational Experiments Challenge (PACE) 2022. The given problem is *Directed Feedback Vertex Set* (DFVS), where you are given a directed graph $G = (V, E)$ and want to find a minimum $S \subseteq V$ such that $G - S$ is acyclic. Our approach first generates an initial greedy solution. This solution is then checked for minimality under exclusion of a single node and exchange of two solution nodes for one new node.

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.CVIT.2016.23

Supplementary Material Source code (Software): `https://doi.org/10.5281/zenodo.6645245`
Source code (Software): `https://github.com/BenBals/mount-doom/tree/heuristic`

1 Preliminaries

Let $G = (V, E)$ be a directed graph. The Directed Feedback Vertex Set problem asks to find a minimum $S \subseteq V$, such that $G - S$ is acyclic.

Let $v, w \in V$. We define $N^+(v)$ as the outgoing neighbors of v and $N^-(v)$ as the incoming neighbors. We call an edge $vw \in E$ bidirectional if $wv \in E$ as well. Let $\mathrm{PIE} \subseteq E$ be the set of all bidirectional edges and let $B \subseteq V$ be the set of all vertices only incident to bidirectional edges. We define the bidirectional neighbors $N(v)$ as those which are incident using bidirectional edges. Additionally, we call $D \subseteq V$ a diclique, if all $u \in D$ have $D \setminus \{u\} \subseteq N(u)$.

Finally, let $v \in V$ be given. Let $V' = V \setminus \{v\}$ and $E' = (E \cap (V' \times V')) \cup (N^-(v) \times N^+(v))$. We call $G' = (V', E')$ the graph obtained from G by short cutting v. In light of the DFVS, this is equivalent to adding the assumption $v \notin S$.

2 Reduction rules

We apply two reduction rules known from literature. These rules can be found in [4] and we adopt their nomenclature.

© Sebastian Angrick, Ben Bals, Katrin Casel, Sarel Cohen, Niko Hastrich, Theresa Hradilak, Davis Isaac, Otto Kißig, Jonas Schmidt and Leo Wendt;
licensed under Creative Commons License CC-BY 4.0
42nd Conference on Very Important Topics (CVIT 2016).
Editors: John Q. Open and Joan R. Access; Article No. 23; pp. 23:1–23:4
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

PIE. Recall that PIE is the set of bidirectional edges. Now consider any edge uv between different strongly connected components in $G - \text{PIE}$. Any cycle using this edge must therefore use at least one bidirectional edge, which must be covered anyways, so we can safely 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 $G - N(v)$ then $S' \cup N(v)$ is optimal for G [4]. 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.

3 Solver Description

After exhaustively applying the reductions we described in Section 2, we produce two solutions, one by a greedy procedure and another by a reduction to vertex cover. The better solution of both approaches is minimized further by applying 2-1 swaps until the timelimit is hit.

3.1 Greedy Routine

For the initial solution, we start with an empty set and greedily add to it the vertex of highest degree until we obtain a feasible DFVS. Checking for feasibility here means searching for a cycle in the graph. To speedup this procedure, we only search for a new cycle, once the old one is covered. Additionally, after a fixed number of nodes are taken into the solution, we reapply all reduction rules. After the initial solution is generated, we remove nodes that do not reintroduce a cycle when added back to the graph. This ensures that we create an inclusion-minimal solution.

3.2 Reduction to Vertex Cover

First note that if a graph contains only bidirectional edges, we can easily reduce the DFVS instance to a vertex cover instance by turning bidirectional edges into undirected edges. Initially, we find a vertex cover S in $G[\text{PIE}]$ and check, whether S is a DFVS for G. If not, we find a set of vertex-disjoint cycles C in $G - S - PIE$ using a DFS. All cycles in C are not covered by S, so we add a gadget to each cycle to ensure, that in the modified graph, there is an optimal vertex cover, which includes a $v \in S$. Finally, we iterate on the modified graph until the vertex cover is also a DFVS. Note, that this can happen multiple times since our choice of C does not guarantee that all cycles in G are covered. When we hit an internal timelimit before finding a feasible DFVS, we apply our greedy approach described in Section 3.1 for the remaining graph.

Let $G = (V, E)$ be an undirected graph and $S \subseteq V$ be a cycle. Our goal is to find the minimum vertex cover in G that also contains a vertex in S. To achieve this, we add a clique of size $|S|$ to G and connect it one-to-one with S. We call the modified graph G'. Consider any optimal vertex cover C in G'. Then, C contains at least $|S| - 1$ vertices in the new clique. Also, C must cover all edges between V and the clique, so it must contain at least one vertex in S or all vertices in the clique. If C contains all vertices in the clique, we exchange one of these vertices for a vertex in S and obtain an optimal vertex cover C' in G' with $C' \cap S \neq \emptyset$. Thus, $C' \cap V$ is a optimal vertex cover of G that also contains a vertex of S.

To solve the vertex cover instance, we first use a kernelization procedure implemented by the winning solver of the 2019 PACE challenge [3]. Then, we use a local-search solver [1] on this kernel.

3.3 2-1 swaps

We apply a local-search approach to improve our current solution, named 2-1 swaps. As the name suggests, its goal is to find 2 vertices from a given DFVS-solution and replace them with one vertex to form a smaller solution. The idea uses the notion of skew separator from Chen et al. [2].

Consider a feasible and minimal solution set S to the DFVS problem, i.e. for all $v \in S$ the set $S - v$ is infeasible. Our goal is to find $v, w \in S$ and $u \notin S$ such that $(S - \{v, w\}) \cup \{u\}$ is a feasible solution. Whenever we find such a triple (v, w, u), we swap $\{v, w\}$ with u in S and repeat the procedure.

We now describe how to find such a triple. First, for each vertex $v \in S$, we find all vertices $u \in G - S$, called candidates, such that $(S - \{v\}) \cup \{u\}$ remains feasible. Let the candidates of a given solution vertex v be C_v.

In order to find C_v, we split the vertex v into two vertices $v_{\mathrm{out}}, v_{\mathrm{in}}$ with out- or ingoing edges of v only. Let $G^v := (G - S) \cup \{v_{out}, v_{in}\}$. Note that there is no edge between v_{out} and v_{in} in either direction in G^v. The candidates are now the set of all vertices x such that x hits all $v_{out} \rightsquigarrow v_{in}$ paths in G^v.

Note that in $G - S$, the vertices in C_v have a topological ordering. Also, notice that each cycle in $(G - S) \cup \{v\}$ has to be hit by each vertex in C_v. This implies that C_v has a unique topological ordering in $G - S$. Let τ_v be this ordering.

We say that the ordered triple (v, w, u) is a swap triple if there is no $v_{out} \rightsquigarrow v_{in}$ path in G^v, there is no $w_{out} \rightsquigarrow w_{in}$ path in G^w, and there is no $v \rightsquigarrow w$ path in $(G - (S \cup \{u\})) \cup \{v, w\}$. Observe that a valid 2-1 swap is possible if and only if there exists such a swap triple. For each vertex v, we now find a w and u that give a swap triple (v, w, u) if such a w and u exist, as follows. Observe that (v, w, u) is a swap triple if and only if $u \in C_v \cap C_w$ and u hits all $v \rightsquigarrow w$ paths in $(G - S) \cup \{v, w\}$. Let W be the set of all vertices not reachable from v in $G - C_v$. For each $w \in W$, let c_w be the first vertex in the ordering τ_v such that there is a $c_w \rightsquigarrow w$ path in $(G - C_v) \cup \{c_w\}$. If there is a vertex u in $C_v \cap C_w$ such that $u \preceq c_w$ in τ_v then we return (v, w, u) as a swap triple.

First, we prove that such a (v, w, u) is indeed a swap triple. Suppose this is not the case. Since $u \in C_u \cap C_w$, this means that there is a $v \rightsquigarrow w$ path P in $(G - S) \cup \{v, w\}$ not hit by u. The path P contains at least one vertex from C_v as w was not reachable from v in $G - C_v$. Let c be the last vertex in P that is from C_v. Then there is a $c \rightsquigarrow w$ path in $(G - C_v) \cup \{c\}$, and hence $c_w \preceq c$ by the choice of c_w. This implies that $u \preceq c$ by choice of u. Let P_1 be the sub-path $v \rightsquigarrow c$ of P. Since u does not hit P, the path P_1 does not contain u. We know there is at least one cycle in $(G - S) \cup \{v\}$ containing $C_v \cup \{v\}$, and hence there should be a $c \rightsquigarrow v$ path in $(G - S) \cup \{v\}$, say P_2. The concatenation $P_1 \cup P_2$ gives a cycle in $G - S \cup \{v\}$ and hence needs to be hit by each vertex in C_v, in particular u. This means that u should hit P_2, but this contradicts that $u \preceq c$ in τ_v.

Next, we prove that if there is a swap triple then our procedure will find one. Suppose we do not find a swap triple and there is a swap triple (v, w, u). Let us consider the processing of u in our procedure. We have $w \in W$ as u hits all $v \rightsquigarrow w$ paths in $(G - S) \cup \{v, w\}$. If $u \preceq c_w$ then since $u \in C_v \cap C_w$, we would have found the swap triple (v, w, u). Thus $c_w \prec u$. Then u cannot be in any $v \rightsquigarrow c_w$ path in $(G - S) \cup \{v, w\}$. But then u has to hit all $c_w \rightsquigarrow w$ paths in $(G - S) \cup \{v, w\}$ as u hits all $v \rightsquigarrow w$ paths in $(G - S) \cup \{v, w\}$. This is a contradiction to the choice of c_w.

References

1. Shaowei Cai, Kaile Su, Chuan Luo, and Abdul Sattar. Numvc: An efficient local search algorithm for minimum vertex cover. *J. Artif. Int. Res.*, 46(1):687–716, jan 2013. URL: https://dl.acm.org/doi/10.5555/2512538.2512555.
2. Jianer Chen, Yang Liu, Songjian Lu, Barry O'Sullivan, and Igor Razgon. A fixed-parameter algorithm for the directed feedback vertex set problem. *J. ACM*, 55:21:1–21:19, 2008.
3. Demian Hespe, Sebastian Lamm, Christian Schulz, and Darren Strash. Wegotyoucovered: The winning solver from the pace 2019 challenge, vertex cover track. In *CSC*, 2020.
4. 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.