The PACE 2022 Parameterized Algorithms and Computational Experiments Challenge: Directed Feedback Vertex Set

Ernestine Großmann
Universität Heidelberg, Germany

Tobias Heuer
Karlsruhe Institute of Technology, Germany

Christian Schulz
Universität Heidelberg, Germany

Darren Strash
Hamilton College, Clinton, NY, USA

Abstract

The Parameterized Algorithms and Computational Experiments challenge (PACE) 2022 was devoted to engineer algorithms solving the NP-hard Directed Feedback Vertex Set (DFVS) problem. The DFVS problem is to find a minimum subset $X \subseteq V$ in a given directed graph $G = (V, E)$ such that, when all vertices of X and their adjacent edges are deleted from G, the remainder is acyclic.

Overall, the challenge had 90 participants from 26 teams, 12 countries, and 3 continents that submitted their implementations to this year's competition. In this report, we briefly describe the setup of the challenge, the selection of benchmark instances, as well as the ranking of the participating teams. We also briefly outline the approaches used in the submitted solvers.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms; Theory of computation → Graph algorithms analysis

Keywords and phrases Feedback Vertex Set, Algorithm Engineering, FPT, Kernelization, Heuristics

Digital Object Identifier 10.4230/LIPIcs.IPEC.2022.26

Funding Partially supported by DFG grant SCHU 2567/3-1.

Acknowledgements The PACE challenge was supported by Networks [1]. The prize money (€4000) was generously provided by Networks [1], an NWO Gravitation project of the University of Amsterdam, Eindhoven University of Technology, Leiden University and the Center for Mathematics and Computer Science (CWI). We are grateful to the whole optil.io team, led by Szymon Wasik, and especially to Jan Badura and Artur Laskowski for the fruitful collaboration and for hosting the competition at the optil.io online judge system.

1 Introduction

Over the last two decades, significant advances have been made in the design and analysis of fixed-parameter algorithms for a wide variety of graph-theoretic problems. This has resulted in an algorithmic toolbox that is by now well-established. Recently, these theoretical algorithmic ideas have received attention from the practical perspective [2, 3, 25, 40, 50]. A large part of this effort is driven by the Parameterized Algorithms and Computational Experiments Challenge (PACE) which was conceived in Fall 2015 to deepen the relationship between parameterized algorithms and practice. Topics from multivariate algorithms, exact algorithms, fine-grained complexity, and related fields are in scope. The mission of PACE is to bridge the divide between the theory of algorithm design and analysis, and the practice of algorithm engineering, inspire new theoretical developments, investigate in how far theoretical algorithms from parameterized complexity and related fields are competitive in practice, produce universally accessible libraries of implementations and repositories of benchmark instances as well as to encourage the dissemination of these findings in scientific papers. In each iteration of the challenge [20, 21, 12, 24, 51, 47] participants of the competition have been asked to provide implementations for one or two specifically chosen problems. Moreover, there are often two types of tracks: a track in which participants have to provide algorithms that solve a problem to optimality and a track in which heuristic solvers are allowed (and solutions are ranked accordingly). The challenge tackled already a wide range of problems. In previous iterations, the challenge tackled the following problems:

- First Iteration: Treewidth and Undirected Feedback Vertex Set [20]
- Second Iteration: Treewidth and Minimum Fill-In [21]
- Third Iteration: Steiner Tree [12]
- Fourth Iteration: Vertex Cover and Hypertree Width [24]
- Fifth Iteration: Treedepth [51]
- Sixth Iteration: Cluster Editing [47]

Since its inception, PACE challenges have established themselves as highly competitive with typically around 50 participants submitting their solvers from all over the world. Moreover, the challenges have already had a significant impact on the community as a whole. There is a wide range of research articles based on concrete implementations competing in previous editions of PACE that were published in prestigious conferences on algorithm engineering such as ACDA, ALENEX, ESA Track B, SEA, and WADS. Moreover, the challenges already successfully inspired new research, i.e. after the challenge there are also new results that improve on the previously best implementations from a particular challenge.

In this article, we report on the seventh iteration of the PACE implementation challenge. The problem chosen for this year's iteration has been the directed feedback vertex set problem. The challenge featured two tracks: an exact track and a heuristic track. In the *exact track*, the task was to find an optimal solution of each directed feedback vertex set instance within a time limit of 30 minutes. In the *heuristic track*, the task was to compute a valid solution that was as small as possible within a time limit of ten minutes.

The PACE 2022 challenge was announced and tracks were specified in September 2021. In January 2022, public instances were made available to the challenge participants. In March 2022, challenge participants were able to submit their solvers into the optil.io platform in which they could test their solvers on the instances that were publicly available. The platform also provided a provisional ranking. The final version of the submissions was due 1st June 2022. Afterwards, the submissions were evaluated on the publicly available as well as the private (hidden) instances. The results were announced in July 2022. The award ceremony took place during the International Symposium on Parameterized and Exact Computation (IPEC 2022).

2 Directed Feedback Vertex Set

The Directed Feedback Vertex Set (DFVS) problem is to find a minimum subset $X \subseteq V$ in a given directed graph $G = (V, E)$ such that, when all vertices of X and their adjacent edges are deleted from G, the remainder is acyclic. Thus a feedback vertex set of a graph is a set of vertices whose deletion leaves a graph acyclic. Figure 1 shows an example.

The DFVS problem has a wide range of applications including deadlock resolution [33], program verification [57] and VLSI chip design [54]. The decision variant of DFVS (asking if there exists a feedback vertex set of size at most k) is NP-complete [46] even if restricted to

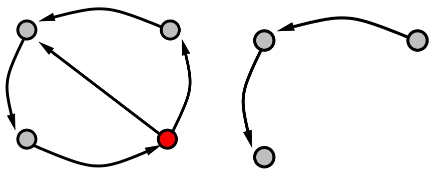

Figure 1 An input graph and a feedback vertex set (red) is shown on the left. In this example, deleting/removing the red vertex and its edges in the left graph results in the graph on the right hand side and leaves the remaining graph without any cycles.

graphs with maximum in- and out-degree two. The optimization variant of DFVS can be solved in $O^*(1.9977^n)$ time due to an algorithm by Razgon [69]. Chen et al. [17] showed that the problem is fixed-parameter tractable if parameterized with the solution size k, giving an algorithm with running time $O(4^k k! k^3 n^4) = 4^k k! n^{O(1)}$. With improvements to solving the Skew Edge Multicut problem, this running time is reduced to $O(4^k k! k^4 nm)$ [18, Corollary 8.47]. Lokshtanov et al. developed an improved algorithm with running time $O(4^k k! k^5 (n+m))$ [60], which has only linear dependence on the input size. It is open whether DFVS has a polynomial kernel in k, however a polynomial kernel exists when parameterized on the feedback vertex set of the underlying undirected graph [10], and for the compound parameterization of k plus the size of a treewidth-η modulator for any constant η [61]. Note that the problem is equivalent to the edge-deletion variant commonly called Feedback Arc Set: there are reductions in both directions that preserve the value of the optimal solution and only increase the size of the graph (sum of vertices and edges) by a polynomial factor [27]. Faster algorithms exist for undirected graphs [49, 76, 29], as well as for orientations of complete graphs (called *tournament* graphs) [34, 29].

The best approximation algorithm for DFVS is due to Even et al. [27], who gave an algorithm with approximation factor $O(\min\{\log \tau^* \log\log \tau^*, \log n \log\log n\})$ where τ^* is a lower bound, such as the optimal fractional solution in the LP relaxation. By Karp's reduction [46], DFVS is APX-hard, meaning that there is no polynomial-time approximation scheme (PTAS) for DFVS assuming P$\neq$NP. Furthermore, assuming the Unique Games Conjecture, DFVS does not admit a polynomial-time constant factor approximation [37, 38]. However, Lokshtanov et al. give a 2-approximation algorithm for DFVS when the input is a tournament graph [59].

3 Challenge Setup

There were two tracks in which the participants could compete: an exact and a heuristic track. For each track the 200 instances were selected by the Program Committee (PC), half of them publicly available before the submission deadline. The instances were sorted by the time our internal solvers needed to solve the instance. In the testing phase the instances were evaluated on the online judging platform optil.io [75]. For the final evaluation, we tested the instances on a local machine: an AMD EPYC 7702P 64-Core CPU, 200W, 2.00GHz, 256MB L3 Cache, DDR4-3200, Turbo Core max. 3.35GHz. Both evaluations used the same time limits: 30 minutes for the exact track and 10 minutes for the heuristic track.

3.1 Track Descriptions

The exact and the heuristic track followed essentially the same rules as in previous iterations of PACE. We now shortly describe the tracks:

Exact Track. In this track submissions had to find an optimal (minimum) feedback vertex set within 30 minutes. We expected each submission to be an exact algorithm, although we did not ask for proof of it. If we found through code checks or experiments that the algorithm of a submission is not an exact algorithm, it was excluded from the track. If for some instance the program returned a solution that has not been optimal within the time limit, either because it is not minimum or not a feedback vertex set, then the submission has been disqualified. The ranking has been determined by the number of solved instances. In case of a tie, the winner has been determined by the time required to solve all instances.

Heuristic Track. In the heuristic track, submissions had to provide a feedback vertex set within 10 minutes for a given instance. The submissions have been ranked by the geometric mean over all instances of $100 \times \frac{\text{best solution size}}{\text{solution size}}$. Here, solution size is the size of the solution returned by the submission and best solution size is the size of the smallest solution known to the PC (which may not be optimal). If the output of the program turned out to be not a feedback vertex set (or there is no output on SIGKILL) for some instance, solution size for the instance has been considered as $|V|$.

3.2 Internal Solver

Our goal was to create instances that are easy to solve, as well as instances that are as challenging as possible for the submissions. We implemented several data reduction rules and heuristics which we then used in our ILP solver to compute optimal solutions for the instances described in Section 3.3. We will now give a brief description of the algorithms.

Reduction Rules. We implemented four data reduction rules which are also summarized in the work of Lin and Jou [58]. We first compute all strongly connected components (SCC) using Tarjan's algorithm [74] and remove all edges connecting two SCCs. We then solve each SCC separately. Further, we contract each node u with in-degree or out-degree one onto its unique predecessor or successor v. Intuitively, all cycles containing u also contain v. The last reduction rule creates an auxiliary graph G' by removing all undirected edges from the original graph (an edge $\{u, v\}$ is undirected if the graph contains the directed edges (u, v) and (v, u)) and then computes all SCCs of G'. An edge that connects two SCCs in G' can be removed from the original graph. For each undirected edge $\{u, v\}$ either u or v must be part of a DFVS (each undirected edge induces a cycle of size two) and therefore, the directed edges (u, v) and (v, u) are not part of the subgraph induced by removing any DFVS. Edges that connect two SCCs in G' are not part of a cycle when we remove one node of each undirected edge, and thus can be removed from the original graph. We apply the data reductions until none of them are applicable.

Random Walk Heuristic. A random walk on a directed graph $G = (V = \{v_1, \dots, v_n\}, E)$ can be modeled as a Markov chain with transition probabilities $p_{ij} = \frac{1}{d(v_i)}$ where $d(v_i)$ is the out-degree of node v_i. The stationary distribution $\pi = (\pi_1, \dots, \pi_n)$ of this Markov chain describes the probability distribution of visiting a node after a sufficiently long time (π is the solution of the linear equation $P\pi = \pi$). Moreover, π_i^{-1} represents the mean return time to node v_i in random walks. Thus, the stationary distribution π encodes information about the global structure of all cycles and a node v_i with the highest π_i value is very likely to be visited most frequently (and also contained in many cycles). A heuristic based on this idea was proposed by Lemaic and Speckenmeyer [70, 55]. We implemented a simple version of this algorithm that performs the random walk explicitly. We select a random start node and visit $10|V|$ nodes. We remove the node that is visited most often. Afterwards, we recompute the SCC that the corresponding node was part of and remove all edges connecting two SCCs. The algorithm terminates if the graph is acyclic.

Maximum Acyclic Subgraph Heuristic. The DFVS problem is equivalent to finding a set V' of maximum cardinality such that the subgraph $G[V']$ is acyclic. The set $V \setminus V'$ is then a minimum DFVS. If a graph $G = (V, E)$ is acyclic, then there exists a topological ordering $T = \langle v_1, \ldots, v_n \rangle$ of the nodes V such that for all edges $(v_i, v_j) \in E$ holds that $i < j$. Galinier et al. [32] propose a local search algorithm that constructs an acyclic subgraph $G[V']$ such that $|V'|$ is maximized. Consider an acyclic subgraph $G[V']$ with $V' \subseteq V$ and its topological ordering $T = \langle v_1, \ldots, v_{n'} \rangle$ with $n' = |V'|$. If we insert a node $u \notin V'$ into T after the position i, we have to remove all nodes in $V_{\text{in}}(u,i) := \{v_j \in V' \mid (v_j, u) \in E \ \wedge \ j > i\}$ and $V_{\text{out}}(u,i) := \{v_j \in V' \mid (u, v_j) \in E \ \wedge \ j \leq i\}$ from T such that T still represents a valid topological ordering of the subgraph $G[V'']$ with $V'' = (V' \cup \{u\}) \setminus (V_{\text{in}}(u) \cup V_{\text{out}}(u))$. Thus, we can efficiently evaluate if a node $u \notin V'$ increases the cardinality of the acyclic subgraph $G[V']$ by computing the *gain* $g(u,i) := 1 - |V_{\text{in}}(u,i)| - |V_{\text{out}}(u,i)|$.

Our local search algorithm uses the well-known label propagation heuristic [68, 79]. The algorithm works in rounds and in each round it visits the nodes in random order. We initially start with an empty topological ordering T ($V' = \emptyset$). If we visit a node $u \notin V'$, we insert u into T after position i that maximizes $g(u,i)$ and remove all nodes in $V_{\text{in}}(u,i)$ and $V_{\text{out}}(u,i)$ from T. Note that we only evaluate positions in $\{i \mid (v_i, u) \in E \ \vee \ (u, v_i) \in E\}$ and add u to T if $g(u,i) \geq 0$. Further, insertions with $g(u,i) = 0$ naturally perturb the solution and we observed that this enables frequent improvements also in later iterations of the algorithm. The algorithm terminates if we reach a predefined number of rounds ($= 200$). To maintain the topological ordering, we use a sparse table priority queue implementation [42] that provides (amortized) constant time operations for access, insertions and removals of nodes.

We additionally made two major improvements to the original algorithm proposed by Galinier et al. [32]. Both exploit the fact that the topological ordering of the subgraph $G[V']$ is not unique and therefore provide some flexibility in the ordering of the nodes in T. If we are not able to insert a node u into the topological ordering T (i.e., $g(u,i) < 0$ for all possible positions i), we shift all nodes $v_i \in T$ adjacent to u via an in-arc $(v_i, u) \in E$ to the left and all nodes $v_j \in T$ adjacent to u via an out-arc $(u, v_j) \in E$ to the right in T (both as far as possible such that T still represents a valid topological ordering of $G[V']$). If then the indices of all nodes in T adjacent via an in-arc to u are smaller than the ones adjacent via an out-arc to u, we can increase the cardinality of the acyclic subgraph by inserting u in between. We further diversify the search by periodically computing a random topological ordering of $G[V']$ using Kahn's algorithm [45] (every fifth round).

Both techniques significantly improved the solution quality of the original algorithm (more than 10% on most instances). In practice, this heuristic was often an order of magnitude faster than our random walk algorithm. We also see more potential in this method as the concepts of gains allow the development of more sophisticated local search techniques.

Exact Solver. For the exact track, we solved the instances using a branch-and-cut ILP formulation and used Gurobi as a solver. Moreover, the we integrate the data reduction rules described above into the solver. On the obtained irreducible instance, we run acyclic subgraph heuristics from above to get an initial feasible DFVS and provide the solution as an upper bound to the ILP solver that solves the following ILP to compute an optimal solution on the instance:

$$\begin{aligned} &\min \sum_{v \in V} x_v \\ &\text{s.t. } \sum_i x_{v_i} \geq 1 \quad \forall \text{ cycles } C = \{v_1, v_2, ..., v_k\} \text{ in } G. \end{aligned}$$

Note that the number of constraints here can be exponential. As this would result in an intractable ILP, we add constraints *lazily* as follows. Initially our solver adds all constraints for cycles of length two. In addition, for each node $u \in V$, we add three cycle constraints representing cycles in which u is the node with smallest ID (ensures that the cycle constraints are distinct). We will call such a cycle an elementary cycle of u. We then solve the ILP using Gurobi and check if the solution is a feasible DFVS, i.e. after removal of the solution vertices there is no cycle remaining. If this is the case, then the solution is also an optimal solution to our input instance. If it is not a DFVS then after removing the vertices from the graph there must be a cycle. We then add for each node u in the remaining graph one additional elementary cycle and repeat the process.

Surprisingly, our exact solver was able to compute optimal solutions for some of the real-world instances in the heuristic track with up to $500k$ edges within a few minutes. However, the running time increases drastically for denser graphs (even if they contain only a few thousand edges). Thus, we believe that the density of a graph is a good indicator for the hardness of an instance.

3.3 Instances

We obtained instances from a wide range of different sources. In particular, as there is a wide range of random graph models available [66], we generated instances from different graph classes using *KaGen* [30], included several real-world instances from public graph repositories, and lastly generated instances that are hard for typical heuristic solvers such as heuristics based on random walks.

Generation and Selection Process. Our instance generation and selection process worked as follows: for both the exact and the heuristic track we generated a very large set of instances. From the graphs that are bidirected, we removed a random amount of edges $p \in \{0, 10, 20, 30, 40, 50\}$ to also obtain directed instances from those models. The amount of instances that we generated internally has been much larger the necessary 200 instances for the public and private set of instances for each track. The instances that we generated are described by a wide-range of parameters of different graph families (see below for more details). From the large set, we filtered instances that had more than 1 000 strongly connected components, less edges than nodes and excluded instances that had a file size above 50MB. On the remaining set of instances, we ran our exact and heuristic solvers. For the exact track, we then further excluded instances that our solver could handle in less than a second. Afterwards, we sampled instances uniformly at random. In particular, we included easy instances that could be solved within a couple of seconds as well as instances that were hard to solve. For the heuristic track, we tried to include instances that are hard for heuristics. From the instances that our exact solver could solve in this track, we included the ones where the result of heuristic solver had significantly more vertices than the optimum solution. Here, we also included real-world instances as well as instances designed to be hard for heuristics.

The instances we used can be categorized as follows:

Erdős-Rényi Graphs. The first version of the Erdős-Rényi (ER) model was proposed by Gilbert [35] and is denoted as the $G(n, p)$ model. Here, each of the $n(n-1)/2$ possible edges of an n-node graph is independently sampled with probability $0 < p < 1$ (Bernoulli sampling of the edges).

The second version, proposed by Erdős and Rényi [26], is denoted as the $G(n, m)$ model. In the $G(n, m)$ model, we choose a graph uniformly at random from the set of all possible graphs which have n vertices and m edges.

Random Geometric Graphs. Random geometric graphs (RGGs) are bidirected spatial networks where we place n vertices uniformly at random in a d-dimensional unit cube $[0, 1)^d$. Two vertices $p, q \in V$ are connected by an edge iff their d-dimensional Euclidean distance $\text{dist}(p, q) = \sqrt{\sum_{i=1}^{d} (p_i - q_i)^2}$ is within a given threshold radius r. Thus, the RGG model can be fully described using the two parameters n and r. Note that the expected degree of any vertex that does not lie on the border, i.e. whose neighborhood sphere is completely contained within the unit cube, is $\bar{d}(v) = \frac{\pi^{\frac{d}{2}}}{\Gamma(\frac{d}{2}+1)} r^d$ [65]. Here, we used two and three dimensional random geometric graphs as available in KaGen.

Random Hyperbolic Graphs. Random hyperbolic graphs (RHGs) are bidirected spatial networks generated in the hyperbolic plane with negative curvature. Analogous to RGGs, RHGs are parameterized by the number of vertices n and a hyperbolic radius $R = 2 \log n + C$.[1] Additionally, this model is given a power-law exponent $\gamma \geq 2$. To generate a RHG graph, n vertices are placed on a disk of radius R in the hyperbolic plane.

Each vertex has an angular coordinate ϕ and a radial coordinate r. The angular coordinate is sampled uniformly at random from the interval $[0, 2\pi)$. The radial coordinate r is chosen using the probability density function

$$f(r) = \alpha \frac{\sinh(\alpha r)}{\cosh(\alpha R) - 1}.$$

The parameter $\alpha = \frac{\gamma - 1}{2}$ controls the growth of the random graph and determines the vertex density. Krioukov et al. [52, 36] show that for $\gamma \geq 2$ the degree distribution follows a power-law distribution with exponent γ. Two vertices p, q are connected iff their hyperbolic distance

$$\text{dist}_H(p, q) = \cosh r_p \cosh r_q - \sinh r_p \sinh r_q \cos |\phi_p - \phi_q|$$

is less than R. Therefore, the neighborhood of a vertex consists of all the vertices that are within the hyperbolic circle of radius R around it.

Random Delaunay Graphs. A d-simplex is a generalization of a triangle ($d = 2$) to d-dimensional space. A d-simplex s is a d-dimensional polytope, i.e. the convex hull of $d + 1$ points. The convex hull of a subset of size $m + 1$ of these $d + 1$ points is called an m-face of s. Specifically, the 0-faces are the vertices of s and the $(d - 1)$-faces are its facets. Given a d-dimensional point set $V = \{v_1, v_2, \ldots, v_n\}$ with $v_i \in \mathbb{R}^d$ for all $i \in \{1, \ldots, n\}$, a triangulation $T(V)$ is a subdivision of the convex hull of V into d-simplices, such that the set of the vertices of $T(V)$ coincides with V and any two simplices of T intersect in a common

[1] The parameter C controls the average degree $\bar{d}$ of the graph [52].

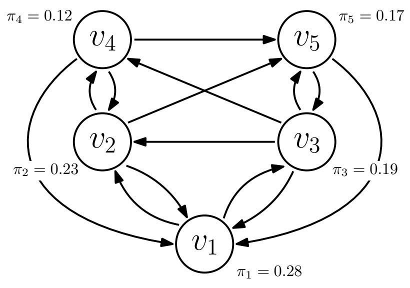

Figure 2 Graph that is hard to solve for heuristics based on random walks. The π_i values denote the stationary distribution if we interpret the graph as a Markov chain with transition probabilities $p_{ij} = \frac{1}{d(v_i)}$.

$d-1$ facet or not at all. The union of all simplices in $T(V)$ is the convex hull of point set V. A Delaunay triangulation $DT(V)$ is a triangulation of V such that no point of V is inside the circumhypersphere of any simplex in $DT(V)$.

Barabási-Albert Graph Model. Barabási and Albert [7] define the model that is perhaps most widely used because of its simplicity and intuitive definition: One starts with an arbitrary seed network consisting of nodes $0..n_0-1$ ($a..b$ is used as a shorthand for $\{a,\dots,b\}$ here). Nodes $i \in n_0..n-1$ are added one at a time. They randomly connect to d neighbors using *preferential attachment*, i.e., the probability to connect to node $j \leq i$ is chosen proportionally to the degree of j. The seed graph, n_0, d, and n are parameters defining the graph family. Since all edges only point to nodes with a smaller node ID the resulting directed network is acyclic. Hence, we generated graphs according to the Barabási Albert and modified them to become cyclic in the following way:
First, we computed a topological ordering of the instance. Then we inserted $p \cdot m$ random edges, where $p \in [0.05, 0.2]$. More precisely, we picked a random node to be a source, and afterwards picked a random node with a smaller number in the topological ordering.

Real-World Instances. This category includes instances from the SNAP [56] data set. We took 15 large directed graphs having between 7 115 and 2 394 385 nodes. In particular, we used web graphs, social networks, wikipedia graphs as well as purchase networks and autonomous system graphs. These instances were used in the heuristic track only.

Generated Hard Instances for Heuristic Solvers. In Figure 2, we show a graph where heuristics based on random walks choose a node that is not in the optimal DFVS with high probability. The optimal solution is to remove v_2 and v_3. However, v_1 has the highest probability in the stationary distribution ($\pi_1 = 0.28$) and, if removed, leads to a DFVS of size 3. We create larger instances by replicating this five-node graph N times (optimal DFVS has size $2N$). We additionally connect the N copies with directed edges (10 edges per copy) such that the size of the optimal DFVS does not change and the probabilities in the stationary distribution of visiting v_2 and v_3 in each copy do not increase. Furthermore, we generate a random graph $G_R = (V_R, E_R)$ with $|V_R| \in [2.5N, 5N]$ nodes (chosen uniformly and at random) and an average degree of 5. We then connect each node $u \in V_R$ to a random node representing v_1 in one of the copies and connect the nodes $\{v_2, v_3, v_4, v_5\}$ of each copy to a random node $v \in V_R$. This hides the internal structure of the graph and adds some noise to the size of the DFVS. We generate 25 instances of this graph with $N \in [10^2, 10^5]$. In our experiments, the size of the DFVS computed by our random walk algorithm was in most cases 1.5 times larger than the size of the optimal DFVS. The instances were used in the heuristic track only.

4 Participants and Results

There were 13 and 17 teams that officially submitted a solution to the exact and heuristic tracks, respectively. Several teams participated in more than one track; in total there were 26 distinct teams. The participants represented 3 continents and the following 12 countries (number of authors from the respective country is given in brackets): Germany (12), China (12), Czechia (10), France (7), Austria (5), India (5), Portugal (5), Norway (3), Romania (2), United States (2), Netherlands (1), Poland (1). The results are listed below.

4.1 Exact Track

The ranking for the exact track is listed subsequently; We list the number of solved instances from the 200 overall instances.

- **Rank 1** goes to solver raki123 having solved a total number of 185 instances; **Authors:** Andre Schidler and Rafael Kiesel; **Affiliation:** TU Wien; **URL** of solver: `https://github.com/ASchidler/dfvs`. **Zenodo:** [48]
- **Rank 2** goes to solver grapa-java having solved a total number of 165 instances; **Authors:** Enna Gerhard, Jona Dirks, Moritz Bergenthal, Jakob Gahde, Thorben Freese, Mario Grobler and Sebastian Siebertz; **Affiliation:** University of Bremen; **URL** of solver: `https://gitlab.informatik.uni-bremen.de/grapa/java/`. **Zenodo:** [9]
- **Rank 3** goes to solver mt-doom having solved a total number of 152 instances; **Authors:** Sebastian Angrich, Ben Bals, Niko Hastrich, Theresa Hradilak, Otto Kissig, Jonas Schmidt, Leo Wendt, Katrin Casel, Sarel Cohen and Davis Issac; **Affiliation:** Hasso Plattner Institute; **URL** of solver: `https://github.com/BenBals/mount-doom/tree/exact`. **Zenodo:** [4]
- **Rank 4** goes to solver goat_exact having solved a total number of 151 instances; **Authors:** Radovan Červený, Michal Dvořák, Xuan Thang Nguyen, Jan Pokorný, Lucie Procházková, Jaroslav Urban, Václav Blažej, Dušan Knop, Šimon Schierreich and Ondrej Suchy; **Affiliation:** Czech Technical University in Prague, Faculty of Information Technology; **URL** of solver: `https://gitlab.fit.cvut.cz/pace-challenge/2022/goat/exact`. **Zenodo:** [80]
- **Rank 5** goes to solver THS_exact having solved a total number of 140 instances; **Authors:** Henri Froese, Jonathan Guthermuth, Lars Huth, Marius Lotz, Johannes Meintrup, Timo Mertin, Manuel Penschuck and Hung Tran; **Affiliation:** Goethe University Frankfurt and THM, University of Applied Sciences Mittelhessen; **URL** of solver: `https://github.com/goethe-tcs/breaking-the-cycle`. **Zenodo:** [19]
- **Rank 6** goes to solver mndmky having solved a total number of 130 instances; **Authors:** Timon Behr; **Affiliation:** University of Konstanz; **URL** of solver: `https://github.com/mndmnky/duck-and-cover`. **Zenodo:** [8]
- **Rank 7** goes to solver DUM having solved a total number of 125 instances; **Authors:** Henri Dickel, Matija Miskovic and Lennart Uhrmacher; **Affiliation:** Philipps-Universität Marburg; **URL** of solver: `https://github.com/HenriDickel/DFVS-Solver/tree/PACE`. **Zenodo:** [22]

- **Rank 8** goes to solver yos having solved a total number of 120 instances; **Authors:** Yosuke Mizutani; **Affiliation:** University of Utah; **URL** of solver: `https://github.com/mogproject/dfvs-2022`. **Zenodo:** [64]
- **Rank 9** goes to solver rubengoetz having solved a total number of 88 instances; **Authors:** Ruben Götz; **Affiliation:** Karlsruher Institut für Technologie; **URL** of solver: `https://gitlab.com/rubenGoetz/dfvs-algo`. **Zenodo:** [39]
- **Rank 10** goes to solver DRIP having solved a total number of 32 instances; **Authors:** Aman Jain, Sachin Agarwal, Nimish Agrawal, Soumyajit Karmakar and Srinibas Swain; **Affiliation:** IIIT, Guwahati; **URL** of solver: `https://zenodo.org/record/6618812`. **Zenodo:** [43]

The following teams submitted a solver, but as described in the rules, their submissions were disqualified because of at least one suboptimal solution. Afterwards the teams sent us an updated version of their solver which then computed only correct results in the challenge. The number of solved instances reported below.

1. Solver Timeroot has solved a total number of 175 instances; **Authors:** Alexander Meiburg; **Affiliation:** UC Santa Barbara; **URL** of solver: `https://github.com/Timeroot/DVFS_PACE2022/tree/pace-2022`. **Zenodo:** [63], **ArXiv:** [62]
2. Solver swats has solved a total number of 160 instances; **Authors:** Sylwester Swat; **Affiliation:** Poznań University Of Technology; **URL** of solver: `https://github.com/swacisko/pace-2022`. **Zenodo:** [72]
3. Solver satanja has solved a total number of 144 instances; **Authors:** Stefan Tanja; **Affiliation:** Eindhoven University of Technology; **URL** of solver: `https://github.com/satanja/Hex`. **Zenodo:** [73]

Strategies Used in the Submissions

Winning Team. The approach by Andre Schidler and Rafael Kiesel [48] from TU Wien, Austria, applies a wide range of preprocessing techniques. These techniques stem a) from well-known reduction rules as well as b) non-trivial adaptations of reduction rules originally designed for the vertex cover problem. On the reduced instance, the team runs a MaxSAT solver that incrementally adds constraints.

Runner-Up. The approach by Enna Gerhard, Jona Dirks, Moritz Bergenthal, Jakob Gahde, Thorben Freese, Mario Grobler and Sebastian Siebertz from University of Bremen also applies a wide range of reduction rules to first decrease the size of the input. The team uses known as well as new reduction rules. Depending on the number of remaining undirected edges (forward and backward edges are present), the authors employ different strategies: a hitting set ILP formulation, an ILP formulation that models the problem as finding a topological order and if the later does not terminate within a specific time limit, the team runs a vertex cover solver to tackle the problem. If the solution of the last solver does not return a solution for the feedback vertex set problem, then no solution is returned.

Third Place. The team that achieved the third place are Sebastian Angrich, Ben Bals, Niko Hastrich, Theresa Hradilak, Otto Kissig, Jonas Schmidt, Leo Wendt, Katrin Casel, Sarel Cohen and Davis Issac from the Hasso Plattner Institute. As the first and second place, the team applies reduction rules first to reduce the input size. The team solves the remaining instance by repeatedly solving vertex cover instances. These instances are further reduced by the reductions of the PACE 2019 winning solver [41] and afterwards the instance is solved using a SAT-and-Reduce solver for the vertex cover problem [67].

4.2 Heuristic Track

The ranking for the heuristic track is listed subsequently; We list the score for the 200 overall instances. The score has been computed as outline in Section 3.1. Larger is better.

1. **Rank 1** goes to the solver swats with a score of 99.912; **Authors**: Sylwester Swat; **Affiliation**: Poznań University Of Technology; **URL** of solver: `https://github.com/swacisko/pace-2022`. **Zenodo:** [72]
2. **Rank 2** goes to the solver Nanored with a score of 99.911; **Authors**: Gabriel Bathie, Gaétan Berthe, Yoann Coudert-Osmont, David Desobry, Amadeus Reinald and Mathis Rocton; **Affiliation**: École normale supérieure de Lyon and Université de Lorraine, CNRS, Inria, LORIA; **URL** of solver: `https://github.com/Nanored4498/DreyFVS`. **Zenodo:** [31]
3. **Rank 3** goes to the solver hust_huawei with a score of 99.852; **Authors**: Yuming Du, Qingyun Zhang, Junzhou Xu, Shungen Zhang, Chao Liao, Zhihuai Chen, Zhibo Sun, Zhouxing Su, Junwen Ding, Chen Wu, Pinyan Lu and Zhipeng Lv; **Affiliation**: SMART, School of Computer Science and Technology, Huazhong University of Science & Technology and Huawei TCS Lab Shanghai; **URL** of solver: `https://github.com/1774150545/PACE-2022`. **Zenodo:** [77]
4. **Rank 4** goes to the solver KennethLangedal with a score of 99.832; **Authors**: Kenneth Langedal, Johannes Langguth and Fredrik Manne; **Affiliation**: University of Bergen and Simula Research Laboratory; **URL** of solver: `https://github.com/KennethLangedal/DFVS`. **Zenodo:** [53]
5. **Rank 5** goes to the solver fedrer with a score of 99.611; **Authors**: Aman Jain, Sachin Agarwal, Nimish Agrawal, Soumyajit Karmakar and Srinibas Swain; **Affiliation**: IIIT, Guwahati; **URL** of solver: `https://zenodo.org/record/6618777`. **Zenodo:** [44]
6. **Rank 6** goes to the solver Florian with a score of 99.435; **Authors**: Florian Sikora; **Affiliation**: LAMSADE; **URL** of solver: `https://github.com/fsikora/pace22`. **Zenodo:** [28]
7. **Rank 7** goes to the solver _UAIC_ANDREIARHIRE_ with a score of 99.156; **Authors**: Andrei Arhire and Paul Diac; **Affiliation**: Alexandru Ioan Cuza University of Iași; **URL** of solver: `https://github.com/AndreiiArhire/PACE2022`. **Zenodo:** [6]
8. **Rank 8** goes to the solver INESCIDteam with a score of 98.619; **Authors**: Daniel Castro, Luis Russo, Aleksandar Ilic, Paolo Romano and Ana Correia; **Affiliation**: INESC-ID & IST; **URL** of solver: `https://github.com/Daniel1993/pace-2022`. **Zenodo:** [16]
9. **Rank 9** goes to the solver goat_heuristic with a score of 98.278; **Authors**: Radovan Červený, Michal Dvořák, Xuan Thang Nguyen, Jan Pokorný, Lucie Procházková, Jaroslav Urban, Václav Blažej, Dušan Knop, Šimon Schierreich and Ondrej Suchy; **Affiliation**: Czech Technical University in Prague, Faculty of Information Technology; **URL** of solver: `https://gitlab.fit.cvut.cz/pace-challenge/2022/goat/heuristic`. **Zenodo:** [81]
10. **Rank 10** goes to the solver orodruin with a score of 98.245; **Authors**: Sebastian Angrich, Ben Bals, Niko Hastrich, Theresa Hradilak, Otto Kißig, Jonas Schmidt, Leo Wendt, Katrin Casel, Sarel Cohen and Davis Issac; **Affiliation**: Hasso Plattner Institute, Potsdam, Germany and Digital Engineering Faculty, University of Potsdam, Potsdam, Germany; **URL** of solver: `https://github.com/BenBals/mount-doom/tree/heuristic`. **Zenodo:** [5]

11. **Rank 11** goes to the solver THS_heuristic with a score of 95.357; **Authors**: Jonathan Guthermuth, Lars Huth, Marius Lotz, Johannes Meintrup, Timo Mertin, Manuel Penschuck, Lukas Schwarz and Hung Tran; **Affiliation**: Goethe University Frankfurt and THM, University of Applied Sciences Mittelhessen; **URL** of solver: `https://github.com/goethe-tcs/breaking-the-cycle`. **Zenodo:** [19]
12. **Rank 12** goes to the solver grapa-rust with a score of 94.744; **Authors**: Ozan Can Heydt, Leon Stichternath, Kenneth Dietrich and Philipp Haker; **Affiliation**: Universität Bremen; **URL** of solver: `https://gitlab.informatik.uni-bremen.de/grapa/rust/mimung`. **Zenodo:** [71]
13. **Rank 13** goes to the solver dfvsp-julia with a score of 92.644; **Authors**: Maria Bresich, Günther Raidl and Johannes Varga; **Affiliation**: TU Wien; **URL** of solver: `https://github.com/NunuNoName/dfvsp-solver`. **Zenodo:** [13]
14. **Rank 14** goes to the solver grapa-java with a score of 91.369; **Authors**: Enna Gerhard, Jona Dirks, Moritz Bergenthal, Jakob Gahde, Thorben Frese, Mario Grobler and Sebastian Siebertz; **Affiliation**: Universität Bremen; **URL** of solver: `https://gitlab.informatik.uni-bremen.de/grapa/java/`. **Zenodo:** [9]
15. **Rank 15** goes to the solver BreakingCycles with a score of 74.613; **Authors**: Mert Biyikli; **Affiliation**: Heidelberg University; **URL** of solver: `https://github.com/MertBiyikli/BreakingCycles.git`. **Zenodo:** [11]

There were two more submissions from the team of the hust_huawei solver (Rank 3), however, only their best solver (hust_huawei) was ranked:

1. Solver xjz_huawei with a score of 99.651; **Authors**: Yuming Du, Qingyun Zhang, Junzhou Xu, Shungen Zhang, Chao Liao, Zhihuai Chen, Zhibo Sun, Zhouxing Su, Junwen Ding, Chen Wu, Pinyan Lu and Zhipeng Lv; **Affiliation**: SMART, School of Computer Science and Technology, Huazhong University of Science & Technology and Huawei TCS Lab Shanghai; **URL** of solver: `https://github.com/xuxu9110/PACE2022.git`. **Zenodo:** [23]
2. Solver adu with a score of 99.618; **Authors**: Yuming Du, Qingyun Zhang, Junzhou Xu, Shungen Zhang, Chao Liao, Zhihuai Chen, Zhibo Sun, Zhouxing Su, Junwen Ding, Chen Wu, Pinyan Lu and Zhipeng Lv; **Affiliation**: SMART, School of Computer Science and Technology, Huazhong University of Science & Technology and Huawei TCS Lab Shanghai; **URL** of solver: `https://github.com/Zhang-qingyun/pace_2022_HUST_solver.git`. **Zenodo:** [78]

Strategies Used in the Submissions

Winning Team. The winning solver was due to Sylwester Swat from Poznań University Of Technology. The solver reduces the input using data reduction rules. It then finds some initial solution of the reduced graph using fast heuristics. It then tries to improve the found solution by using a variety of heuristic approaches. In the end the solution is transferred to the input instance. More specifically, if the input instance is somewhat close to a bidirected graph, then fast vertex cover solvers NuMVC [15] and FastVC [14] are used to compute an initial solution. Another heuristic employed is based on agent flows. Specifically, each node is assigned a fixed number of tokens. Then the algorithm proceeds in rounds. In each round, each token assigned to a node is moved to a random out-neighbor. When all rounds are finished, the node having most tokens is added to the feedback vertex set and the process is repeated until the obtained graph is acyclic.

Runner-Up. The team scoring the second rank consists of Gabriel Bathie, Gaétan Berthe, Yoann Coudert-Osmont, David Desobry, Amadeus Reinald and Mathis Rocton from École normale supérieure de Lyon and Université de Lorraine CNRS, Inria, LORIA. After performing data reductions, their algorithm first performs a guess on the reduced instance by leveraging the Sinkhorn-Knopp algorithm. The solution is then improved by pipelining two local search methods. The first local search algorithm is a vertex swapping algorithms, i.e. the algorithms removes a vertex from the current solution and if this creates a cycle it adds a random vertex of the current cycle to the solution. If removing the vertex does not create a cycle, then the solution size has been improved by one. The second local search algorithm uses the fact that a digraph is acyclic if and only if a topological ordering can be computed. The team then shows that a unique feedback vertex set can be created from any topological ordering and thus obtain a local search method by shuffling vertices in the topological ordering.

Third Place. The team Yuming Du, Qingyun Zhang, Junzhou Xu, Shungen Zhang, Chao Liao, Zhihuai Chen, Zhibo Sun, Zhouxing Su, Junwen Ding, Chen Wu, Pinyan Lu and Zhipeng Lv from SMART, School of Computer Science and Technology, Huazhong University of Science & Technology as well as Huawei TCS Lab Shanghai scored the third place. As the other solvers, data reduction rules are applied to first reduce the size of the instance. Afterwards, the authors use a simulated annealing algorithm. To obtain an initial solution the authors first transform the problem into a vertex cover problem and then solve it using a heuristic for this problem. This is done using a time constraint. The time constraint depends on the number of bidirectional edges, i.e. the larger the fraction of bidirectional edges, the more time is assigned to the vertex cover heuristic. The local search used to improve the solution is based on topological orderings of the graph [32].

5 PACE Organization

The program committee of PACE 2022 consisted of Ernestine Großmann, Tobias Heuer, Christian Schulz (chair) and Darren Strash. During the organization of PACE 2022 the Steering Committee was as follows:

- (since 2016) Holger Dell (Goethe University Frankfurt and IT University of Copenhagen)
- (since 2019) Johannes Fichte (Technische Universität Dresden)
- (since 2019) Markus Hecher (Technische Universität Wien)
- (since 2016) Bart M. P. Jansen (chair) (Eindhoven University of Technology)
- (since 2020) Łukasz Kowalik (University of Warsaw)
- (since 2021) André Nichterlein (Technical University of Berlin)
- (since 2020) Marcin Pilipczuk (University of Warsaw)
- (since 2020) Manuel Sorge (Technische Universität Wien)

6 Conclusion and Future Editions of PACE

We thank all the participants for their enthusiasm, strong and interesting contributions. Special thanks go to the participants who also presented at IPEC 2022. We are very happy that this edition attracted many people as well as strong contributions and hope that this will continue for future editions by considering popular problems to the community or even by repeating previously posted problems. As in previous challenges, we provided the public and private instance set in a public data library[2].

[2] `https://github.com/PACE-challenge/pacechallenge.org/tree/master/files`

We welcome anyone who is interested to add their name to the mailing list on the PACE website to receive updates and join the discussion. We look forward to the next edition. Detailed information will be posted on the website at pacechallenge.org. Also see the Twitter account[3]. In particular, plans for PACE 2023 will be posted there.

References

1 Networks project, 2017. URL: http://www.thenetworkcenter.nl.

2 Faisal N. Abu-Khzam, Sebastian Lamm, Matthias Mnich, Alexander Noe, Christian Schulz, and Darren Strash. Recent advances in practical data reduction. *Special Issue of SPP Big Data*, 2022. arXiv:2012.12594.

3 N. Faisal Abu-Khzam, R. Michael Fellows, A. Michael Langston, and Henry W. Suters. Crown structures for vertex cover kernelization. *Theory Comput. Syst.*, 41(3):411–430, 2007. doi:10.1007/s00224-007-1328-0.

4 Sebastian Angrick, Ben Bals, Katrin Casel, Sarel Cohen, Niko Hastrich, Theresa Hradilak, Davis Issac, Otto Kißig, Jonas Schmidt, and Leo Wendt. Mount Doom – An Exact Solver for Directed Feedback Vertex Set, June 2022. doi:10.5281/zenodo.6645235.

5 Sebastian Angrick, Ben Bals, Katrin Casel, Sarel Cohen, Niko Hastrich, Theresa Hradilak, Davis Issac, Otto Kißig, Jonas Schmidt, and Leo Wendt. Orodruin — A Heuristic Solver for Directed Feedback Vertex Set, June 2022. doi:10.5281/zenodo.6645245.

6 Andrei Arhire and Paul Diac. _UAIC_ANDREIARHIRE_ – A Heuristic Solver for the Directed Feedback Vertex Set Problem, June 2022. doi:10.5281/zenodo.6646187.

7 Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999. doi:10.1126/science.286.5439.509.

8 Timon Behr. Direfever, June 2022. doi:10.5281/zenodo.6651761.

9 Moritz Bergenthal, Jona Dirks, Thorben Freese, Jakob Gahde, and Enna Gerhard. GraPA-JAVA, June 2022. doi:10.5281/zenodo.6647003.

10 Benjamin Bergougnoux, Eduard Eiben, Robert Ganian, Sebastian Ordyniak, and M. S. Ramanujan. Towards a polynomial kernel for directed feedback vertex set. *Algorithmica*, 83(5):1201–1221, 2021. doi:10.1007/s00453-020-00777-5.

11 Mert Biyikli. MertBiyikli/BreakingCycles: Fourth release, June 2022. doi:10.5281/zenodo.6674065.

12 Édouard Bonnet and Florian Sikora. The PACE 2018 parameterized algorithms and computational experiments challenge: The third iteration. In Christophe Paul and Michal Pilipczuk, editors, *13th International Symposium on Parameterized and Exact Computation, IPEC 2018, August 20-24, 2018, Helsinki, Finland*, volume 115 of *LIPIcs*, pages 26:1–26:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.IPEC.2018.26.

13 Maria Bresich, Günther Raidl, and Johannes Varga. HyMeHeu-solver – A Hybrid Metaheuristic Solver for the Directed Feedback Vertex Set Problem, June 2022. doi:10.5281/zenodo.6643236.

14 Shaowei Cai, Jinkun Lin, and Chuan Luo. Finding a small vertex cover in massive sparse graphs: Construct, local search, and preprocess. *J. Artif. Intell. Res.*, 59:463–494, 2017. doi:10.1613/jair.5443.

15 Shaowei Cai, Kaile Su, Chuan Luo, and Abdul Sattar. NuMVC: An efficient local search algorithm for minimum vertex cover. *J. Artif. Intell. Res.*, 46:687–716, 2013. doi:10.1613/jair.3907.

16 Daniel Castro. Daniel1993/pace-2022: pace-2022, June 2022. doi:10.5281/zenodo.6634725.

17 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(5):21:1–21:19, 2008. doi:10.1145/1411509.1411511.

[3] https://twitter.com/pace_challenge

18 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

19 Holger Dell, Henri Froese, Lukas Geis, Jonathan Guthermuth, Anselm Haak, Lars Huth, Frank Kammer, Marius Lotz, Johannes Meintrup, Timo Mertin, Manuel Penschuck, and Lukas Schwarz. BreakingTheCycle, June 2022. This work was partially funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under grants 379157101, ME 2088/5-1 (FOR 2975 — Algorithms, Dynamics, and Information Flow in Networks). doi:10.5281/zenodo.6602946.

20 Holger Dell, Thore Husfeldt, Bart M. P. Jansen, Petteri Kaski, Christian Komusiewicz, and Frances A. Rosamond. The first parameterized algorithms and computational experiments challenge. In Jiong Guo and Danny Hermelin, editors, *11th International Symposium on Parameterized and Exact Computation, IPEC 2016, August 24-26, 2016, Aarhus, Denmark*, volume 63 of *LIPIcs*, pages 30:1–30:9. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.IPEC.2016.30.

21 Holger Dell, Christian Komusiewicz, Nimrod Talmon, and Mathias Weller. The PACE 2017 parameterized algorithms and computational experiments challenge: The second iteration. In Daniel Lokshtanov and Naomi Nishimura, editors, *12th International Symposium on Parameterized and Exact Computation, IPEC 2017, September 6-8, 2017, Vienna, Austria*, volume 89 of *LIPIcs*, pages 30:1–30:12. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.IPEC.2017.30.

22 Dickel, Miskovic, and Uhrmacher. pace-2022-dum, May 2022. doi:10.5281/zenodo.6599645.

23 Yuming Du, Qingyun Zhang, Junzhou Xu, Shungen Zhang, Chao Liao, Zhihuai Chen, Zhibo Sun, Zhouxing Su, Junwen Ding, Chen Wu, Pinyan Lu, and Zhipeng Lv. Huawei_tcs_dfvs_solver, June 2022. doi:10.5281/zenodo.6638370.

24 M. Ayaz Dzulfikar, Johannes Klaus Fichte, and Markus Hecher. The PACE 2019 parameterized algorithms and computational experiments challenge: The fourth iteration (invited paper). In Bart M. P. Jansen and Jan Arne Telle, editors, *14th International Symposium on Parameterized and Exact Computation, IPEC 2019, September 11-13, 2019, Munich, Germany*, volume 148 of *LIPIcs*, pages 25:1–25:23. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.IPEC.2019.25.

25 John D Eblen, Charles A Phillips, Gary L Rogers, and Michael A Langston. The maximum clique enumeration problem: algorithms, applications, and implementations. In *BMC Bioinformatics*, volume 13, page S5, 2012. doi:10.1186/1471-2105-13-S10-S5.

26 Paul Erdős and Alfréd Rényi. On Random Graphs I. *Publicationes Mathematicae (Debrecen)*, 6:290–297, 1959 1959.

27 Guy Even, Joseph Naor, Baruch Schieber, and Madhu Sudan. Approximating minimum feedback sets and multicuts in directed graphs. *Algorithmica*, 20(2):151–174, 1998. doi:10.1007/PL00009191.

28 Florian. fsikora/pace22: First version for pace, June 2022. doi:10.5281/zenodo.6624196.

29 Fedor V. Fomin, Serge Gaspers, Daniel Lokshtanov, and Saket Saurabh. Exact algorithms via monotone local search. *J. ACM*, 66(2), March 2019. doi:10.1145/3284176.

30 Daniel Funke, Sebastian Lamm, Ulrich Meyer, Manuel Penschuck, Peter Sanders, Christian Schulz, Darren Strash, and Moritz von Looz. Communication-free massively distributed graph generation. *J. Parallel Distributed Comput.*, 131:200–217, 2019. doi:10.1016/j.jpdc.2019.03.011.

31 Bathie Gabriel, Berthe Gaétan, Coudert-Osmont Yoann, Desobry David, Reinald Amadeus, and Rocton Mathis. Dreyfvs, June 2022. doi:10.5281/zenodo.6638217.

32 Philippe Galinier, Eunice Adjarath Lemamou, and Mohamed Wassim Bouzidi. Applying local search to the feedback vertex set problem. *J. Heuristics*, 19(5):797–818, 2013. doi:10.1007/s10732-013-9224-z.

33 Georges Gardarin and Stefano Spaccapietra. Integrity of data bases: A general lockout algorithm with deadlock avoidance. In G. M. Nijssen, editor, *Modelling in Data Base Management Systems, Proceeding of the IFIP Working Conference on Modelling in Data Base Management Systems, Freudenstadt, Germany, January 5-8, 1976*, pages 395–412. North-Holland, 1976.

34 Serge Gaspers and Matthias Mnich. Feedback vertex sets in tournaments. *J. Graph Theory*, 72(1):72–89, 2013. doi:10.1002/jgt.21631.

35 E. N. Gilbert. Random graphs. *Ann. Math. Statist.*, 30(4):1141–1144, December 1959. doi:10.1214/aoms/1177706098.

36 Luca Gugelmann, Konstantinos Panagiotou, and Ueli Peter. Random hyperbolic graphs: Degree sequence and clustering. In *Automata, Languages, and Programming – 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part II*, volume 7392 of *Lecture Notes in Computer Science*, pages 573–585. Springer, 2012. doi:10.1007/978-3-642-31585-5_51.

37 Venkatesan Guruswami, Johan HÅstad, Rajsekar Manokaran, Prasad Raghavendra, and Moses Charikar. Beating the random ordering is hard: Every ordering CSP is approximation resistant. *SIAM Journal on Computing*, 40(3):878–914, 2011. doi:10.1137/090756144.

38 Venkatesan Guruswami and Euiwoong Lee. Simple proof of hardness of feedback vertex set. *Theory Comput.*, 12(1):1–11, 2016. doi:10.4086/toc.2016.v012a006.

39 Ruben Götz. Ruben Bachelor Thesis, June 2022. doi:10.5281/zenodo.6604728.

40 Monika Henzinger, Alexander Noe, Christian Schulz, and Darren Strash. Finding all global minimum cuts in practice. In *Proc. ESA 2020*, volume 173 of *Leibniz Int. Proc. Informatics*, pages 59:1–59:20, 2020. doi:10.4230/LIPIcs.ESA.2020.59.

41 Demian Hespe, Sebastian Lamm, Christian Schulz, and Darren Strash. WeGotYouCovered: The winning solver from the PACE 2019 challenge, vertex cover track. In H. Martin Bücker, Xiaoye Sherry Li, and Sivasankaran Rajamanickam, editors, *Proceedings of the SIAM Workshop on Combinatorial Scientific Computing, CSC 2020, Seattle, USA, February 11-13, 2020*, pages 1–11. SIAM, 2020. doi:10.1137/1.9781611976229.1.

42 A. Itai, A. G. Konheim, and M. Rodeh. A sparse table implementation of priority queues. In S. Even and O. Kariv, editors, *Proceedings of the 8th Colloquium on Automata, Languages and Programming*, volume 115 of *LNCS*, pages 417–431. Springer, 1981. doi:10.1007/3-540-10843-2_34.

43 Aman Jain, Sachin Agarwal, Nimish Agrawal, Soumyajit Karmakar, and Srinibas Swain. DRIP: Directed feedback vertex set computation using Reductions and Integer Programming, June 2022. doi:10.5281/zenodo.6618812.

44 Aman Jain, Sachin Agarwal, Nimish Agrawal, Soumyajit Karmakar, and Srinibas Swain. FEDRER: Feedback vertex set using Edge Density and REmove Redundant, June 2022. doi:10.5281/zenodo.6618777.

45 Arthur B. Kahn. Topological Sorting of Large Networks. *Commun. ACM*, 5(11):558–562, 1962. doi:10.1145/368996.369025.

46 Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher, editors, *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972. doi:10.1007/978-1-4684-2001-2_9.

47 Leon Kellerhals, Tomohiro Koana, André Nichterlein, and Philipp Zschoche. The PACE 2021 parameterized algorithms and computational experiments challenge: Cluster editing. In Petr A. Golovach and Meirav Zehavi, editors, *16th International Symposium on Parameterized and Exact Computation, IPEC 2021, September 8-10, 2021, Lisbon, Portugal*, volume 214 of *LIPIcs*, pages 26:1–26:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.IPEC.2021.26.

48 Rafael Kiesel and Andre Schidler. DAGger – An Exact Directed Feedback Vertex Set Solver, June 2022. doi:10.5281/zenodo.6627405.

49 Tomasz Kociumaka and Marcin Pilipczuk. Faster deterministic feedback vertex set. *Information Processing Letters*, 114(10):556–560, 2014. doi:10.1016/j.ipl.2014.05.001.

50 Viatcheslav Korenwein, André Nichterlein, Rolf Niedermeier, and Philipp Zschoche. Data reduction for maximum matching on real-world graphs: Theory and experiments. In *Proc. ESA 2018*, volume 112 of *Leibniz Int. Proc. Informatics*, pages 53:1–53:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.ESA.2018.53.

51 Lukasz Kowalik, Marcin Mucha, Wojciech Nadara, Marcin Pilipczuk, Manuel Sorge, and Piotr Wygocki. The PACE 2020 parameterized algorithms and computational experiments challenge: Treedepth. In Yixin Cao and Marcin Pilipczuk, editors, *15th International Symposium on Parameterized and Exact Computation, IPEC 2020, December 14-18, 2020, Hong Kong, China (Virtual Conference)*, volume 180 of *LIPIcs*, pages 37:1–37:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.IPEC.2020.37.

52 Dmitri V. Krioukov, Fragkiskos Papadopoulos, Maksim Kitsak, Amin Vahdat, and Marián Boguñá. Hyperbolic geometry of complex networks. *CoRR*, abs/1006.5169, 2010. arXiv:1006.5169.

53 Kenneth Langedal. KennethLangedal/DFVS: pace-2022, June 2022. doi:10.5281/zenodo.6630611.

54 Charles E. Leiserson and James B. Saxe. Retiming synchronous circuitry. *Algorithmica*, 6(1):5–35, 1991. doi:10.1007/BF01759032.

55 Mile Lemaic. *Markov-Chain-Based Heuristics for the Feedback Vertex Set Problem for Digraph*. PhD thesis, University of Cologne, 2008. URL: http://kups.ub.uni-koeln.de/id/eprint/2547.

56 Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.

57 Orna Lichtenstein and Amir Pnueli. Checking that finite state concurrent programs satisfy their linear specification. In Mary S. Van Deusen, Zvi Galil, and Brian K. Reid, editors, *Conference Record of the Twelfth Annual ACM Symposium on Principles of Programming Languages, New Orleans, Louisiana, USA, January 1985*, pages 97–107. ACM Press, 1985. doi:10.1145/318593.318622.

58 Hen-Ming Lin and Jing-Yang Jou. On Computing the Minimum Feedback Vertex Set of a Directed Graph by Contraction Operations. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, 19(3):295–307, 2000. doi:10.1109/43.833199.

59 Daniel Lokshtanov, Pranabendu Misra, Joydeep Mukherjee, Fahad Panolan, Geevarghese Philip, and Saket Saurabh. 2-approximating feedback vertex set in tournaments. *ACM Trans. Algorithms*, 17(2), April 2021. doi:10.1145/3446969.

60 Daniel Lokshtanov, M. S. Ramanuajn, and Saket Saurabh. When recursion is better than iteration: A linear-time algorithm for acyclicity with few error vertices. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '18, pages 1916–1933, USA, 2018. Society for Industrial and Applied Mathematics. doi:10.1137/1.9781611975031.125.

61 Daniel Lokshtanov, M. S. Ramanujan, Saket Saurabh, Roohani Sharma, and Meirav Zehavi. Wannabe bounded treewidth graphs admit a polynomial kernel for DFVS. In Zachary Friggstad, Jörg-Rüdiger Sack, and Mohammad R. Salavatipour, editors, *Algorithms and Data Structures – 16th International Symposium, WADS 2019, Edmonton, AB, Canada, August 5-7, 2019, Proceedings*, volume 11646 of *Lecture Notes in Computer Science*, pages 523–537. Springer, 2019. doi:10.1007/978-3-030-24766-9_38.

62 Alex Meiburg. Reduction Rules and ILP Are All You Need: Minimal Directed Feedback Vertex Set, 2022. doi:10.48550/ARXIV.2208.01119.

63 Alexander Meiburg. PACE 2022 – DFVS-Via-Flattening Solver (DVFS), June 2022. Derived from https://github.com/Timeroot/DVFS_PACE 2022/tree/pace-2022. doi:10.5281/zenodo.6650921.

64 Yosuke Mizutani. PACE 2022 – Exact, June 2022. doi:10.5281/zenodo.6604875.

65 Mathew Penrose. *Random geometric graphs.* Number 5 in Oxford Studies in Probability. Oxford University Press, 2003.

66 Manuel Penschuck, Ulrik Brandes, Michael Hamann, Sebastian Lamm, Ulrich Meyer, Ilya Safro, Peter Sanders, and Christian Schulz. Recent advances in scalable network generation. *CoRR*, abs/2003.00736, 2020. arXiv:2003.00736.

67 Rick Plachetta and Alexander van der Grinten. SAT-and-reduce for vertex cover: Accelerating branch-and-reduce by SAT solving. In Martin Farach-Colton and Sabine Storandt, editors, *Proceedings of the Symposium on Algorithm Engineering and Experiments, ALENEX 2021, Virtual Conference, January 10-11, 2021*, pages 169–180. SIAM, 2021. doi:10.1137/1.9781611976472.13.

68 U. N. Raghavan, R. Albert, and S. Kumara. Near Linear Time Algorithm to Detect Community Structures in Large-Scale Networks. *Physical Review E*, 76(3), 2007. doi:10.1103/PhysRevE.76.036106.

69 Igor Razgon. Computing minimum directed feedback vertex set in $O^*(1.9977^n)$. In Giuseppe F. Italiano, Eugenio Moggi, and Luigi Laura, editors, *Theoretical Computer Science, 10th Italian Conference, ICTCS 2007, Rome, Italy, October 3-5, 2007, Proceedings*, pages 70–81. World Scientific, 2007. doi:10.1142/9789812770998_0010.

70 Ewald Speckenmeyer. On Feedback Problems in Digraphs. In *Graph-Theoretic Concepts in Computer Science, 15th International Workshop, WG '89, Castle Rolduc, The Netherlands, June 14-16, 1989, Proceedings.* Springer, 1989. doi:10.1007/3-540-52292-1_16.

71 Leon Stichternath, Ozan Heydt, Kenneth Dietrich, and Philipp Haker. Grapa-Rust, June 2022. doi:10.5281/zenodo.6603799.

72 Sylwester Swat. swacisko/pace-2022: First release of DiVerSeS, a solver for the Directed Feedback Vertex Set problem, June 2022. doi:10.5281/zenodo.6643144.

73 Stefan Tanja. satanja/hex: pace-2022, June 2022. doi:10.5281/zenodo.6609797.

74 Robert Endre Tarjan. Depth-First Search and Linear Graph Algorithms. *SIAM J. Comput.*, 1(2):146–160, 1972. doi:10.1137/0201010.

75 Szymon Wasik, Maciej Antczak, Jan Badura, Artur Laskowski, and Tomasz Sternal. Optil.io: Cloud based platform for solving optimization problems using crowdsourcing approach. In *Proceedings of the 19th ACM Conference on Computer Supported Cooperative Work and Social Computing Companion*, CSCW '16 Companion, pages 433–436, New York, NY, USA, 2016. Association for Computing Machinery. doi:10.1145/2818052.2869098.

76 Mingyu Xiao and Hiroshi Nagamochi. An improved exact algorithm for undirected feedback vertex set. *J. Comb. Optim.*, 30(2):214–241, 2015. doi:10.1007/s10878-014-9737-x.

77 YuMingDu, QingYunZhang, and ShunGenZhang. pace-2022, June 2022. doi:10.5281/zenodo.6644409.

78 Zhang-qingyun. Zhang-qingyun/pace_2022_HUST_solver: pace_2022_HUST_solver, June 2022. doi:10.5281/zenodo.6643002.

79 Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical Report CMU-CALD-02-107, Carnegie Mellon University, 2002. URL: http://reports-archive.adm.cs.cmu.edu/anon/cald/CMU-CALD-02-107.pdf.

80 Radovan Červený, Michal Dvořák, Xuan Thang Nguyen, Jan Pokorný, Lucie Procházková, Jaroslav Urban, Václav Blažej, Dušan Knop, Šimon Schierreich, and Ondřej Suchý. G2OAT solver for PACE 2022 (DFVS) exact track, June 2022. doi:10.5281/zenodo.6637464.

81 Radovan Červený, Michal Dvořák, Xuan Thang Nguyen, Jan Pokorný, Lucie Procházková, Jaroslav Urban, Václav Blažej, Dušan Knop, Šimon Schierreich, and Ondřej Suchý. G2OAT solver for PACE 2022 (DFVS) heuristic track, June 2022. doi:10.5281/zenodo.6637495.