Honda Research Institute EU
Honda Research Institute Europe GmbH
https://www.honda-ri.de/

Mixed Integer Linear Programming Based Large Neighborhood Search Approaches for the Directed Feedback Vertex Set Problem

Maria Bresich, Johannes Varga, Guenther Raidl, Steffen Limmer

2024

Preprint:

This is a post-peer-review, pre-copyedit version of an article published in 9th International Conference on Metaheuristics and Nature Inspired Computing (META). The final authenticated version is available online at: https://doi.org/10.1007/978-3-031-69257-4_1

Mixed Integer Linear Programming Based Large Neighborhood Search Approaches for the Directed Feedback Vertex Set Problem*

Maria Bresich[1]✉[0009−0000−8291−6765], Johannes Varga[1][0000−0003−1413−7115], Günther R. Raidl[1][0000−0002−3293−177X], and Steffen Limmer[2][0000−0003−2385−7886]

[1] Institute of Logic and Computation, TU Wien, Vienna, Austria
{mbresich, jvarga, raidl}@ac.tuwien.ac.at
[2] Honda Research Institute Europe GmbH, 63073 Offenbach/Main, Germany
steffen.limmer@honda-ri.de

Abstract. A directed feedback vertex set (DFVS) of a directed graph is a subset of vertices whose removal makes the graph acyclic. Finding a DFVS of minimum cardinality is the goal of the directed feedback vertex set problem, an NP-hard combinatorial optimization problem. We first consider two mixed integer linear programming (MILP) models for this problem, which, when solved with Gurobi, are effective on graphs of small to medium complexity but do not scale well to large instances. Aiming at better scalability and higher robustness over a large variety of graphs, we investigate a large neighborhood search (LNS) in which a destroy operator removes randomly chosen nodes from an incumbent DFVS and one of the MILP models is used for repair. Regarding the destroy operator, finding a best degree of destruction is challenging. A main contribution lies in proposing several selection strategies for this parameter as well as a strategy for choosing the more promising MILP model for repair. We evaluate the performance of the MILP models and different LNS variants on benchmark instances and compare the approaches to each other as well as to state-of-the-art procedures. Results show that our LNS variants yield clearly better solutions on average than standalone MILP solving. Even though our approaches cannot outperform the state-of-the-art, we gain valuable insights on beneficially configuring such a MILP-based LNS.

Keywords: Large neighborhood search · Mathematical programming · Mixed integer linear programming · Directed feedback vertex set problem.

1 Introduction

The directed feedback vertex set problem (DFVSP) is a classical combinatorial optimization problem in which the goal is to remove from a given directed graph as few vertices as possible in order to make it acyclic. More specifically, a *directed*

* This project was financially supported by Honda Research Institute Europe GmbH.

feedback vertex set (DFVS) of a directed graph is a subset of vertices that contains at least one vertex of every simple cycle in the graph, and we are interested in a DFVS of smallest cardinality. The removal of such a DFVS from the graph in conjunction with the incident arcs results in a *directed acyclic graph* (DAG). This problem is one of the first problems shown to be NP-complete by Karp [15]. The DFVSP is a variant of the feedback vertex set problem (FVSP), which only considers undirected graphs. Exact algorithms for the (D)FVSP are limited in their practical applicability to instances of rather small and medium size due to the complexity of the problem. Research thus has also focused on approximation algorithms [2], parameterized algorithms [8,11], or graph classes for which efficient algorithms are possible [6,7]. Moreover, in the last decade, heuristics and especially (hybrid) metaheuristic procedures have also been proposed for addressing large (D)FVSP instances [9,13,18,26]. These approaches do not guarantee optimality but aim at providing good or near-optimal solutions in a reasonable time also for larger instances. Compared to exact methods, this often makes them more suitable for real-world applications, such as deadlock detection and recovery in operating systems, program verification, and VLSI chip design, where oftentimes a short runtime is more important than achieving optimality.

We follow this line of research and more specifically investigate diverse variants of large neighborhood search (LNS) in which partially destroyed solutions are repaired by means of mixed integer linear programming (MILP). Large neighborhood search [22,24] is a prominent metaheuristic that has already been successfully applied to a multitude of challenging combinatorial optimization problems including routing, scheduling, and location problems [14,22,24]. In our LNS, we apply a neighborhood structure that is based on destroying a current solution by moving multiple vertices from the current DFVS back into the corresponding DAG, thus introducing cycles again. For making the solutions feasible again, i.e., repairing them, we propose two MILP formulations for the DFVSP, which we solve with a leading general purpose MILP solver. These MILP formulations are also evaluated as standalone solving procedures, and we will see that they perform reasonably well for small to medium sized instances but can hardly be applied to large instances. The LNS framework therefore is a natural and promising choice to exploit the power of the MILP approaches in order to also solve large instances heuristically. Our main goal is to gain insights on beneficial configurations and on the performance of such a hybrid LNS. We investigate various strategies for selecting the degree of destruction for the LNS and also a technique for dynamically choosing the MILP formulation for each instance. The resulting variants are tested on three sets of benchmark instances and compared to each other as well as to two state-of-the-art solving approaches from the literature. This work is based on the first author's master's thesis [4].

In the next section, we review related work before formally defining the DFVSP and introducing the two MILP formulations in Section 3. In Section 4, our graph reduction procedure is explained and Section 5 deals with the construction heuristic and local search. Our proposed LNS variants are presented in detail in Section 6, and Section 7 discusses the computational study and experimental

results. Finally, we conclude this article and give an outlook on possible future work in Section 8.

2 Related Work

In the literature, various types of FVSPs are considered, which generally differ in two dimensions: graph orientation and involvement of weights. Oftentimes, the main concepts of solving approaches that were designed for one variant can be adapted and applied to the others. This is also true for the closely related feedback arc set (FAS) problem for directed graphs, whose goal is to find a subset of arcs of minimum cardinality that contains at least one arc of every cycle in the graph. This problem can be efficiently reduced to the DFVSP and vice versa. One approach to the FAS problem on unweighted graphs is given by Noughabi and Baghbani [19] who propose a genetic algorithm. They represent solutions as vertex orderings where all backward arcs constitute a FAS. This is possible due to the relationship of the FAS problem and the linear ordering problem because the latter can be reduced to the complement of the FAS problem, the maximum acyclic subgraph problem. Baharev et al. [1] also make use of this relationship and introduce an integer programming (IP) model for the FAS problem on edge-weighted graphs, which employs triangle inequalities to encode a minimum cost ordering of the vertices. The authors further propose two other IP models, which are both based on a minimum set cover formulation of the FAS problem, with the second model being an extension of the first one by adding lazy constraint generation.

For the undirected FVSP on vertex-weighted graphs, Cutello et al. [9] introduce a metaheuristic that combines an immune algorithm and a local search (LS) procedure and is specifically designed for large-scale instances. Melo et al. [18] tackle the same problem by solving its complement, which is the maximum weighted induced forest problem, and they introduce the first two compact MILP models for this problem. One formulation is flow-based and the other is based on an adaptation of a lifted version of the Miller-Tucker-Zemlin (MTZ) constraints. As both formulations are designed for connected and directed graphs, they are also applicable to the DFVS problem. Based on these MILP models, Melo et al. propose two variants of a hybrid metaheuristic that combines a multi-start iterated local search with a MILP-based local search procedure. In the next section, we will adapt the MTZ-based formulation to obtain one of our two MILP models for the unweighted DFVSP, which results in the first compact model for this problem to the best of our knowledge.

Considering heuristic approaches, there have not been many for the unweighted DFVSP until recently. The first ones were greedy heuristics and a greedy randomized adaptive search procedure (GRASP) by Pardalos et al. [20]. Galinier et al. [13] propose a local search procedure with an efficient neighborhood structure based on a new solution representation of the DFVS. Instead of minimizing the size of the DFVS, they focus on its complement and try to maximize the size of a contained DAG that is represented by a topological ordering. On

top of the LS, the authors describe a simulated annealing (SA) algorithm called the SA-FVSP. This approach is extended by Tang et al. [26] with nonuniform neighborhood sampling (NNS) resulting in an algorithm called SA-FVSP-NNS. In this approach, new priority and sampling functions are applied to guide the search. The priority function considers vertex degrees in a greedy manner similarly as in Cai et al. [5]. We will also utilize this concept in our procedures.

Recently in 2022, the development of new heuristic approaches for the unweighted DFVSP was sparked by the Parameterized Algorithms and Computational Experiments (PACE) challenge[1], which was one of the inspirations for our work. We also participated with an early version of our solving procedures, which, however, only ended up in the bottom third of the ranking. This paper describes improved variants, but as the challenge took place at the same time as the development of our solving approaches, we were not able to already learn from its results. Two of the three best performing heuristic solvers, one by Swat [25] and one by Du et al. [10], apply a transformation of the DFVSP into a vertex cover problem as well as diverse extensions of SA-FVSP. The solver by Bathie et al. [3] employs two greedy local search heuristics based on vertex swapping and perturbations of topological orderings.

What all of the finally leading solvers from the challenge have in common is the application of partly advanced graph reduction operations to decrease the size of the input graph and possibly partition it into unconnected components that can be independently addressed. Five popular reduction rules were already introduced in 1988 by Levy and Low [16]: IN0, OUT0, IN1, OUT1, and LOOP. Lin and Jou [17] proposed another three called PIE, CORE, and DOME. Six rules were also described by Fleischer et al. [11], but three of them were already covered by Levy and Low. An important property of all these operations is that they preserve optimal solutions. In their work on the FAS problem on unweighted graphs, Park and Akers [21] suggest four reduction operations, two of which also coincide with rules by Levy and Low.

3 Problem Formalization and MILP Models

The unweighted DFVSP we consider in this work is defined on a directed graph $G = (V, E)$ with vertex set V and arc set E. We only consider simple graphs that do not have parallel arcs or self-loops. The subgraph of G induced by a subset of vertices $V' \subseteq V$ is $G[V'] = (V', E')$ with $E' = E \cap (V' \times V')$ being the set of arcs from E whose endpoints are both contained in V'. A directed feedback vertex set (DFVS) is a subset $F \subseteq V$ of vertices that contains at least one vertex of every simple cycle in G. Let $\overline{F} = V \setminus F$ be the complement of F, then the induced subgraph $G[\,\overline{F}\,]$ is a DAG if and only if F is a valid DFVS of G. The goal of the DFVSP is to find a DFVS F^* of minimum cardinality, i.e., $|F^*| \leq |F|$ for every DFVS $F \subseteq V$. By these definitions, solving the DFVS problem is equivalent to finding a vertex set $\overline{F^*}$ of maximum cardinality such that the induced subgraph $G[\,\overline{F^*}\,]$ is acyclic.

[1] https://pacechallenge.org/

We introduce two mixed integer linear programming formulations for the DFVSP. The first model is inspired by the subtour elimination constraints from Miller, Tucker, and Zemlin and is therefore called the *MTZ-based formulation*. The other formulation uses so-called *cycle elimination constraints* (CECs), which prevent cycles in the DAG corresponding to the DFVS.

3.1 MTZ-Based Formulation

This formulation is derived from the MILP model by Melo et al. [18] for the vertex-weighted, undirected FVSP. In order to consider the aspect that we are dealing with a directed graph as input, we add an artificial *source vertex* s to V and arcs from s to all other vertices. Thus, graph $G = (V, E)$ is augmented to the new graph $G_s = (V_s, E_s)$ with $V_s = V \cup \{s\}$ and $E_s = E \cup \{(s, v) \,:\, v \in V\}$. Our MTZ-based formulation actually models the complement of the DFVSP in which the aim is to find a maximum cardinality subset $\overline{F} \subseteq V$ whose induced subgraph is acyclic. We use binary decision variables y_v, $v \in V$, indicating with value one that $v \in \overline{F}$, i.e., the vertex is selected for the DAG, and consequently with value zero that v is part of the DFVS. Moreover, continuous variables $\Phi_v \in [0, |V_s| - 1]$ represent a *potential* for each vertex $v \in V$. In a valid solution, the potentials have to increase with the distance of v to s. The MTZ-based formulation of the DFVS problem is then:

$$\min \quad |V| - \sum_{v \in V} y_v \tag{1}$$

$$\text{s.t.} \quad \Phi_u - \Phi_v + |V_s| \cdot y_v \leq |V_s| - 1 \qquad \forall (u, v) \in E_s \tag{2}$$

$$y_v \leq \Phi_v \qquad \forall v \in V \tag{3}$$

$$y_s = 1 \tag{4}$$

$$\Phi_s = 0 \tag{5}$$

$$y_v \in \{0, 1\} \qquad \forall v \in V \tag{6}$$

$$0 \leq \Phi_v \leq |V_s| - 1 \qquad \forall v \in V \tag{7}$$

The objective function (1) minimizes the cardinality of the DFVS, which corresponds to maximizing the number of vertices $v \in V$ with $y_v = 1$ constituting $\overline{F}$ and thus belonging to the DAG. Constraints (2) are the MTZ inequalities ensuring that for each arc (u, v) whose target vertex v is chosen for $\overline{F}$, the potential Φ_v of v must be larger by at least one than the potential Φ_u of vertex u. In this way, the potentials of the nodes on any path originating from s will always increase monotonically and cycles are prevented. The linear programming relaxation is strengthened by Inequalities (3), which explicitly enforce that the potential of selected vertices (except s) is at least one. Equations (4) and (5) define that the source node has to be connected and has potential zero. Lastly, (6) and (7) define the domains of the decision variables.

We remark that the original MILP formulation from Melo et al. [18] additionally uses binary variables for all arcs and the MTZ inequalities are based on these. We also considered such an approach in a corresponding model for the

DFVSP, but computational experiments with Gurobi indicated that the more compact MTZ-based model presented above performs significantly better; see the first author's thesis [4] for details. Therefore, we only consider this variant in the following.

3.2 CEC-Based Formulation

The CEC-based formulation builds on cycle elimination constraints, which explicitly state for each cycle in the input graph G that at least one of its nodes must be removed to avoid this cycle. The model uses the same binary variables y_v, $v \in V$, as before, where $y_v = 1$ indicates that vertex $v \in \overline{F}$, i.e., is part of the DAG and, thus, not included in the DFVS, and is stated as follows:

$$\min |V| - \sum_{v \in V} y_v \tag{8}$$

$$\text{s.t.} \sum_{v \in C} y_v \leq |C| - 1 \qquad \forall C \in \mathcal{C} \tag{9}$$

$$\sum_{v \in K} y_v \leq 1 \qquad \forall K \in \mathcal{K} \tag{10}$$

$$y_v \in \{0, 1\} \qquad \forall v \in V \tag{11}$$

The objective function (8) is the same as in the MTZ-based formulation and minimizes the number of vertices included in the DFVS. The cycle elimination constraints are given by Inequalities (9), where $\mathcal{C}$ denotes the set of all simple cycles in G, and $C \in \mathcal{C}$ is one of these cycles represented as the set of the involved vertices.

As there can be exponentially many cycles in an input graph, the model has exponentially many such constraints. For practically solving the model, we therefore rely on lazy constraint generation for dynamically adding the cycle elimination constraints during the solving process only for cycles actually occurring in intermediate integral solutions. Such cycles are identified by breadth-first search. To speed up the solving process, we statically include from the beginning constraints for *2-cycles*, i.e., cycles of length two. To do so, we derive a graph $G_{2\text{cyc}}$ that consists of the vertices involved in 2-cycles and has an undirected edge for each 2-cycle in the original graph. We then strengthen the initial model by greedily identifying a set $\mathcal{K}$ of cliques in this graph $G_{2\text{cyc}}$. From each such clique, only a single vertex can appear in any DAG, which is expressed in our model by Inequalities (10). Note that a DFVS in G has to be a vertex cover in $G_{2\text{cyc}}$ and adding constraints for cliques is a common way to strengthen formulations for the vertex cover problem. For a detailed description of the greedy construction of cliques, we refer to [4]. The domain of the decision variables is given by (11).

4 Graph Reduction

Our solving procedure starts with the preprocessing of the input graph, where various measures for graph reduction are taken, with the main step being the

repeated application of five reduction rules. We adopt all five rules by Levy and Low [16] as mentioned in Section 2, but we combine the rules IN0 and OUT0 into a single step called IN/OUT0. Our fifth reduction operation called SCC-reduction is inspired by the work of Park and Akers [21]. It is based on the partitioning of the graph into its strongly connected components (SCCs) and directly solving all subproblems with less than three vertices. This rule as well as the LOOP rule are able to identify a part of the vertices belonging to a minimum DFVS, which are then stored in a preliminary DFVS and added to the final solution in the end. As all these operations preserve optimal solutions, their application order can be arbitrary. Nevertheless, a certain order can be beneficial for performance reasons and is in our work as follows: IN/OUT0, IN1, OUT1, LOOP, and SCC.

After the exhaustive application of these reduction rules, the remaining graph is partitioned into its SCCs, which are then sorted according to non-decreasing size. If their number exceeds 1000, we successively merge SCCs in the obtained ordering into subgraphs of up to 100 vertices to reduce memory consumption. All resulting subproblems are processed separately and sequentially, each starting with another repeated application of the five reduction rules. Afterwards, components with exactly three remaining vertices are always cliques that are dealt with by randomly selecting two vertices that are also added to the preliminary DFVS. Subgraphs or SCCs with up to 100 vertices are also handled differently from bigger ones as they are directly solved with the MILP solver. This is also the reason why we chose the size of the merged subgraphs in case of many SCCs to be at most 100 as the small SCCs would have fallen into this category as well. The remaining bigger SCCs are then passed to the LNS.

5 Construction Heuristic and Local Search

An initial solution for the LNS is generated by the consecutive execution of a construction heuristic (CH) and a local search. The employed CH combines the concept of topological orderings with the greedy function by Cai et al. [5]:

$$h(v) = \deg^-(v) + \deg^+(v) - \lambda \cdot |\deg^-(v) - \deg^+(v)| \tag{12}$$

for vertices $v \in V$, where $\deg^-(v)$ and $\deg^+(v)$ denote v's out- and in-degrees, and we use the recommended balancing factor $\lambda = 0.3$. According to this heuristic, vertices with lower h-values are less likely to belong to a minimum DFVS, or in other words, they are more promising to be contained in a maximum topological ordering. Thus, CH sorts the vertices in non-decreasing h-value order and greedily builds a topological ordering of maximal cardinality, skipping all vertices that would introduce an arc to an already selected vertex. In the end, the vertices that are not contained in the topological ordering constitute a valid DFVS and provide an initial solution.

This solution is then possibly further improved by applying LS with the one-flip neighborhood structure, which is based on moving one vertex from the current DFVS to the corresponding DAG if no cycle is introduced. In this LS, the vertices in the DFVS are again considered in non-decreasing h-value ordering.

When accepting a solution, we additionally store the position of the last selected vertex in the ordering to avoid repeated checks of already rejected candidate solutions in the following iterations of the LS. This enhancement does not only strengthen the guidance of the local search but also speeds up the search process.

6 Large Neighborhood Search

Large neighborhood search [22,24] is a prominent LS-based metaheuristics where the key idea is to apply pairs of *destroy* and *repair* methods for identifying promising candidate solutions. A neighborhood is here implicitly defined by the applied destroy operator, which usually selects a subset of the problem's decision variables to be re-optimized while the remaining variables are kept as set in the current incumbent solution. In this way, a subproblem is induced, for which the repair method tries to find an optimal or promising heuristic solution. As in classical LS, we accept new solutions that are better than the incumbent while worse solutions are discarded. Besides, we use a combination of two termination criteria: a runtime limit and a limit on the number of consecutive iterations without improvement.

6.1 Enlarge-DAG Neighborhood Structure

In our *enlarge-DAG neighborhood*, the basic idea is to remove vertices from a current DFVS, which implies to add them and their original incident edges back into the corresponding DAG. The obtained graph is then larger but in general not acyclic anymore. A DFVS subproblem is induced, which is typically smaller than the original DFVSP and therefore should be easier to solve.

More formally, let F be a valid DFVS and the vertex set of the corresponding DAG be denoted by $D = V \setminus F$. The destroy method selects a k-element subset $A \subseteq F$ of the DFVS. The destroyed solution is then represented by $F' := F \setminus A$ and the DAG is augmented to graph $G' = G[\, D' \,]$ with $D' := D \cup A$. In this induced DFVS subproblem, the repair operator has to remove a set $F'' \subset D'$ of vertices to obtain a DAG again. The repair method first applies the graph reduction rules from Section 4 again to possibly reduce G'. As the MILP models we proposed in Section 3 can be highly effective on smaller instances, we ultimately apply them as repair methods.

Still, we need to clarify crucial details on how the selection of the nodes in the destroy operation is done and how the MILP solver actually is applied. In the following paragraphs, we propose different options for these, which we will later experimentally evaluate and compare.

6.2 Degree of Destruction

Parameter k specifies the number of vertices selected in the destroy operation and thus controls the *degree of destruction*. Selecting a suitable value is crucial for an LNS procedure to perform well. If k is too small, the neighborhood is too

restricted and the benefit of applying a powerful repair is mitigated. On the other hand, a too large degree of destruction can result in too complex subproblems. The repair method may then either require too long for single iterations or may yield unsatisfactory results in limited time. Due to the importance of this parameter, we consider different techniques for choosing k as described in the following.

Fixed Degree. In the simplest case, we use a constant value k throughout all LNS iterations and for all DFVSP instances. We denote this strategy by *fixed_ degree*$(\cdot)$ where $\cdot$ stands for the selected value.

Random Selection. Similar to Ropke and Pisinger [23], k is here randomly selected at each LNS iteration from a certain range. While not every choice of k will be ideal in any situation, the hope is that this random selection makes the LNS less sensitive to occasional bad choices and thus generally more robust.

Dynamic Selection. In this more advanced technique, we choose the degree of destruction for a DFVSP instance in dependence on properties of the instance graph and the MILP formulation used for repair, i.e., defining rules to predict suitable fixed values for each instance. To do so, we build on experience gained from earlier performed test runs with the fixed_degree strategy for a range of different values k. We refer to Section 7 and [4] for the details on how these experiments were performed. More specifically, we consider the following five strategies for the dynamic selection.

#2-cycles. In the results for the fixed_degree strategy, we observed that the number of 2-cycles in the input graphs has a substantial impact on how promising different values for k in general are. Making use of these results, we came up with a partitioning of the numbers of 2-cycles into seven consecutive ranges, for which we identified different values for k to yield the best average final solution values. Ties were broken in favor of the smallest k. As the most suitable values also depend significantly on whether the MTZ- or CEC-based model is used for repair, we also distinguish between these as second dimension.

best_ triple. For this strategy, we preselect for each MILP formulation three values for k, called *best-mean*, *mode*, and *most-best*, which performed best over all fixed_degree strategy tests in different senses. Best-mean denotes the degree of destruction that achieves the best mean solution quality just as above but now over all test instances. The mode value was selected by identifying for each instance all k-values that find a solution of smallest known size, taking the smallest in each case, and then determining the mode of these values over all instances. Similarly, most-best is the smallest k for which in general the most solutions of smallest size, hence the best solutions, were found. The LNS then selects in each iteration one of these three values at random.

#2-cycles_best_triple. This variant combines the above two approaches. Thus, we reuse the classification of the graphs into one of the seven number of 2-cycle ranges and the distinction between the used MILP models, and utilize individually derived best-mean and most-best values for k, from which we again select randomly in each LNS iteration.

#2-cycles_regression. Here, we performed a regression to obtain a function yielding the degree of destruction in dependence on the number of 2-cycles. As the distribution of the number of 2-cycles of our benchmark instances as well as the distribution of the instance-best values for k are rather skewed, we found it most meaningful to perform a linear regression over the base-10 logarithms of the numbers of 2-cycles and the base-10 logarithms of the k-values that performed best for each instance. In simplified form, this resulted in

$$k = \begin{cases} 15.85 \cdot z^{0.365} & \text{for the MTZ model} \\ 20.14 \cdot z^{0.433} & \text{for the CEC model,} \end{cases} \tag{13}$$

where z is the the number of 2-cycles. The k-value determined by this function for a given DFVSP instance is then used in each LNS iteration again.

#vertices_regression. For this strategy, we performed a regression over the number of nodes $|V|$ of the graph and the best k-value for each instance. Again, we more specifically applied a linear regression over the base-10 logarithmic values of both. This led to

$$k = \begin{cases} 0.2917 \cdot |V|^{0.808} & \text{for the MTZ model} \\ 0.1159 \cdot |V|^{1.004} & \text{for the CEC model.} \end{cases} \tag{14}$$

6.3 Element Selection

Aside from the degree of destruction, it is also important how elements are chosen in the destroy method as this also impacts the speed of convergence, diversification, and intensification of the search. Remember that in our destroy method, we denoted by A the set of vertices that are selected for removal from the DFVS. Our selection again makes use of the heuristic function $h(v)$ given in Equation (12). However, as a purely greedy selection would lead to a too narrow search and stagnation, we choose each vertex for A by tournament selection. Preliminary experiments indicated that a tournament size of three is a robust choice, which we apply throughout this work.

6.4 Dynamic MILP Selection

We consider two ways of determining which MILP formulation is used in the repair operator of the LNS: preselecting one without any knowledge or choosing it in dependence on instance characteristics as described in the following. Most importantly, we observed differences in the average performance of the MILP

formulations as repair operator for various instances, which seem to be linked to the product of the number of 2-cycles and the density of the graphs. Similarly to the #2-cycles strategy for the selection of the degree of destruction, we therefore determined ranges for the values of these products, in which one of the two MILP models performed better than the other in terms of final solution qualities. A general conclusion for the largest product values is that in the context of the LNS, the CEC-based model is usually the better choice, while for the standalone application the MTZ-based model is on average more beneficial. Detailed results can be found in Chapter 5.6.3 of [4].

7 Computational Study

All experiments were performed on single cores of a cluster with Intel Xeon E5540 processors with a memory limit of 23 GB. Our solving procedures were implemented in Julia 1.7.1, and we used Gurobi 9.5.1[2] as MILP solver in single-threaded mode with a memory limit of 20 GB. Each run had a general time limit of 550 s whereof at most 60 s are used for the LS to improve the initial solution.

We use three existing benchmark instances sets, which all consist of simple, directed graphs without any self-loops or multi-edges. Two of the data sets are taken from the heuristic track of the PACE 2022 challenge[3] and one set[4] comes from Pardalos et al. [20]. We will refer to the two sets from PACE 2022 as *pace-public* and *pace-private* and to the third as *fsp-data*. The sets pace-public and pace-private consist of 100 instances each, and the graphs have quite different sizes and structures with $|V| \in \{843, \ldots, 2394385\}$ nodes and $|E| \in \{2103, \ldots, 5105039\}$ arcs. These instances were mostly generated using KaGen[5] [12], a set of generators for network models, and five graph models, but also contain real-world instances. The fsp-data set consists of 40 smaller graphs which are divided into four subsets with 50, 100, 500, and 1000 vertices, respectively. Each subset contains ten randomly created instances of varying density.

Our main evaluation metric is the *solution quality*, which is calculated for a solution F of an instance with a best known solution F^* as $100\% \frac{|F^*|}{|F|}$. Besides, we employ the *geometric mean* for averaging the solution qualities over all instances of a benchmark set. Both is adopted from the ranking method used in the PACE 2022 challenge. For the pace-public instances, we utilize the smallest solutions recorded by the PACE 2022 challenge as best known solution values. As the submitted solutions for the pace-private instances are not accessible, we have to rely on our own results[6] for this data set. For the fsp-data benchmark

[2] https://www.gurobi.com
[3] https://pacechallenge.org/2022/tracks/#heuristic-track
[4] http://mauricio.resende.info/data/index.html (feedback set problem)
[5] https://github.com/sebalamm/KaGen
[6] https://www.ac.tuwien.ac.at/files/resources/results/DFVSP/DFVSP-benchmarks-results.csv

set, the best known solution values[6] are derived from the results reported by Pardalos et al. [20], Galinier et al. [13], and Tang et al. [26].

7.1 Comparison of MILP Models in Standalone Solving

Before employing the proposed MILP models within the LNS, we have a look at their standalone solving capabilities for the pace-public instances. As for the LNS, we initially apply our graph reduction procedure to each input graph. In case that the reduction yields multiple subproblems, we split the allowed computation time equally among the subproblems and solve them sequentially. If a subproblem is solved to optimality before its assigned time is used up, the spare time is evenly split and distributed over the remaining subproblems. For the CEC-based formulation, we additionally use a time limit for the construction of the 2-cycle constraints as this part may already need too much time for the largest instances. Following preliminary tests, we decided on half of the total time allotted for the solving of a (sub-)instance. We observe that over the whole benchmark set, models MTZ and CEC show similar performances with MTZ being about 2.34 percentage points better in terms of the geometric mean but CEC finding more proven optimal solutions, as shown in Table 1.

As for the LNS an initial solution is always created by CH+LS, we also evaluate the impact of warm starting the standalone MILP solving with solutions from CH+LS. Table 1 and Figure 1 show that providing such initial solutions is clearly beneficial for the MILP solving. There is less variance in the results produced with the warm starts, no solution has less than 45% of the best known solution values, and the solutions exhibit an overall higher quality. The difference between the performances of the CEC- and MTZ-based formulations in terms of average solution quality decreases to less than one percentage point.

Formulation CEC achieves now on average 93.43% quality and outperforms MTZ with an average of 92.58%. The main reason for the significant improvement of especially CEC seems to be linked to the fact that this model performs particularly poorly on a small number of instances. There, the MILP solver returned the solutions from CH+LS but could not further improve on them.

7.2 Comparison of LNS Variants

Now, we evaluate various configurations of the proposed LNS with the enlarge-DAG neighborhood. We consider the different selection strategies for the degree of destruction and the two alternative MILP models for the repair. All experiments are first performed on the pace-public data set and in order to test the generalization capabilities of our dynamic selection strategies, we further apply the approaches to the pace-private data set. The overall time limit is again 550 s per instance. Graph reduction is applied, and if the graph is split into independent subgraphs, the time limit is again distributed as described above for the standalone MILP solving. Moreover, we terminate the LNS also after 50 consecutive unsuccessful LNS iterations. In this case, we expect no or only minor further improvements, and time is probably better spent for solving successive

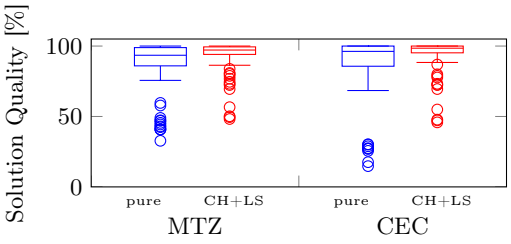

Fig. 1: Box plots showing the solution qualities of standalone MILP solving without (pure) and with warm starts (CH+LS) on pace-public instances.

Table 1: The geometric mean of the solution qualities of standalone MILP solving without (pure) and with warm starts (CH+LS).

Formulation	Avg. Solution Quality [%]	Best Known Solutions
MTZ_{pure}	84.93	18
MTZ_{CH+LS}	92.58	18
CEC_{pure}	82.59	38
CEC_{CH+LS}	93.43	38

subproblems. Gurobi is now given up to 90 s for solving each model to repair a solution. If it fails to find an optimal solution within the first 60 s, the optimality gap is increased to 0.5% for the remaining optimization.

Regarding the degree of destruction, we first investigated the fixed_degree and random selection approaches. For fixed_degree, we considered 15 different values from 25 up to 100000, and for random selection the range to randomly choose from was decided to be $\{5, \ldots, 1000\}$ following results of preliminary tests. We tested the resulting variants with both the MTZ- and the CEC-based formulations on the pace-public benchmark set and compared them in terms of the average solution quality. Figure 2 illustrates the results and Table 2 shows average solution qualities of some of the best performing approaches over the whole pace-public benchmark set. When using CEC, the highest average solution quality of 95.10% is measured for $k = 75$ whereas the MTZ formulation peaks at 93.42% with $k = 25$. The average performance of the random selection lies right in between as both combinations with the two MILP formulations achieve a higher mean than fixed_degree(25) with MTZ and a lower mean than fixed_degree(75) with CEC. This shows that fixing the value of k for all DFVSP instances to a generally best value is not always advantageous as different instances benefit from different degrees of destruction. Note that in comparison to the standalone MILP results, especially those without CH+LS, we can already see a significant advantage of the LNS variants. They achieve improvements of up to twelve percentage points compared with the MILP solving without warm starts and in the range of two to three percentage points compared with the MILP warm start approaches.

We now move on to our dynamic strategies for selecting the degree of destruction as presented in Section 6.2, where we make use of the results gained from the fixed_degree tests. Table 3 shows average solution qualities and how often the best known solutions were achieved for the LNS using the five different dynamic selection strategies in conjunction with either the CEC model, the MTZ model, or the dynamic model selection strategy.

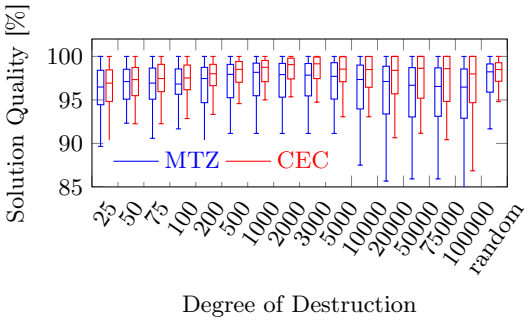

Fig. 2: Box plots showing the solution qualities of fixed_degree and random selection approaches, each with MTZ (left) and CEC (right), on the pace-public instances.

Table 2: The geometric mean of the solution qualities for the pace-public instances.

Selection Strategy	MILP	Avg. Solution Quality [%]
fixed_degree(25)	MTZ	93.42
	CEC	94.20
fixed_degree(75)	MTZ	92.92
	CEC	95.10
random	MTZ	93.72
	CEC	94.87

It turns out that on average, the approaches employing the CEC-based model always achieve better results than the corresponding ones with the MTZ-based model. We also observe that not all combinations of the MILP models with the dynamic selection strategies for k perform better than the random or fixed_degree selection in terms of the average solution quality. This illustrates that even rather simple selection techniques can provide good solutions. However, the best average solution quality is found by the #2-cycles strategy for both MILP models, with a mean of 96.15% for CEC and 94.57% for MTZ. The best_triple strategy when employing the CEC-based formulation achieves 32 best known solutions, which is by far the most and may be linked to the possibility of using very different values for k up to 75000. Such a high value can result in total re-optimization done in the repair operator and this might also be the reason why this strategy does not perform as well on average because it leads to large subproblems that are frequently also too hard to solve with the MILP solver.

When combining the dynamic selection of the employed MILP model with three of the dynamic selection strategies for the degree of destruction, we are able to improve the average solution quality for two of them compared to the corresponding variants with a preselected formulation. The combination with the #2-cycles strategy shows the best performance out of the three with a mean solution quality of 96.21% and 15 best known solutions for the pace-public data set, as given in Table 3. Regarding the average solution quality, this is the overall best achieved result for this benchmark set, which validates the application of the dynamic selection of the MILP formulation.

In order to test the generalization capabilities of the dynamic selection strategies to other DFVSP instances, we have a look in particular at the pace-private data set. We again examine the geometric mean of the solution qualities and the number of found best known solutions, which are shown in Table 3 in the columns labeled pace-private. The observed mean values illustrate that the

Table 3: Mean solution qualities and number of found best known solutions of the LNS variants for three benchmark sets and of the SA-based approaches from the literature.

MILP-based LNS Algorithms		Average Solution Quality [%]			Best Known Solutions		
Selection Strategy	Formulation	pace-public	pace-private	fsp-data	pace-public	pace-private	fsp-data
#2-cycles	MTZ	94.57	97.25	95.49	13	17	15
	CEC	96.15	98.83	**96.37**	12	26	18
	dynamic	**96.21**	**98.96**	96.01	15	26	15
best_triple	MTZ	93.68	96.48	93.38	13	15	15
	CEC	94.53	96.33	94.77	**32**	**39**	**19**
#2-cycles_best_triple	MTZ	94.34	97.10	95.54	13	17	15
	CEC	95.84	97.48	96.23	17	27	**19**
#2-cycles_regression	MTZ	93.89	95.83	94.80	8	12	15
	CEC	95.62	97.53	95.71	15	25	18
	dynamic	95.63	97.44	95.57	14	21	15
#vertices_regression	MTZ	93.62	95.81	94.53	6	10	15
	CEC	94.81	96.77	95.58	9	27	18
	dynamic	94.80	96.79	95.42	9	20	15
SA Algorithms							
SA-FVSP [13]				**99.77**			**27**
SA-FVSP [26]				63.24			1
SA-FVSP-NNS [26]				70.40			1

approaches using the MTZ-based formulation have a worse average performance than the others, with the exception of the best_triple combination where MTZ slightly outperforms CEC. However, the best_triple strategy achieves the lowest average solution quality of all CEC-based approaches, although it again finds the most best known solutions. In terms of mean values, two of the three variants employing the dynamic selection of the MILP formulation achieve improvements and the combination with the #2-cycles strategy again obtains the highest value with an average of 98.96%. Overall, we see similar results for the pace-public and pace-private data set, which confirms that our selection strategies generalize well.

This is also reflected in the observed results for the fsp-data set, as given in Table 3. The #2-cycles strategy in combination with the CEC-based model achieves the highest average solution quality and CEC-based approaches again achieve higher means than MTZ-based ones. The dynamic MILP selection is also superior to the MTZ model but it performs worse than solely using CEC models, which is in contrast to the results for the two pace data sets and indicates room for improvement of this selection strategy.

Lastly, we compare our LNS variants also with two state-of-the-art solving approaches for the DFVSP, which are the simulated annealing based methods SA-FVSP by Galinier et al. [13] and the newer SA-FVSP-NNS by Tang et al. [26]. As we do not have access to the respective implementations, we have to rely on the results reported in these works for the fsp-data benchmark set. Table 3 shows derived average solution qualities and numbers of obtained best solutions at the bottom. As Tang et al. [26] claim to have re-implemented the SA-FVSP from Galinier et al. and tested independently, the table also lists corresponding results

in line SA-FVSP [26]. Unfortunately, there is a strong discrepancy in the results for SA-FVSP originally reported by Galinier et al. and later by Tang et al. for their re-implementation, for which no explanation is given. Thus, all these results need to be taken with care. In comparison to our LNS, the results reported by Galinier et al. are significantly better, while our results clearly dominate those reported by Tang et al. for both, SA-FVSP as well as SA-FVSP-NNS. The performance gap between our approach and SA-FVSP [13] on the fsp-data set might be linked to our focus on the mostly substantially larger instances in pace-public and pace-private. Note also that it is unclear how the SA-based approaches scale to these larger instances and thus, more experiments with these approaches would be highly desirable.

8 Conclusion and Future Work

We proposed two MILP models for the DFVSP, one based on Miller-Tucker-Zemlin inequalities and one based on cycle elimination constraints. Both are effective for graphs up to a certain complexity but do not scale well to larger instances. Therefore, we investigated an LNS framework in which a MILP model is utilized for repairing solutions. Applying a suitable degree of destruction is crucial for the performance of the LNS, and consequently we proposed different selection strategies for this parameter. Among these are five dynamic strategies that exploit results obtained by tests with fixed values. Additionally, we also suggested a dynamic selection strategy for the MILP model to apply. Incorporating these strategies into the LNS framework resulted in multiple variants, which we evaluated on three benchmark sets.

Experiments with standalone MILP solving show that the MTZ model performs best in terms of average solution quality when no initial solution is provided, whereas the CEC model is superior in the case of warm starts with CH+LS. Using the CEC model within the repair operator of the LNS also leads to a higher mean quality than employing the MTZ model. The different LNS variants achieve average solution qualities that are up to 12 percentage points higher than those of standalone MILP solving. Regarding the selection strategies for the degree of destruction, even rather simple techniques such as random selection provide solutions of good quality but are clearly outperformed by some of our dynamic selection strategies, in particular #2-cycles and #2-cycles_best_triple. Combining the dynamic selection techniques with the dynamic selection of the MILP model leads to further improvements and to the overall best performing LNS variant for two benchmark sets.

There is still room left for improvement. With regard to the results of the PACE 2022 challenge, we assume that extending the graph reduction with even more sophisticated rules would be beneficial. Moreover, adopting ideas from the state-of-the-art simulated annealing based approaches, e.g., by trying to improve solutions from the MILP by a fast local search, appears promising. Last but not least, also more advanced machine learning approaches from the field of automated algorithm configuration can be applied.

References

1. Baharev, A., Schichl, H., Neumaier, A., Achterberg, T.: An exact method for the minimum feedback arc set problem. ACM Journal of Experimental Algorithmics **26**, 1–28 (2021). https://doi.org/10.1145/3446429
2. Bar-Yehuda, R., Geiger, D., Naor, J., Roth, R.M.: Approximation algorithms for the vertex feedback set problem with applications to constraint satisfaction and bayesian inference. In: Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms. pp. 344–354. SODA '94, Society for Industrial and Applied Mathematics, USA (1994). https://doi.org/10.5555/314464.314514
3. Bathie, G., Berthe, G., Coudert-Osmont, Y., Desobry, D., Reinald, A., Rocton, M.: PACE Solver Description: DreyFVS. In: Dell, H., Nederlof, J. (eds.) 17th Int. Symposium on Parameterized and Exact Computation. LIPIcs, vol. 249, pp. 31:1–31:4. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2022). https://doi.org/10.4230/LIPIcs.IPEC.2022.31
4. Bresich, M.: Hybrid Metaheuristics Based on Large Neighborhood Search and Mixed Integer Linear Programming for the Directed Feedback Vertex Set Problem. Master's thesis, TU Wien, Austria (2023). https://doi.org/10.34726/hss.2023.102404
5. Cai, X., Huang, J., Jian, G.: Search algorithm for computing minimum feedback vertex set of a directed graph. Jisuanji Gongcheng/Computer Engineering **32**(4), 67–69 (2006)
6. Carrabs, F., Cerulli, R., Gentili, M., Parlato, G.: Minimum weighted feedback vertex set on diamonds. Electronic Notes in Discrete Mathematics **17**, 87–91 (2004). https://doi.org/10.1016/j.endm.2004.09.001
7. Carrabs, F., Cerulli, R., Gentili, M., Parlato, G.: A linear time algorithm for the minimum weighted feedback vertex set on diamonds. Information Processing Letters **94**(1), 29–35 (2005). https://doi.org/10.1016/j.ipl.2004.12.008
8. Chen, J., Liu, Y., Lu, S., O'Sullivan, B., Razgon, I.: A fixed-parameter algorithm for the directed feedback vertex set problem. Journal of the ACM **55**(5), 1–19 (2008). https://doi.org/10.1145/1411509.1411511
9. Cutello, V., Oliva, M., Pavone, M., Scollo, R.A.: An immune metaheuristics for large instances of the weighted feedback vertex set problem. In: 2019 IEEE Symposium Series on Computational Intelligence. pp. 1928–1936. IEEE (2019). https://doi.org/10.1109/SSCI44817.2019.9002988
10. Du, Y., Zhang, Q., Xu, J., Zhang, S., Liao, C., Chen, Z., Sun, Z., Su, Z., Ding, J., Wu, C., Lu, P., Lv, Z.: PACE Solver Description: Hust-Solver – A Heuristic Algorithm of Directed Feedback Vertex Set Problem. In: Dell, H., Nederlof, J. (eds.) 17th Int. Symposium on Parameterized and Exact Computation. LIPIcs, vol. 249, pp. 29:1–29:3. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2022). https://doi.org/10.4230/LIPIcs.IPEC.2022.29
11. Fleischer, R., Wu, X., Yuan, L.: Experimental study of FPT algorithms for the directed feedback vertex set problem. In: Fiat, A., Sanders, P. (eds.) Algorithms – ESA 2009. LNCS, vol. 5757, pp. 611–622. Springer (2009). https://doi.org/10.1007/978-3-642-04128-0_55
12. Funke, D., Lamm, S., Sanders, P., Schulz, C., Strash, D., von Looz, M.: Communication-free massively distributed graph generation. In: 2018 IEEE Int. Parallel and Distributed Processing Symposium. pp. 336–347. IEEE (2018). https://doi.org/10.1109/IPDPS.2018.00043
13. Galinier, P., Lemamou, E., Bouzidi, M.W.: Applying local search to the feedback vertex set problem. Journal of Heuristics **19**(5), 797–818 (2013). https://doi.org/10.1007/s10732-013-9224-z

14. Jatschka, T., Rodemann, T., Raidl, G.R.: A Large Neighborhood Search for a Cooperative Optimization Approach to Distribute Service Points in Mobility Applications. In: Dorronsoro, B., Yalaoui, F., Talbi, E.G., Danoy, G. (eds.) Metaheuristics and Nature Inspired Computing. pp. 3–17. Springer (2022). https://doi.org/10.1007/978-3-030-94216-8_1
15. Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W., Bohlinger, J.D. (eds.) Complexity of Computer Computations. pp. 85–103. Springer (1972). https://doi.org/10.1007/978-1-4684-2001-2_9
16. Levy, H., Low, D.W.: A contraction algorithm for finding small cycle cutsets. Journal of Algorithms **9**(4), 470–493 (1988). https://doi.org/10.1016/0196-6774(88)90013-2
17. Lin, H.M., Jou, J.Y.: On computing the minimum feedback vertex set of a directed graph by contraction operations. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **19**(3), 295–307 (2000). https://doi.org/10.1109/43.833199
18. Melo, R.A., Queiroz, M.F., Ribeiro, C.C.: Compact formulations and an iterated local search-based matheuristic for the minimum weighted feedback vertex set problem. European Journal of Operational Research **289**(1), 75–92 (2021). https://doi.org/10.1016/j.ejor.2020.07.006
19. Noughabi, H.A., Baghbani, F.G.: An efficient genetic algorithm for the feedback set problems. In: 2014 Iranian Conference on Intelligent Systems. pp. 1–4. IEEE (2014). https://doi.org/10.1109/IranianCIS.2014.6802571
20. Pardalos, P.M., Qian, T., Resende, M.G.C.: A greedy randomized adaptive search procedure for the feedback vertex set problem. Journal of Combinatorial Optimization **2**, 399–412 (1998). https://doi.org/10.1023/A:1009736921890
21. Park, S., Akers, S.B.: An efficient method for finding a minimal feedback arc set in directed graphs. In: Proc. of the IEEE Int. Symposium on Circuits and Systems. vol. 4, pp. 1863–1866. IEEE (1992). https://doi.org/10.1109/ISCAS.1992.230449
22. Pisinger, D., Ropke, S.: Large neighborhood search. In: Gendreau, M. (ed.) Handbook of Metaheuristics, pp. 399–419. Springer (2010). https://doi.org/10.1007/978-1-4419-1665-5_13
23. Ropke, S., Pisinger, D.: An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transportation Science **40**(4), 455–472 (2006). https://doi.org/10.1287/trsc.1050.0135
24. Shaw, P.: Using constraint programming and local search methods to solve vehicle routing problems. In: Maher, M., Puget, J.F. (eds.) Principles and Practice of Constraint Programming. pp. 417–431. Springer (1998). https://doi.org/10.1007/3-540-49481-2_30
25. Swat, S.: PACE Solver Description: DiVerSeS – A Heuristic Solver for the Directed Feedback Vertex Set Problem. In: Dell, H., Nederlof, J. (eds.) 17th Int. Symposium on Parameterized and Exact Computation. LIPIcs, vol. 249, pp. 27:1–27:3. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2022). https://doi.org/10.4230/LIPIcs.IPEC.2022.27
26. Tang, Z., Feng, Q., Zhong, P.: Nonuniform neighborhood sampling based simulated annealing for the directed feedback vertex set problem. IEEE Access **5**, 12353–12363 (2017). https://doi.org/10.1109/ACCESS.2017.2724065