Threshold-Based Responsive Simulated Annealing for Directed Feedback Vertex Set Problem

Qingyun Zhang[1], Yuming Du[1], Zhouxing Su[1*], Chu-Min Li[2], Junzhou Xu[3], Zhihuai Chen[3] Zhipeng Lü[1*]

[1]School of Computer Science and Technology, Huazhong University of Science and Technology, China
[2]MIS, University of Picardie Jules Verne, France
[3]TCS Lab, Huawei Technologies Co., Ltd., China
{qingyun_zhang, yumingdu, suzhouxing}@hust.edu.cn, chu-min.li@u-picardie.fr, {xujunzhou, chenzhihuai1}@huawei.com, zhipeng.lv@hust.edu.cn

Abstract

As a classical NP-hard problem and the topic of the PACE 2022 competition, the directed feedback vertex set problem (DFVSP) aims to find a minimum subset of vertices such that, when vertices in the subset and all their adjacent edges are removed from the directed graph, the remainder graph is acyclic. In this paper, we propose a threshold-based responsive simulated annealing algorithm called TRSA for solving DFVSP. First, we simplify the problem instances with two new reduction rules proposed in this paper and eight reduction rules from the literature. Then, based on a new solution representation, TRSA solves DFVSP with a fast local search procedure featured by a swap-based neighborhood structure and three neighborhood acceleration strategies. Finally, all these strategies are incorporated into a threshold-based responsive simulated annealing framework. Computational experiments on 140 benchmark instances show that TRSA is highly competitive compared to the state-of-the-art methods. Specifically, TRSA can improve the best known results for 53 instances, while matching the best known results for 79 ones. Furthermore, some important features of TRSA are analyzed to identify its success factors.

Introduction

The directed feedback vertex set problem (DFVSP) is a classical discrete optimization problem and has been proven to be NP-hard (Yannakakis 1978). It involves obtaining a directed acyclic graph by removing as few vertices as possible from a given directed graph. DFVSP has wide applications in various domains, ranging from Very Large Scale Integration (VLSI) circuit design (Hudli and Hudli 1994; Orenstein, Kohavi, and Pomeranz 1995), partial scan design of circuit (Lee and Reddy 1990), program verification (Seymour 1995), deadlock resolution (Jain, Hajiaghayi, and Talwar 2005), network attack (Mugisha and Zhou 2016), constraint satisfaction (Bar-Yehuda et al. 1994), Bayesian inference (Bar-Yehuda et al. 1998), tournament (Ramanujan and Szeider 2017; Zehavi 2023), and so on.

DFVSP is the topic of the Parameterized Algorithms and Computational Experiments (PACE) 2022 competition, which consists of two tracks: Exact track and heuristic track. As a classical combinatorial optimization, DFVSP has been studied for nearly 60 years (Lempel and Cederbaum 1966). The solution methods of DFVSP can be mainly categorized into exact algorithms, approximation algorithms, and metaheuristic algorithms. Specifically, Smith and Walford (1975) presented a graph partitioning technique, which was the first algorithm in the literature to find an optimal solution for DFVSP. Lin and Jou (2000) used three new powerful reduction rules, and developed an exact algorithm based on the branch-and-bound framework. Qian, Ye, and Pardalos (1996) designed a non-polynomial pseudo ϵ-approximate algorithm for solving DFVSP. Pardalos, Qian, and Resende (1998) developed a greedy randomized adaptive search (GRASP) algorithm for solving DFVSP. Based on Pardalos, Qian, and Resende (1998), Festa, Pardalos, and Resende (2001) presented FORTRAN subroutines using GRASP for solving DFVSP and directed feedback arc set problem (DFASP). Galinier, Lemamou, and Bouzidi (2013) presented a powerful simulated annealing algorithm (SA-FVSP), which uses a new representation of solutions based on topological ordering and a new neighborhood structure. Cutello and Pappalardo (2015) proposed an enhanced genetic algorithm with a special local search improvement strategy. Zhou (2016) presented a belief propagation-guided decimation (BPD) algorithm to solve DFVSP. Based on SA-FVSP (Galinier, Lemamou, and Bouzidi 2013), Tang, Feng, and Zhong (2017) introduced a nonuniform neighborhood sampling (NNS) strategy, and Russo et al. (2022) proposed a stochastic simulated annealing algorithm. Recently, Sun et al. (2023) presented an efficient stochastic local search algorithm called IDTS which alternates between a thresholding search stage and a descent stage to solve DFVSP.

This paper presents a threshold-based responsive simulated annealing algorithm (TRSA) to solve DFVSP. TRSA is a combination of threshold search strategy and simulated annealing strategy with 'temperature rise', aiming to strengthen the search capabilities by employing a neighborhood structure based on the closing-opening swap, an incremental neighborhood evaluation technique, and three neighborhood acceleration strategies. The main contributions can be summarized as follows:

- Different from other metaheuristic algorithms such as SA-FVSP and IDTS that use topological ordering for so-

*Corresponding Authors

lution representation, TRSA represents a solution by directly using the set of the remaining vertices, which is more meaningful and essential because each solution exactly corresponds to a unique graph.

- TRSA reduces the problem instances by 10 reduction rules, two of which are newly proposed in this paper.
- TRSA proposes an effective threshold-based responsive simulated annealing algorithm for local improvement, which enables a balance between intensification and diversification of the search.
- TRSA employs a new dedicated swap-based neighborhood structure based on a closing operation and k opening operations. In order to improve the search efficiency, TRSA employs an incremental evaluation strategy and three neighborhood acceleration strategies which consist of threshold-based first-fit strategy, and elimination of bidirectional edges and cycles.
- Tested on two sets of totally 140 instances including 40 classical instances and 100 instances used in the PACE 2022 competition on the heuristic track, TRSA outperforms the state-of-the-art algorithms in the literature and the top competitors in the competition by improving the best known results for 53 instances.

Problem Description

Let $G = (V, E)$ be a directed graph, where V is the vertex set and $E \subseteq V \times V$ is the edge set. If G becomes acyclic by removing all vertices in a subset and their adjacent edges, such a subset is called a feedback vertex set, denoted as FVS. The directed feedback vertex set problem (DFVSP) aims to find a minimum FVS. Formally, let $X \subseteq V$ denote the selected subset to be removed, DFVSP consists of minimizing $|X|$ while satisfying the constraint that the subgraph $G' = G\backslash(X, Y)$ is acyclic, where $Y = \{(u, v) \in E | u \in X \vee v \in X\}$. The subgraph G' is a directed acyclic graph (DAG). Then, the subset X is a FVS of graph G. The objective function $f(X)$ of DFVSP can be defined as Eq. (1).

$$\min f(X) = |X| \tag{1}$$

For an edge $(u, v) \in E$, we say that u is a predecessor or in-coming neighbor of v, and v is a successor or out-going neighbor of u. For vertex $v \in V$, $N^-(v)$ and $N^+(v)$ denote the sets of in-coming and out-going neighbors of vertex v, respectively, and $|N^-(v)|$ and $|N^+(v)|$ represent the in-degree and out-degree of vertex v, respectively.

Threshold-based Responsive Simulated Annealing Algorithm (TRSA)

General Framework

Different from the classical simulated annealing used in Galinier, Lemamou, and Bouzidi (2013), we propose a new effective threshold-based responsive simulated annealing algorithm (TRSA) for solving DFVSP, which combines threshold search strategy and simulated annealing strategy with 'temperature rise'. In traditional simulated annealing algorithm, the temperature keeps falling all the time. Therefore, it tends to be a random-walk algorithm due to high temperature in the early search period, while it would behave like a hill-climbing algorithm in the late search period when the temperature falls so that it is easy to fall into the local optimal trap. This will lead to the poor stability of the traditional simulated annealing algorithm for solving DFVSP. To overcome this drawback, TRSA employs a threshold search strategy, which uses a threshold upper bound to effectively limit the range of search space so that it can avoid searching too randomly, and a warming mechanism at the late stage of the search to effectively jump out of the local optimal trap.

The framework of TRSA is presented in Algorithm 1. After applying two new proposed reduction rules, together with eight reduction rules from the literature, to simplify the problem instances (line 1), TRSA employs a random greedy construction heuristic to generate an initial feasible solution X (line 2). Next, TRSA optimizes the incumbent solution X iteratively through a number of search rounds with different temperature values (lines 8–12). At each search round, a fast local search is performed for $maxIter$ iterations at the current temperature T. After performing $maxIter$ iterations, if the best solution is improved, parameter T is decreased by a constant cooling factor α ($0 < \alpha < 1$), and the best objective value is updated (lines 13–16). When the best solution has not been improved for $maxUnImp$ rounds, temperature T is heated γ times by the constant heating factor $1/\alpha$ (lines 20–23). Once the time limit is reached, TRSA terminates and returns the best found solution X^*.

Reduction Rules

Reduction rules play an important role in solving DFVSP because they can significantly reduce the instance size while preserving the information necessary for finding the minimum FVS, enhancing the search efficiency for large scale instances, and not making the solution worse. We adopt ten reduction rules to eliminate some vertices and edges, and deduce that some vertices must be included in the optimal solution. These rules include SL, IN0/OUT0 and IN1/OUT1 proposed in Levy and Low (1988), PIE, CORE and DOME proposed in Lin and Jou (2000), and DOMV and MC proposed in this paper. They are repeatedly applied in sequence until no more vertices or edges can be removed or fixed.

1) SL: If there is one vertex in a cycle, the cycle is called a self-loop. For a vertex $v \in V$ with a self-loop, it can be fixed to FVS. 2) & 3) IN0/OUT0: If there is a vertex v whose in-degree $|N^-(v)| = 0$ or out-degree $|N^+(v)| = 0$, obviously, this vertex is not on any cycle, so this vertex together with its corresponding edges can be eliminated from the graph G. 4) & 5) IN1/OUT1: For a no self-loop vertex v with $|N^-(v)| = 1$ or $|N^+(v)| = 1$, where u is the only in-coming neighbor or out-going neighbor of v, we merge v into u as a single vertex. 6) PIE: Let PIE be the set of all bidirectional edges. In a graph G without self-loops, the PIE operation eliminates the acyclic edges between strongly connected graphs and in the subgraph $G\backslash$PIE. 7) CORE: Given a subgraph $G' = (V', E')$ without self-loops whose edges are bidirectional, if each vertex is the neighbor of other vertices in V', then G' is said to be a d-clique of G. A vertex

Algorithm 1: The main framework of the TRSA algorithm

Input: A directed graph G
Output: The best solution found so far X^*

Reduce the graph by ten reduction rules
Initial solution $X \leftarrow \text{Init}(G)$
$X^* \leftarrow X$, the best objective value $f^* \leftarrow f(X^*)$
Temperature $T \leftarrow T_0$
Unimproved counter $unImprove \leftarrow 0$
while Termination condition is not met **do**
  Iteration counter $Iter \leftarrow 0$
  while $Iter < maxIter$ **do**
    Threshold value $tv \leftarrow$ Calculate_tv(X^*)
    $X, X^* \leftarrow$ITSP(G, X, X^*, tv, T) /* Algorithm 2 */
    $Iter \leftarrow Iter + 1$
  end while
  if $f^* > f(X^*)$ **then**
    $f^* \leftarrow f(X^*)$
    $T \leftarrow \alpha \times T$ /* Temperature down */
    $unImprove \leftarrow 0$
  else
    $unImprove \leftarrow unImprove + 1$
  end if
  if $unImprove \geq maxUnImp$ **then**
    $T \leftarrow \alpha^{-\gamma} \times T$ /* Temperature up */
    $unImprove \leftarrow 0$
  end if
end while
return X^*

$v \in V'$ is a CORE of G' if its incident edges are in E'. For a d-clique $G' \subseteq G$, if a vertex v is a CORE, we can eliminate it, fix $V' \backslash \{v\}$ to FVS and remove all edges in E'. 8) DOME: For a vertex u, if (u, v) is a bidirectional edge, vertex $u[v]$ is called Π-neighbor of vertex $v[u]$. Otherwise, vertex $u[v]$ is called non-Π-in-coming (non-Π-out-going) neighbor of vertex $v[u]$. In a graph G without self-loops, if an edge (u, v) satisfies the condition that the non-Π-in-coming neighbors set of vertex u is the subset of the in-coming neighbors set of vertex v or the non-Π-out-coming neighbors set of vertex v is the subset of the out-going neighbors set of vertex u, the edge (u, v) can be removed.

The rules DOMV and MC are as follows:

- **Dominated vertex (DOMV)**: Let $N_\Pi(u)$ denote the vertex set that consists of Π-neighbors of vertex u. In a graph G without self-loops, if (u, v) is a bidirectional edge and in-coming or out-going neighbors set of v except for u is the subset of Π-neighbors set of u, that is, $N^-(v) \backslash \{u\} \subseteq N_\Pi(u)$ or $N^+(v) \backslash \{u\} \subseteq N_\Pi(u)$, then vertex v is dominated by vertex u. Since edge (u, v) has at least one vertex in FVS, we can fix vertex u to FVS and remove all corresponding edges from G.

  Proof. For a bidirectional edge (u, v), vertex v is dominated by vertex u. In order to eliminate the cycle $\langle u, v, u \rangle$, at least one of the vertices u and v must be in FVS. Assuming u is not in FVS, then v must be in FVS. Since $N_\Pi(u)$ contains all in-coming or out-going neighbors of v, in the worst case, there must exist an equivalent solution by replacing v with u.

  An example is illustrated in Figure 1, where (a, c) is a bidirectional edge, and vertices a and c are the Π-neighbors for each other. Vertex set $\{b, c, d\}$ is Π-neighbors set of vertex a, while set $\{d\}(\{b\})$ is in-coming (out-going) neighbors set of vertex c except for vertex a. It satisfies condition DOMV. Thus, vertex c is dominated by vertex a and vertex a can be fixed to FVS.

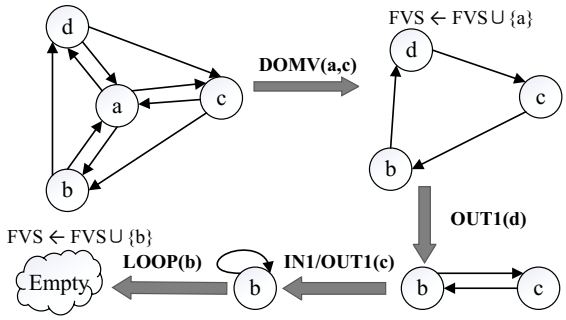

Figure 1: A reduction of a graph without self-loops.

- **Maximum clique (MC)**: For a strongly-connected component $G' \subseteq G$ with bidirectional edges only, it is necessary to remove at least one of any two vertices u, v connected by an edge from the subgraph, in order to break the cycle $\langle u, v, u \rangle$, i.e., there are no connected vertices. We can transform this problem into a maximum clique problem of the complementary graph. If the size of a subgraph G' is small enough ($n < 600$), we solve the maximum clique problem with the exact algorithm proposed in Li et al. (2018), eliminate the exact vertices and all their adjacent edges, and fix other vertices in G' to FVS.

  Proof. To eliminate the cycles formed by bidirectional edges in the strongly connected subgraph, some vertices must be fixed to FVS so that the remaining vertices in the subgraph are not connected. Hence, we need to remove the minimum number of vertices from the strongly-connected subgraph (according to the objective of minimizing FVS) so that the remaining vertices are the (maximum) independent set (Luby 1986). Since the maximum independent set of the original graph is the maximum clique of its complementary graph, the minimum FVS of this subgraph can be obtained by solving the maximum clique of the complementary graph.

Initial Solution

After simplifying the graph by the reduction rules, we employ a random greedy constructive heuristic to generate an initial feasible solution. First, all vertices are opened as feedback vertices and are added to the current solution X. Then, vertices are selected and closed from X successively while ensuring the solution feasibility, until no vertices can be directly closed. At each iteration, with equal probability TRSA selects a vertex randomly or with the minimum degree in X, and then a better solution is obtained by closing the selected vertex $u \in X$, which consists of removing u from X and adding it into $G \backslash \{X\}$ with its original adjacent edges with-

Algorithm 2: Iterated threshold search procedure

function ITSP(G, X, X^*, tv, T)

$X' \leftarrow X$, $S \leftarrow$ with equal probability, shuffle X randomly or sort X in an ascending order of degree
for $u \in S$ **do**
  if $f(X) \geq tv$ **then**
    $l \leftarrow 1$ /* Maximum number of opened vertices */
  else
    Random number $p \leftarrow random(0, 1]$
    $l \leftarrow$ the largest value that satisfies $e^{((-l+1)/T)} > p, l \in [L, L-1, ..., 1]$ /* L is the maximum number of open vertices*/
  end if
  Close vertex u and $X' \leftarrow X' \backslash \{u\}$
  if $l \geq |N_\Pi(u)|$ **then**
    Open vertices in $N_\Pi(u)$ and $X' \leftarrow X' \cup N_\Pi(u)$
    $k \leftarrow |N_\Pi(u)|$
    if OPEN_VERTICES(G, X', l, k) = True **then**
      $X \leftarrow X'$
    end if
  end if
  if $f(X^*) > f(X)$ **then**
    $X^* \leftarrow X$
  end if
end for
return X, X^*

Algorithm 3: Choosing opened vertices

function OPEN_VERTICES(G, X', l, k)

if G is acyclic **then**
  return True
end if
if $l < k$ **then**
  return False
end if
Find a cycle C from G
With equal probability, shuffle the vertex set V_C of C randomly or sort V_C in a descending order of degree
for $v \in V_C$ **do**
  Open vertex v and $X' \leftarrow X' \cup \{v\}$
  if OPEN_VERTICES($G, X', l, k+1$) = True **then**
    return True
  else
    Close vertex v and $X' \leftarrow X' \backslash \{v\}$
  end if
end for

out forming any cycle. This procedure is repeated until all vertices in X have been tried to be closed.

Neighborhood Structure and Evaluation

The neighborhood structure describes the adjacency relationship between the incumbent solution and its neighboring solutions in the solution space. For an incumbent solution X, performing a neighborhood move m produces a new neighboring solution, i.e., $X \oplus m$. To tackle DFVSP, the proposed TRSA adopts a swap-based neighborhood structure, which consists of a closing operation and k ($k \geq 0$) opening operations. Specifically, at each iteration, it tries to obtain a neighboring solution X' based on the current solution X by closing a vertex $u \in X$ and opening k vertices $v_1, v_2, ...v_k$ ($v_i \in V \backslash X$) such that $X' = X \oplus m = X \cup \{v_1, v_2, ..., v_k\} \backslash \{u\}$.

In a local search procedure, the neighborhood evaluation is an extremely important ingredient. Instead of naively evaluating the objective of each neighboring solution by Eq. (1), TRSA adopts an incremental evaluation technique. We maintain the improvement $\Delta(m)$ that performs a neighborhood move m. Specifically, the objective of a neighboring solution can be calculated incrementally by the numbers of opening and closing vertices, i.e., $f(X \oplus m) = f(X) + \Delta(m) = f(X) - 1 + k$. Obviously, when $k = 0$, an improved solution can be obtained.

Neighborhood Acceleration Strategies

According to the definition of the neighborhood structure, a naive neighborhood evaluation technique selects k vertices from $V \backslash X$ for opening, whose complexity is $O(|V \backslash X|^k)$. For large-scale instances, this kind of neighborhood evaluation is too time-consuming. Therefore, we adopt the following three strategies to accelerate the search.

- **Neighborhood reduction based on cycle elimination**: If a solution is feasible, there is no cycle in the graph. That is to say, when a vertex is closed in the current solution X, all cycles must be eliminated in order to find a new feasible solution. Therefore, an opening operation selects a vertex from a cycle instead of the entire space $V \backslash X$, called "critical one-operation". We can reduce the neighborhood by finding a cycle in turn and eliminating it. To eliminate a cycle, just need to open one vertex on the cycle. In most cases, the number of vertices on a cycle is much less than $|V \backslash X|$ such that the complexity of the neighborhood evaluation can be significantly reduced.
- **Neighborhood reduction based on bidirectional edge elimination**: If there is a bidirectional edge (u, v) in the graph G, at least one of the two endpoints of this edge must be in FVS in order to break the cycle $\langle u, v, u \rangle$. Thus, when we close a vertex u which is an endpoint of some bidirectional edges, the Π-neighbors of u must be opened such that the bidirectional edges can be eliminated.
- **Threshold-based first-fit strategy**: It is obvious that the bigger the value of k is, the worse a neighboring solution could be. Therefore, we adopt a threshold-based first-fit strategy, which limits the number k of opened vertices. Specifically, we open vertices one by one until the new solution is feasible, while the number of opened vertices should be no more than a maximum value, which is decided by a threshold and a random factor.

Iterated Threshold Search Procedure

We employ an iterated threshold search procedure (ITSP) to iteratively improve the initial solution.

Algorithm 2 presents the framework of ITSP. It first reorders the FVS $S \leftarrow X$ by randomly shuffling or sorting

it in an ascending order of degree with equal probability. Then, it iteratively performs the swap move by closing vertex u and opening at most l vertices to explore a large solution space. Based on threshold-based first-fit strategy, at each iteration, ITSP first determines the maximum number of vertices l that can be opened, which decides the size of the current neighborhood and the tolerance of accepting worse solutions. There are two circumstances: 1) If the number of feedback vertices of the current solution X reaches the threshold tv, i.e., $f(X) \geq tv$, the quality of the neighboring solution cannot be worse than the current solution, i.e., $l = 1$ (lines 3–4); 2) If $f(X) < tv$, we can explore larger search space. According to the upper limit of the number of opened vertices L and the current temperature T, we evaluate the tolerance of accepting deteriorating solutions in the worst case and obtain the maximum number of vertices that can be opened. We choose the largest value of l which satisfies $e^{((-l+1)/T)} > p$, where $l \in [L, L-1..., 1]$ (lines 5–7).

Second, according to neighborhood reduction based on bidirectional edge elimination, to eliminate the bidirectional edges caused by closing vertex u, all vertices in $N_{\Pi}(u)$ need to be opened. However, if $|N_{\Pi}(u)| > l$, the algorithm does not search for neighborhood solutions related to closing u, because a feasible solution cannot be found by opening l or fewer vertices.

Third, as presented in Algorithm 3, according to neighborhood reduction based on cycle elimination, ITSP recursively finds a cycle, and eliminates it by opening a vertex on the cycle to obtain a new solution X' until the number of opened vertices reaches l or X' is feasible. For a directed acyclic graph G, if new cycles are generated after adding a new vertex, this vertex must be on these cycles. Therefore, when closing vertex $u \in X$, we use depth-first search (DFS) (Tarjan 1972) to iteratively find a cycle starting from u. For each cycle to be eliminated, we iteratively try to open a vertex on the cycle in randomly shuffled order or descending order of degree with equal probability (lines 8–16). The complexity of finding a cycle is $O(|V - X| + |E|)$.

Finally, X is updated to be X' iff it is feasible (lines 13–15), and if the current solution X improves the best solution found so far, X^* is updated with X (lines 17–19).

In addition, the threshold value tv is critical to the performance of ITSP. It ensures that if the objective reaches a limit, only better or equal solutions can be accepted. A large threshold value may cause a low convergence rate for ITSP, while a too small threshold value can greatly weaken its exploration power. Thus, in order to balance diversification and intensification, the threshold value tv is dynamically tuned according to the objective value of the recorded best local optimum f^*, the threshold ratio tr, and the percentage of the remaining time $\frac{T_{\text{remain}}}{T_{\text{total}}}$, which can be defined as Eq. (2).

$$tv = (f^* + L) + tr \times f^* \times \frac{T_{\text{remain}}}{T_{\text{total}}} \tag{2}$$

where T_{total} is the total time limit, T_{remain} is the remaining time, and tr is a threshold ratio inspired from Chen and Hao (2015) and is defined to be $tr = \frac{tr_a}{f^*} + tr_b$, with tr_a and tr_b being two fixed coefficients. During the search process, tr is dynamically recalculated as long as f^* is updated.

Algorithm	Description
SA-FVSP	Simulated annealing (Galinier, Lemamou, and Bouzidi 2013)
IDTS	Iterated dynamic thresholding search (Sun et al. 2023)
DiVerSeS	Simulated annealing
DreyFVS	Two greedy local search heuristics
HustFVS	Two-stage local search[1]

Table 1: Description of reference algorithms.

Parameter	Value	Tested values	Description
T_0	0.6	{0.5, 0.6, 0.7}	Initial temperature
α	0.99	{0.99, 0.98}	Cooling factor
γ	5	{4, 5, 6}	Heating magnitude
L	5	{3, 4, 5, 6, 7}	Maximum number of open vertices
$maxIter$	20	{15, 20, 30}	Maximum iteration counter
$maxUnImp$	50	{40, 50, 60}	Maximum unimproved counter
tr_a	30	[10, 100]	A coefficient to calculate tr
tr_b	0.004	[0.001, 0.01]	A coefficient to calculate tr

Table 2: Parameter settings of our TRSA.

Experiments and Analysis

In order to evaluate the effectiveness of our proposed TRSA, we conduct extensive experiments on two datasets consisting of totally 140 instances, and compare the performance of TRSA with the state-of-the-art algorithms in the literature and the participating solvers in the PACE 2022 competition.

Experimental Protocol

There are mainly two sets of benchmark instances for DFVSP. The first set consists of 40 public classical instances, which were generated in Pardalos, Qian, and Resende (1998). For this benchmark, there are four groups with 50, 100, 500, and 1000 vertices, respectively, with 10 graphs in each group. The second set consists of 100 instances introduced in the PACE 2022 competition on the heuristic track, with up to 800,000 vertices and 5,000,000 edges[2]. Table 1 gives the reference algorithms. SA-FVSP and IDTS are the state-of-the-art algorithms for DFVSP in the literature, while DiVerSeS, DreyFVS, and HustFVS are the top-three algorithms in the PACE 2022 competition on the heuristic track. IDTS were run on a computer with Intel(R) Core(TM)2 Duo CPU T7700 2.4GHz.

Our proposed TRSA is programmed in C++. For a fair comparison, we rerun the algorithms of the top three teams in the PACE 2022 competition on the heuristic track, and reimplement and rerun the classical SA-FVSP. All experiments are carried out on Server 2012 x64 with Intel Xeon E5-2609v2 2.5 GHz CPU and these algorithms are performed for 30 independent runs for each instance in both sets. For 40 public classical instances, we execute SA-FVSP, DiVerSeS, DreyFVS, HustFVS, and our TRSA on each instance under a 30-second time limit on a single CPU core,

[1]Note that this is the preliminary version of our TRSA.

[2]https://pacechallenge.org/2022/01/12/public-instances

Ins.	SA-FVSP		IDTS		DiVerSeS		TRSA	
	f_{min}	f_{avg}	f_{min}	f_{avg}	f_{min}	f_{avg}	f_{min}	f_{avg}
P50-100	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**
P50-150	**9**	**9**	**9**	**9**	**9**	**9**	**9**	**9**
P50-200	**13**	**13**	**13**	**13**	**13**	**13**	**13**	**13**
P50-250	**17**	**17**	**17**	**17**	**17**	**17**	**17**	**17**
P50-300	**19**	**19**	**19**	**19**	**19**	**19**	**19**	**19**
P50-500	**28**	**28**	**28**	**28**	**28**	**28**	**28**	**28**
P50-600	**31**	32	**31**	**31**	**31**	**31**	**31**	**31**
P50-700	**33**	**33**	**33**	**33**	**33**	**33**	**33**	**33**
P50-800	**34**	**34**	**34**	**34**	**34**	**34**	**34**	**34**
P50-900	**36**	**36**	**36**	**36**	**36**	**36**	**36**	**36**
P100-200	**9**	**9**	**9**	**9**	**9**	**9**	**9**	**9**
P100-300	**17**	17.3	**17**	**17**	**17**	**17**	**17**	**17**
P100-400	**23**	**23**	**23**	**23**	**23**	**23**	**23**	**23**
P100-500	**32**	**32**	**32**	**32**	**32**	**32**	**32**	**32**
P100-600	**36**	37.2	37	37	**36**	**36**	**36**	**36**
P100-1000	**53**	53.2	**53**	**53**	**53**	**53**	**53**	**53**
P100-1100	**54**	55	**54**	54.7	**54**	**54**	**54**	**54**
P100-1200	**57**	57.4	**57**	**57**	**57**	**57**	**57**	**57**
P100-1300	**60**	**60**	**60**	**60**	**60**	**60**	**60**	**60**
P100-1400	**61**	**61**	**61**	**61**	**61**	**61**	**61**	**61**
P500-1000	**31**	**31**	**31**	**31**	**31**	**31**	**31**	**31**
P500-1500	**63**	64.8	**63**	63.8	**63**	63.2	**63**	**63**
P500-2000	102	103.8	101	102.8	101	102.4	**100**	**100**
P500-2500	133	134.8	132	135.1	133	134	**131**	**131.3**
P500-3000	163	165.1	163	164.9	163	163.9	**161**	**161.5**
P500-5000	**237**	239.2	**237**	240.1	**237**	**237**	**237**	**237**
P500-5500	252	253.6	252	254.7	**251**	252.2	**251**	**251.4**
P500-6000	265	266.9	**264**	267.6	265	265.1	**264**	**264.3**
P500-6500	**276**	278.6	**276**	278.5	**276**	276.3	**276**	**276**
P500-7000	287	289	287	288.7	**286**	286.9	**286**	**286**
P1000-3000	130	132.1	128	129.9	128	129	**127**	**127**
P1000-3500	165	167.3	162	164.3	162	164.5	**159**	**159.7**
P1000-4000	192	195.9	193	195.5	192	194.3	**190**	**190.3**
P1000-4500	230	231.7	229	231.5	229	230.5	**226**	**226.2**
P1000-5000	260	265.5	261	263.2	259	261	**256**	**256.3**
P1000-10000	472	476.7	472	475.1	469	471.9	**466**	**468.7**
P1000-15000	582	584.4	580	585.6	579	581.4	**578**	**579.7**
P1000-20000	653	655.7	652	657.3	651	652.5	**650**	**651.8**
P1000-25000	702	704.7	700	704.4	**699**	**701.4**	**699**	701.5
P1000-30000	741	744.1	741	744.1	740	741	**739**	**740**
p-value	4.4E-04	1.2E-05	4.4E-04	6.0E-05	1.5E-03	2.3E-04		
#best	24	15	24	19	27	23	40	39
gap(%)	19.56	49.45	16.42	35.99	11.48	18.94	0	0.01

Table 3: Experimental results on 40 classical instances.

which is the same as adopted in Galinier, Lemamou, and Bouzidi (2013). For the 100 instances used in the PACE 2022 competition, since they are large and challenging, the cutoff time for each instance is 600 seconds, which is the same as the time limit used in the competition.

Parameter Settings

Table 2 shows the parameter settings of our TRSA, which can be considered as the default setting of the algorithm. Unless otherwise specified, this default setting is consistently used throughout all the experiments presented in this study. The parameters of our algorithm are tested respectively according to the values in column "Tested values". We consider all combinations of these parameter values on several different scale instances (P1000-3000, h_025, h_085, h_105, h_155) and finally choose the parameters with the best overall performance on these instances. The final value of each parameter is recorded in "Value".

Computational Results

The detailed computational results are reported in Tables 3 and 4. Column "Ins." gives the names of the instances. Columns "f_{min}" and "f_{avg}" report the best and the average results of the solutions found over 30 independent runs. Rows "#B", "#E", and "#W" indicate the number of instances where our TRSA obtains better, equal, and worse results comparing to the corresponding algorithms, respectively. To verify the statistical significance of the comparison between TRSA and the reference algorithms, we give the p-values by the non-parametric Wilcoxon test in row "p-value", where a p-value less than 0.05 indicates a significant difference. Row "#best" reports the number of instances for which each algorithm obtains the best results among all algorithms. Row "gap" gives the total gap between the results obtained by the corresponding algorithms and the best results on the current set of instances.

Table 3 reports the results on 40 classical instances by SA-FVSP, IDTS, DiVerSeS, and our TRSA. In terms of the best results, TRSA improves the best known results of all reference algorithms on 13 instances and matches the best results for all the remaining ones. In terms of the average results, TRSA obtains better results on 17 instances and matches the best results of all reference algorithms on the remaining ones except for P1000-25000, where it performed only slightly worse than DiVerSeS. Moreover, as for the best results obtained by all reference algorithms, TRSA keeps an advantage of over 11% and 18% in terms of the best and average results, respectively. The small p-values (<0.05) confirm that there are significant differences between our results and those of all the reference algorithms.

Table 4 shows the results of TRSA, SA-FVSP, and the top three competitors on the 100 instances used in the PACE 2022 competition. TRSA obtains the best results for 92 (out of 100) instances and improves the best known results on 40 instances. The total gap between the best results of our TRSA and the best results found by all the algorithms (including ours) is only 0.4%, while other competitors obtain the best results only on half of the 100 instances or less. Compared to the best-performing algorithm DiVerSeS, TRSA obtains 47 better, 47 equal, and 6 worse solutions in terms of the best results. The small p-values (<0.05) also indicate the dominance of our TRSA in terms of the best and average results. In general, the results of TRSA are significantly better than all reference algorithms.

Effectiveness of the Proposed Strategies

In order to evaluate the merits of two new reduction rules, the neighborhood reduction based on bidirectional edge elimination, and the threshold-based first-fit strategy, we compare TRSA with three alternative versions.

- **TRSA-A**: Disable the DOMV and MC rules.
- **TRSA-B**: Disable the neighborhood reduction based on bidirectional edge elimination.
- **TRSA-C**: Disable the threshold-based first-fit strategy, that is, the number k of opened vertices is not limited.

Ins.	SA-FVSP		DiVerSeS		DreyFVS		HustFVS		TRSA	
	f_{min}	f_{avg}	f_{min}	f_{avg}	f_{min}	f_{avg}	f_{min}	f_{avg}	f_{min}	f_{avg}
h_001	**46**	**46**	**46**	**46**	**46**	**46**	**46**	**46**	**46**	**46**
h_003	**38**	**38**	**38**	**38**	**38**	**38**	**38**	**38**	**38**	**38**
h_005	**235**	236	**235**	**235**	**235**	**235**	**235**	**235**	**235**	**235**
h_007	78	80.4	**77**	**77**	**77**	**77**	**77**	**77**	**77**	**77**
h_009	**33**	**33**	**33**	**33**	**33**	**33**	**33**	**33**	**33**	**33**
h_011	76	76.8	**73**	**73**	**73**	**73**	**73**	**73**	**73**	**73**
h_013	172	173.8	168	168	**167**	167.2	**167**	167.6	**167**	**167**
h_015	72	73.9	**70**	**70**	**70**	**70**	**70**	**70**	**70**	**70**
h_017	184	185.7	**182**	182.2	**182**	**182**	**182**	**182**	**182**	**182**
h_019	122	124.2	**121**	**121**	**121**	**121**	**121**	**121**	**121**	**121**
h_021	62	62.2	**61**	**61**	**61**	**61**	**61**	**61**	**61**	**61**
h_023	130	130.4	**126**	**126**	**126**	**126**	**126**	**126**	**126**	**126**
h_025	193	194.8	**190**	191	**190**	190.6	**190**	**190**	**190**	190.2
h_027	127	128.7	**124**	**124**	**124**	**124**	**124**	**124**	**124**	**124**
h_029	254	256.9	252	252.2	**251**	**251**	**251**	**251**	**251**	**251**
h_031	192	194	**186**	**186**	**186**	**186**	**186**	**186**	**186**	**186**
h_033	193	195.8	**190**	191	**190**	**190**	**190**	190.2	**190**	**190**
h_035	**3933**	3933.2	**3933**	**3933**	**3933**	**3933**	**3933**	**3933**	**3933**	**3933**
h_037	**3922**	3922.1	**3922**	**3922**	**3922**	**3922**	**3922**	**3922**	**3922**	**3922**
h_039	2469	2473.9	2427	2427.4	2419	2419.6	2427	2429.2	**2416**	**2416.8**
h_041	**4019**	4019.3	**4019**	**4019**	**4019**	**4019**	**4019**	**4019**	**4019**	**4019**
h_043	**3914**	**3914**	**3914**	**3914**	**3914**	**3914**	**3914**	**3914**	**3914**	**3914**
h_045	3940	3941.6	**3937**	**3937.6**	3938	3938	3938	3938.2	**3937**	3937.8
h_047	3889	3891	**3886**	**3886**	3887	3887	3888	3888	**3886**	**3886**
h_049	**3889**	**3889**	**3889**	**3889**	**3889**	**3889**	**3889**	**3889**	**3889**	**3889**
h_051	**3999**	3999.9	**3999**	**3999**	**3999**	**3999**	**3999**	**3999**	**3999**	**3999**
h_053	**4014**	**4014**	**4014**	**4014**	**4014**	**4014**	**4014**	**4014**	**4014**	**4014**
h_055	4561	4565.4	4487	4488.8	4481	4481.6	4493	4494.8	**4478**	**4479.6**
h_057	**3884**	**3884**	**3884**	**3884**	**3884**	**3884**	**3884**	**3884**	**3884**	**3884**
h_059	499	500.8	**496**	496.8	497	497	498	498.4	**496**	**496**
h_061	**3744**	3744.9	**3744**	**3744**	**3744**	**3744**	**3744**	**3744**	**3744**	**3744**
h_063	920	926.6	899	900.6	886	887.4	908	912.6	**881**	**881.6**
h_065	**3683**	**3683**	**3683**	**3683**	**3683**	**3683**	**3683**	**3683**	**3683**	**3683**
h_067	5424	5425.1	**5422**	**5422**	**5422**	5422.8	**5422**	5422.8	**5422**	**5422**
h_069	6312	6314.1	6307	6309	6316	6317.6	6311	6311.4	**6305**	**6306.4**
h_071	**6259**	6259.3	**6259**	**6259**	**6259**	**6259**	**6259**	**6259**	**6259**	**6259**
h_073	9102	9105.8	9083	9084.6	9090	9091.8	9099	9103.8	**9077**	**9077.4**
h_075	10326	10330.5	10142	10143.6	10131	10132.6	10167	10169.6	**10121**	**10124.4**
h_077	1256	1260.4	1220	1223.8	**1190**	1193.6	1241	1247	**1190**	**1191.4**
h_079	10958	10962	10945	10948	10956	10958	10956	10958.8	**10941**	**10942.4**
h_081	21877	21878.5	**21867**	**21867**	**21867**	21868.8	21877	21878.2	**21867**	**21867**
h_083	20631	20633.3	**20628**	**20628**	**20628**	**20628**	**20628**	20628.6	**20628**	**20628**
h_085	12816	12819.4	12814	12817.2	12814	12816	12814	12816.4	**12807**	**12808.4**
h_087	14986	15008.3	14727	14730.4	14658	14667.6	14786	14796	**14655**	**14660.8**
h_089	7235	7238.1	7235	7235.6	7252	7256.6	7234	7237.2	**7228**	**7229**
h_091	12125	12126.3	12112	12115.4	12143	12148.4	12126	12130.4	**12099**	**12099.8**
h_093	21969	21976.6	21953	21956.2	21961	21962.2	21960	21964.4	**21948**	**21950.2**
h_095	23824	23827.3	23817	23819	23814	23816.4	23822	23824.6	**23812**	**23815.6**
h_097	12312	12317.9	12310	12313.6	12325	12340	12320	12324.4	**12275**	**12279.2**
h_099	24867	24871.8	24444	24462.4	24350	24364.4	24552	24571.2	**24320**	**24329.2**
h_101	46488	46490.3	**46473**	**46473**	**46473**	46473.4	46488	46493.4	**46473**	**46473**
h_103	**13303**	**13303**	**13303**	**13303**	**13303**	**13303**	**13303**	**13303**	**13303**	**13303**
h_105	48891	48895.4	**48880**	48880.2	**48880**	48881	48899	48902.2	**48880**	**48880**
h_107	23787	23794.1	23792	23814.4	23806	23820.4	23822	23828.6	**23722**	**23726.6**
h_109	7625	7628.3	7628	7629.4	7656	7658.2	7632	7634.4	**7621**	**7621.6**
h_111	**22658**	**22658**	**22658**	**22658**	**22658**	**22658**	**22658**	**22658**	**22658**	**22658**
h_113	**18574**	**18574**	**18574**	**18574**	**18574**	**18574**	**18574**	**18574**	**18574**	**18574**
h_115	130680	130684	**130675**	**130675**	**130675**	**130675**	**130675**	**130675**	**130675**	**130675**
h_117	41142	41143.4	**41111**	41111.6	**41111**	**41111**	41113	41115.8	**41111**	**41111**
h_119	97583	98009.4	93489	93492.8	**93377**	**93415.4**	94132	94155.2	93427	93488.8
h_121	15480	15483	15478	15482.4	15515	15528	15484	15489.6	**15473**	**15474.4**
h_123	4002	4003.1	**4001**	4003	4004	4006.2	4002	4003.4	4002	**4002.4**
h_125	15019	15031.4	**15018**	**15024**	15042	15056.4	15055	15067.4	15028	15039.2
h_127	29568	29576.4	29555	29562.6	29684	29702	29565	29573.4	**29514**	**29521.6**
h_129	28036	28049.4	28009	**28019**	28145	28156.4	28181	28184.8	**28008**	28023.8
h_131	15740	15743.8	15745	15745	15763	15765	15746	15749	**15738**	**15740.2**
h_133	94003	94061.3	93429	93436	93466	93469.4	93594	93599.4	**93222**	**93226.6**
h_135	28047	28057.5	28076	28082.2	28080	28092.4	28075	28086.2	**27983**	**27988.6**
h_137	28045	28048	28068	28070.6	28070	28079	28071	28088.2	**27968**	**27982.8**
h_139	**15852**	15853.9	15856	15856.8	15861	15865.6	**15852**	15855.2	**15852**	**15853.6**
h_141	57017	57049.5	56892	56901.2	57090	57101.2	56960	56963.2	**56864**	**56870.4**
h_143	30239	30256.8	30228	30235.8	30355	30359.8	30249	30255.4	**30210**	**30213.8**
h_145	30256	30268.6	30240	30243	30359	30369.2	30257	30265	**30226**	**30228.6**
h_147	28717	28726	28732	28743.8	28754	28773.8	28745	28754.8	**28648**	**28655**

Table 4: Experimental results on 100 PACE 2022 instances.

Ins.	SA-FVSP		DiVerSeS		DreyFVS		HustFVS		TRSA	
	f_{min}	f_{avg}	f_{min}	f_{avg}	f_{min}	f_{avg}	f_{min}	f_{avg}	f_{min}	f_{avg}
h_149	91608	91622.9	**91607**	**91607**	**91607**	**91607**	**91607**	91609	**91607**	**91607**
h_151	58048	58115.1	57940	57947.4	58066	58087.4	57983	57993.4	**57920**	**57928**
h_153	30587	30598.5	30561	30562.8	30648	30657.8	30583	30588.4	**30547**	**30551.6**
h_155	101168	101274	100426	100442	100586	100599	100560	100588	**100310**	**100312**
h_157	89694	89707	**89626**	**89626**	**89626**	**89626**	89630	89630.4	**89626**	**89626**
h_159	29226	29247.1	29232	29239	29298	29303.8	29249	29275.2	**29165**	**29172**
h_161	28847	28852	**28799**	**28810.6**	28895	28907.2	28974	29000.4	28850	28872.2
h_163	30922	30931.3	30908	30915.4	30945	30953.8	30919	30924.6	**30901**	**30904.2**
h_165	104271	104501	103535	103541	103701	103718	103664	103674	**103451**	**103462**
h_167	61307	61311.1	61299	**61301.2**	61301	61306.6	**61297**	61304	61301	61309
h_169	29696	29707.5	29685	29690	29754	29761.4	29716	29729.4	**29630**	**29634.6**
h_171	90683	90724.5	**90647**	**90647**	**90647**	**90647**	90652	90653.8	**90647**	**90647**
h_173	31676	31679.5	31696	31697.4	**31674**	**31677.4**	31689	31694.8	31684	31688.4
h_175	100743	100909	**100540**	**100541**	100542	100543	100562	100569	**100540**	100542
h_177	100704	100871	**100466**	**100467**	100469	100469	100489	100493	100468	100468
h_179	95757	95898.5	**95730**	**95730**	**95730**	**95730**	95734	95735.2	**95730**	**95730**
h_181	109963	110153	109354	109374	109362	109369	109402	109417	**109290**	**109300**
h_183	112805	112974	112410	112422	112359	112371	112432	112459	**112344**	**112350**
h_185	29398	29421.8	**29353**	**29372.8**	29427	29448.2	29543	29568	29365	29378.2
h_187	**110074**	**110074**	**110074**	**110074**	**110074**	**110074**	**110074**	**110074**	**110074**	**110074**
h_189	216567	217744	213507	213511	213586	213593	213562	213588	**213495**	**213504**
h_191	**116156**	116157	**116156**	**116156**	**116156**	**116156**	**116156**	**116156**	**116156**	**116156**
h_193	31437	31447.4	31444	31444.4	31468	31476.2	31443	31452.4	**31431**	**31433.6**
h_195	4051	4051.3	4050	4050.4	4051	4052.4	**4049**	4050.4	**4049**	**4049.6**
h_197	8002	8004.6	7999	8001	8002	8003.2	8001	8004	**7998**	**7999.4**
h_199	178898	179336	**178686**	**178686**	**178686**	**178686**	**178686**	**178686**	**178686**	**178686**
#B/#E/#W	75/22/3	85/12/3	47/47/6	53/39/8	51/47/2	58/39/3	58/41/1	66/32/2		
p-value	3.4E-13	2.1E-14	6.8E-08	4.0E-08	3.9E-09	6.1E-10	3.6E-11	1.8E-12		
#best	21	12	52	47	48	41	41	33	92	89
gap(%)	64.28	87.00	11.25	13.09	11.25	12.63	19.70	21.82	0.40	0.57

Table 4 (Continue): Experimental results on 100 PACE 2022 instances.

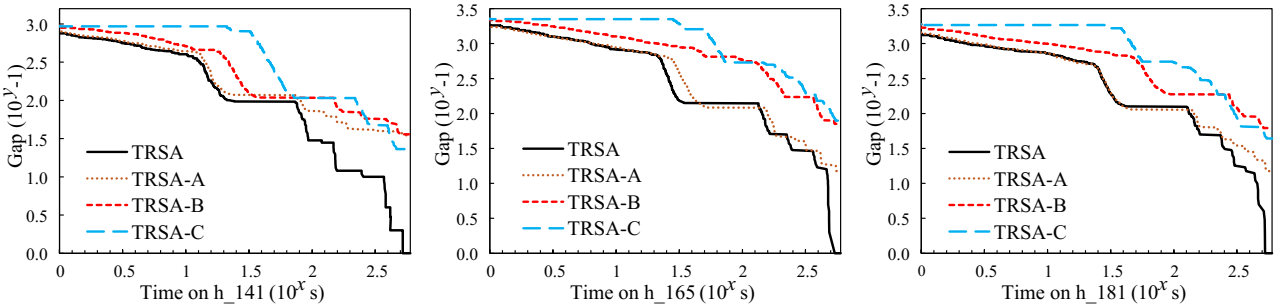

Figure 2: Evolution of the objective value gaps by TRSA, TRSA-A, TRSA-B, and TRSA-C on three largest instances.

We conduct experiments on three hard instances (h_141, h_165, and h_181). Figure 2 depicts the evolution of the objective value gaps of TRSA, TRSA-A, TRSA-B, and TRSA-C as the search proceeds. Each point $(x; y)$ on the curves denotes that the gap between the number of FVS of the current solution and the best known one is y at x seconds.

From Figure 2, one can observe that TRSA is the best performing algorithm among the four versions. Moreover, the improvement rate of TRSA is obviously faster than TRSA-B and TRSA-C. These results indicate that the new reduction rules, the neighborhood reduction based on bidirectional edge elimination, and the threshold-based first-fit strategy all play important roles, and they are essential for the efficacy of TRSA in terms of both effectiveness and efficiency. (The detailed results are available at https://github.com/Zhang-qingyun/DFVSP-TRSA.)

Conclusion

This paper proposes an effective threshold-based responsive simulated annealing (TRSA) algorithm for solving the directed feedback vertex set problem. TRSA adopts several reduction techniques to simplify the problem instances, as well as employing a threshold-based responsive simulated annealing technique. In the local search procedure, TRSA uses three neighborhood acceleration strategies to accelerate the search. The proposed algorithm improves the best known results for totally 53 instances out of the 140 ones used in the literature and in the PACE 2022 competition.

Acknowledgments

We are grateful to the anonymous reviewers for their helpful comments. Qingyun Zhang and Yuming Du contributed equally to this work. This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 72101094 and the Special Project for Knowledge Innovation of Hubei Province under Grant 2022013301015175.

References

Bar-Yehuda, R.; Geiger, D.; Naor, J.; and Roth, R. M. 1994. Approximation algorithms for the vertex feedback set problem with applications to constraint satisfaction and Bayesian inference. In *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, 344–354. Citeseer.

Bar-Yehuda, R.; Geiger, D.; Naor, J.; and Roth, R. M. 1998. Approximation algorithms for the feedback vertex set problem with applications to constraint satisfaction and Bayesian inference. *SIAM Journal on Computing*, 27(4): 942–959.

Chen, Y.; and Hao, J. K. 2015. Iterated responsive threshold search for the quadratic multiple knapsack problem. *Annals of Operations Research*, 226(1): 101–131.

Cutello, V.; and Pappalardo, F. 2015. Targeting the minimum vertex set problem with an enhanced genetic algorithm improved with local search strategies. In *International Conference on Intelligent Computing*, 177–188. Springer.

Festa, P.; Pardalos, P. M.; and Resende, M. G. 2001. Algorithm 815: Fortran subroutines for computing approximate solutions of feedback set problems using GRASP. *ACM Transactions on Mathematical Software (TOMS)*, 27(4): 456–464.

Galinier, P.; Lemamou, E.; and Bouzidi, M. W. 2013. Applying local search to the feedback vertex set problem. *Journal of Heuristics*, 19(5): 797–818.

Hudli, A. V.; and Hudli, R. V. 1994. Finding small feedback vertex sets for VLSI circuits. *Microprocessors and Microsystems*, 18(7): 393–400.

Jain, K.; Hajiaghayi, M. T.; and Talwar, K. 2005. The generalized deadlock resolution problem. In *International Colloquium on Automata, Languages, and Programming*, 853–865. Springer.

Lee, D. H.; and Reddy, S. M. 1990. On determining scan flip-flops in partial-scan designs. In *1990 IEEE International Conference on Computer-Aided Design*, 322–323. IEEE Computer Society.

Lempel, A.; and Cederbaum, I. 1966. Minimum feedback arc and vertex sets of a directed graph. *IEEE Transactions on Circuit Theory*, 13(4): 399–403.

Levy, H.; and Low, D. W. 1988. A contraction algorithm for finding small cycle cutsets. *Journal of Algorithms*, 9(4): 470–493.

Li, C. M.; Fang, Z.; Jiang, H.; and Xu, K. 2018. Incremental upper bound for the maximum clique problem. *INFORMS Journal on Computing*, 30(1): 137–153.

Lin, H. M.; and Jou, J. Y. 2000. On computing the minimum feedback vertex set of a directed graph by contraction operations. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Cystems*, 19(3): 295–307.

Luby, M. 1986. A simple parallel algorithm for the maximal independent set problem. *SIAM Journal on Computing*, 15(4): 1036–1053.

Mugisha, S.; and Zhou, H. 2016. Identifying optimal targets of network attack by belief propagation. *Physical Review E*, 94(1): 012305.

Orenstein, T.; Kohavi, Z.; and Pomeranz, I. 1995. An optimal algorithm for cycle breaking in directed graphs. *Journal of Electronic Testing*, 7(1): 71–81.

Pardalos, P. M.; Qian, T.; and Resende, M. G. 1998. A greedy randomized adaptive search procedure for the feedback vertex set problem. *Journal of Combinatorial Optimization*, 2(4): 399–412.

Qian, T.; Ye, Y.; and Pardalos, P. M. 1996. A pseudo ϵ-approximate algorithm for feedback vertex set. In *State of the Art in Global Optimization*, 341–351. Springer.

Ramanujan, M.; and Szeider, S. 2017. Rigging nearly acyclic tournaments is fixed-parameter tractable. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

Russo, L. M. S.; Castro, D.; Ilic, A.; Romano, P.; and Correia, A. S. D. 2022. Stochastic simulated annealing for directed feedback vertex set. *Applied Soft Computing*, 129: 109607.

Seymour, P. D. 1995. Packing directed circuits fractionally. *Combinatorica*, 15(2): 281–288.

Smith, G.; and Walford, R. 1975. The identification of a minimal feedback vertex set of a directed graph. *IEEE Transactions on Circuits and Systems*, 22(1): 9–15.

Sun, W.; Hao, J. K.; Wu, Z.; Li, W.; and Wu, Q. 2023. Dynamic thresholding search for the feedback vertex set problem. *PeerJ Computer Science*, 9: e1245.

Tang, Z.; Feng, Q.; and Zhong, P. 2017. Nonuniform neighborhood sampling based simulated annealing for the directed feedback vertex set problem. *IEEE Access*, 5: 12353–12363.

Tarjan, R. E. 1972. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2): 146–160.

Yannakakis, M. 1978. Node-and edge-deletion NP-complete problems. In *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing*, 253–264.

Zehavi, M. 2023. Tournament fixing parameterized by feedback vertex set number is FPT. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 5876–5883.

Zhou, H. J. 2016. A spin glass approach to the directed feedback vertex set problem. *Journal of Statistical Mechanics: Theory and Experiment*, 2016(7): 073303.