A Knowledge-based Iterated Local Search for the Weighted Total Domination Problem

Wen Sun [a,b], Chaofan Chen [a], Jin-Kao Hao [c,*], Wenlong Li [a], Qinghua Wu [d], Yuning Chen [e]

[a] *School of Cyber Science and Engineering, Southeast University, 2 Road Southeast University, 211189 Nanjing, China*

[b] *Frontiers Science Center for Mobile Information Communication and Security, Southeast University, 2 Road Southeast University, 211189 Nanjing, China*

[c] *LERIA, Université d'Angers, 2 Boulevard Lavoisier, 49045 Angers, France*

[d] *School of Management, Huazhong University of Science and Technology, 1037 Road Luoyu, 430074 Wuhan, China*

[e] *College of System Engineering, National University of Defense Technology, 410073 Changsha, China*

Information Sciences 665: 120364
https://doi.org/10.1016/j.ins.2024.120364

Abstract

For a simple undirected weighted graph $G = (V, E, w, c)$, the weighted total domination problem is to find a total dominating set S with the minimum weight cost. A total dominating set S is a vertex subset satisfying that for each vertex in V there is at least one neighboring vertex in S. We propose a knowledge-based iterated local search algorithm for this problem that combines a reduction procedure to reduce the input graph, a learning-based initialization to generate high-quality initial solutions and a solution-based iterated local search to conduct intensive solution examination. Experiments on 342 benchmark instances show that the algorithm outperforms state-of-the-art algorithms. In particular, it reports 93 new upper bounds and 249 same results (including 165 known optimal results). The impact of each component of the algorithm is examined.

Keywords: Solution-based search; Learning-based perturbation; Total dominating set.

* Corresponding author.

Nomenclature

BKV	best known value of a WTDP instance
CPLEX	IBM ILOG mathematical program solver
GA	genetic algorithm
GRASP	greedy randomized adaptive search procedure
ILP	integer linear programming model
KILS	knowledge-based iterated local search algorithm
MIP	mixed integer programming model
MTDS	minimum total dominating set problem
SILS	solution-based iterated local search procedure
VNS	variable neighborhood search
WTDP	weighted total domination problem

1 Introduction

Given a simple undirected graph $G = (V, E)$, where V is the set of vertices and $E \subseteq V \times V$ is the set of edges. The Minimum Total Dominating Set (MTDS) problem is to find the total dominating set $S \subseteq V$ with minimum cardinality [3]. A total dominating set S satisfies the condition that every vertex in V, including those in S has at least one neighboring vertex in S.

The MTDS is NP-hard in simple undirected graphs [17], and first appeared in the 1980s [3]. A typical application of the MTDS is the committee formation problem in notification systems [10]. Suppose that we have a large number of candidate committee members and we know for each member the members with whom he/she can communicate. We want to establish a restrictive committee by selecting as few members as possible such that each selected or non-selected member is able to communicate with at least one selected committee member. By considering that V is the set of members, and E is the edge set such that member a is linked to member b if they can communicate with each other. Then the committee formation problem can be conveniently modeled by the MTDS. Other applications of the MTDS include network resource allocation [10], computer network backup [13], and biological network analysis [21].

As a variant of the MTDS, the weighted total domination problem (WTDP) on a vertex weighted and edge weighted graph aims to find a total dom-

inating set S to minimize a weight cost function [20]. Specifically, let $G = (V, E, w, c)$ be an undirected graph with a vertex set V, an edge set E, a vertex weight vector w, and an edge weight vector c. Then, given a vertex subset S of V, its weight cost $f(S)$ consists of 1) the sum weight of all vertices in S; 2) the sum weight of all edges between any two vertices that are both in S; and 3) the sum of the minimum weights of edges between each vertex in $V \setminus S$ and its neighbors in S. The WTDP is to find a vertex subset which minimizes these three sums. Formally, the WTDP can be stated as the following mathematical model [20].

$$\begin{aligned} \text{Minimize} \quad f(S) = &\sum_{i=1}^{|V|} w(v_i)x_i + \sum_{i=1}^{|V|} \sum_{\{v_i,v_j\}\in E} c(v_i,v_j)x_i x_j + \\ &\sum_{i=1}^{|V|} \sum_{\{v_i,v_j\}\in E} \min\{c(v_i,v_j)|x_j = 1\}(1 - x_i) \end{aligned} \tag{1}$$

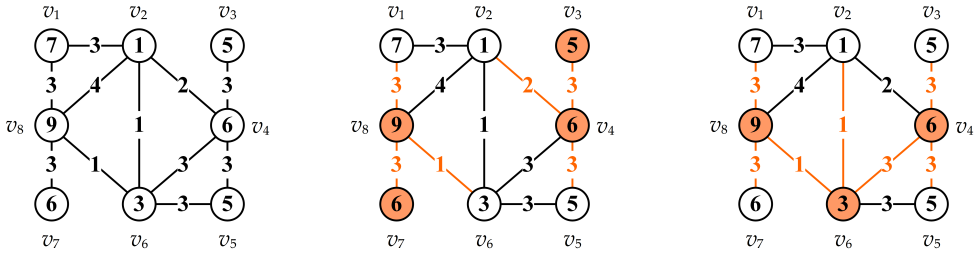
$$\text{subject to} \quad \sum_{\{v_i,v_j\}\in E} x_j \geq 1, \forall i \in \{1,\ldots,|V|\} \tag{2}$$

$$x_i \in \{0,1\}, \forall i \in \{1,\ldots,|V|\} \tag{3}$$

where $|V|$ is the number of vertices; $x_i = 1$ if vertex v_i is in S, otherwise, $x_i = 0$; $\{v_i, v_j\} \in E$ is an edge connecting vertices v_i and v_j; $c(v_i, v_j)$ is the weight of edge $\{v_i, v_j\} \in E$ and $w(v_i)$ is the weight of vertex $v_i \in V$. Equation (1) (objective function) is to minimize the total weight cost. Constraint (2) guarantees that each vertex in V has at least one neighbor vertex in S. Constraint (3) ensures the binary nature for variable x_i.

Figure 1(a) shows a graph G with 8 vertices $\{v_1, \ldots, v_8\}$ whose weights are given in the vertices, and 11 edges whose weights are on the edges respectively. Figure 1(b) shows a total dominating set $S = \{v_3, v_4, v_7, v_8\}$, leading to an objective value of 41, since the sum weight of vertices of S is 26, the sum weight of edges between two vertices of S, i.e., edges $\{v_3, v_4\}$ and $\{v_7, v_8\}$, is 6, and the sum of minimum edge weights from vertices v_1, v_2, v_5 and v_6 to S (i.e., the weights of edges $\{v_1, v_8\}$, $\{v_2, v_4\}$,$\{v_5, v_4\}$ and $\{v_6, v_8\}$) is 9. Figure 1(c) illustrates an optimal total dominating set with a minimum objective value of 35.

The WTDP is also an NP-hard problem [20], as it degenerates to the well-known MTDS problem if we set each vertex weight to 1 and each edge weight to 0. The WTDP is notable for its ability to formulate a variety of real-life applications, such as social network information spreading [12], organization network global decisions [11], and rescue facility placement [4]. For instance, the WTDP can be used to formulate a more realistic version of the committee establishment problem mentioned previously, where the



(a) A given graph G (b) A total dominating set $\{v_3, v_4, v_7, v_8\}$ (c) An optimal total dominating set $\{v_4, v_6, v_8\}$

Fig. 1. A graph, a total dominating set, and an optimal total dominating set (the vertices and edges involved by the objective function are in orange, and the other vertices and edges are not colored).

weight of each vertex represents the cost of hiring the member, and the weight of each edge represents the communication cost.

By reviewing the existing literature (see Section 2), we find that there are few practical algorithms that can effectively solve the WTDP. In particular, the exact algorithms in [1, 20] could only be used on instances with up to $|V| = 500$ vertices due to the inevitable exponential time complexity. To address such instances, computationally efficient heuristic algorithms have been developed to yield high-quality solutions in reasonable computation time. The heuristic algorithms in [1, 12] show greater success in solving instances with up to $|V| = 1000$, but still lack stability on large instances with at least 500 vertices. However, WTDP instances derived from real-world applications can be much larger in size. Thus, there is a need to develop more effective algorithms capable of finding satisfactory solutions for large WTDP instances.

Recent studies have shown that the integration of knowledge-based techniques and conventional heuristic approaches can yield exceptional performance for various combinatorial optimization problems [6, 26, 28]. These studies motivate us to investigate hybrid approaches to take advantage of both the powerful local optimization and the knowledge-based strategies. In this study, we introduce the first knowledge-based iterated local search algorithm (KILS), which integrates a knowledge-based mechanism into the popular iterated local search algorithm framework. We summarize the contributions as follows.

We propose a two-phase local search (see the flow-chart in Section 4.1) that integrates a remove-insert neighborhood to ensure an effective exploitation in the first phase, and an adaptive penalty-based fitness function to guide the local search process in the second phase. The algorithm uses historical information learned from the discovered local optimal solutions to generate good-quality initial solutions, and calls for probability learning to guide

its perturbation procedure. The algorithm also uses a fast neighborhood evaluation to shorten the computation time in both phases.

We evaluate the proposed algorithm on a total of 315 commonly used benchmark instances, and report 71 new record results (improved upper bounds) and 244 equally best results including 160 known optimal results. Moreover, the proposed algorithm achieves the previous best results in a fraction ($< 10\%$) of the time required by the best existing algorithm in the literature. In addition, we show 22 record-breaking best solutions out of the 27 real-world benchmark instances.

The rest of the paper is organized as follows. Section 2 provides a literature review on exact and heuristic algorithms for the WTDP. Section 3 presents some basic definitions. Section 4 presents the proposed algorithm. Section 5 reports comparative results. Section 6 analyzes the impacts of key components of the proposed algorithm. Conclusions and future work are discussed in the final section.

2 Literature review

This section provides an overview of both exact algorithms and heuristic algorithms for the WTDP, while also addressing current research gaps in this area.

2.1 Exact algorithms

Ma et al. [20] (2019) presented three integer linear programming (ILP) models, which are used to solve 45 small instances ($20 \leq |V| \leq 100$) with a time limit of 1800 seconds using the mathematical programming solver CPLEX (version 12.8). This approach is only applicable to instances of limited sizes (with no more than 50 vertices).

Álvarez-Miranda and Sinnl [1] (2021) investigated two mixed integer programming (MIP) models and generated 135 medium instances ($75 \leq |V| \leq 125$). The two MIP models solved by the CPLEX solver (version 12.9) respectively reported 133 and 148 optimal solutions for a total of 180 small and medium instances with a time limit of 1800 seconds. Compared to ILP [20], the MIP models obtained 15 better results and 30 equal results out of the 45 small instances.

Kapunac et al. [12] (2022) used the CPLEX solver (version 20.1) to solve the best MIP model of [1] (denoted by re-MIP) on a more powerful computing

platform with an Intel Core i9-9900KF processor (3.6 GHz and 6 GB RAM against the 2.5 GHz processor in [1]). With re-MIP, [12] obtained better results than those reported in [1] (160 vs. 148 optimal results) for the 180 small and medium instances under the same time limit as in [20]. Therefore, we use the results of [12] as our references of exact algorithms.

2.2 Heuristic algorithms

Due to their high complexity, exact algorithms do not scale well for large instances and typically become unpractical for graphs with more than 125 vertices. Thus, heuristic algorithms are preferred to find suboptimal solutions for large instances within a reasonable computation time.

Álvarez-Miranda and Sinnl [1] (2021) proposed a greedy randomized adaptive search procedure (GRASP) and a genetic algorithm (GA). In addition to its greedy randomized solution construction procedure based on the remove operator, GRASP is characterized by its descent search procedure which combines the insert operator and the remove operator. The GA incorporates a crossover operator, a random mutation, and the descent search. The crossover operator produces an offspring by combining two parents and iteratively removing vertices from the offspring with a greedy randomized strategy. The offspring is then mutated by randomly removing several vertices. The descent search used to improve the offspring is the same as GRASP. GRASP and GA were tested on the instances with $|V| \leq 125$. Compared to GRASP, GA obtained 85 better results and 50 equal results on the 135 medium instances, indicating that GA dominated GRASP under the same time limit.

Kapunac et al. [12] (2022) introduced a variable neighborhood search (VNS) algorithm, which alternates between a variable neighborhood descent phase and a shaking phase. An initial solution constructive procedure is used to generate a random solution. The variable neighborhood descent phase relies on two types of basic moves (flip operator and 1-swap operator) to attain a local optimum. The shaking phase carries out random perturbations to produce new starting solutions. Besides, 135 large instances ($250 \leq |V| \leq 1000$) are generated in [12]. In addition, Kapunac et al. [12] re-implemented the GA algorithm in [1], which is called re-GA in this paper. Compared to re-GA, VNS obtained 100 better results, 141 equal results, and 29 worse results on the 270 medium and large instances. Besides, to reach its best solutions, VNS consumed less time than any previous algorithm.

Therefore, re-GA and VNS are the most advanced existing heuristic algorithms for solving the WTDP on the three sets of 315 benchmark in-

stances including small instances ($20 \leq |V| \leq 100$), medium instances ($75 \leq |V| \leq 125$) and large instances ($250 \leq |V| \leq 1000$). We use these two algorithms as our main references of heuristic algorithms for our computational study on these 315 benchmark instances. Besides, VNS was also run on 27 real-world instances ($1912 \leq |V| \leq 81306$) from the Stanford large network dataset collection [18] on the information spreading problem. However, we find that VNS reports incorrect weight cost of WTDP for problem instances containing self-loops and multiple edges. Therefore, when we test these real-world instances, we remove the self-loops and multiple edges to ensure that they are simple graphs.

Table 1 summarizes the methods discussed in this section. Column 1 shows the authors and publication year of the literature. Column 2 indicates the algorithm framework of the corresponding literature. Column 3 shows the move operators used by the algorithm (not applicable to exact algorithms). Column 4 indicates the groups of the instances on which each algorithm has run.

Table 1
Representative exact and heuristic algorithms for the WTDP.

Literature	Framework	Operators	Instance groups
Exact algorithms			
Ma et al. (2019) [20]	ILP	-	small
Álvarez-Miranda and Sinnl (2021) [1]	MIP	-	small, medium
Kapunac et al. (2022) [12]	re-MIP	-	small, medium, large
Heuristic methods			
Álvarez-Miranda and Sinnl (2021) [1]	GRASP	remove, insert	small, medium
Álvarez-Miranda and Sinnl (2021) [1]	GA	remove, insert	medium
Kapunac et al. (2022) [12]	re-GA	remove, insert	medium, large
Kapunac et al. (2022) [12]	VNS	flip, 1-swap	small, medium, large, real-world

2.3 Research gaps

As discussed in the previous sections, considerable efforts have been made to develop various solution methods for the WTDP. Nevertheless, there is still room for improvement in terms of more effective solution techniques, considering the following research gaps: (1) Recently, approaches combining knowledge-based techniques and optimization methods have shown promising results for various combinatorial optimization problems [6, 26, 28]. However, the use of knowledge-based approaches in solving the WTDP is still missing; (2) The WTDP possesses distinctive features that can be used to guide dedicated search algorithms. By effectively learning these features, algorithm performance can be significantly enhanced. For instance, viewing the WTDP as a strongly constrained problem allows for the ex-

ploration of tunneling methods that examine both feasible and infeasible regions to find high-quality solutions. However, these methods have not yet been studied specifically for solving the WTDP.

To address these gaps, this study introduces the first knowledge-based iterated local search for the WTDP. The primary objective of this work is twofold. First, given the importance of WTDP, we aim to enrich the existing literature by proposing a novel solution approach capable of achieving high-quality solutions with reduced computing effort. Second, we aim to make methodological contributions to the integration of knowledge-based techniques with optimization methods and to the improvement of the popular iterated local search method for general combinatorial optimization.

3 Definitions

For a precise description of the KILS algorithm (see Section 4), we first introduce the following definitions, where $G_0 = (V_0, E_0, w_0, c_0)$ is a given undirected weighted graph.

Definition 1 : A *crucial vertex* is a vertex that must be contained by all total dominating sets. A *redundant vertex* is a vertex that is out of any optimal solution. The remaining vertices are *unknown vertices*.

We use S_c to denote the set of *crucial vertices*, S_r the set of *redundant vertices*, and $U = V_0 \setminus \{S_c \cup S_r\}$ the set of *unknown vertices*.

Figure 2 presents a graph G_0 with 10 vertices $\{v_1, \ldots, v_{10}\}$. According to Rule 1 (see Appendix A), v_4 and v_8, which have a neighbor of degree 1, are crucial vertices, and $S_c = \{v_4, v_8\}$ (in violet). According to Rule 2 (see Appendix A), vertices v_9 and v_{10} are redundant vertices because they have a degree of 1 and are neighbors of crucial vertices, but don't have the minimum vertex weight. Thus, $S_r = \{v_9, v_{10}\}$ (in grey). The set of unknown vertices $U = \{v_1, v_2, v_3, v_5, v_6, v_7\}$ (in blue).

Definition 2: For a WTDP instance, a *feasible solution* is any candidate subset satisfying the constraint that it is a total dominating set, i.e., each vertex in V_0 has at least one neighbor vertex in this subset. Otherwise, it is an *infeasible solution*.

Definition 3: For a given WTDP instance, its *feasible solution space* Ω contains all feasible solutions. Formally, Ω is given by

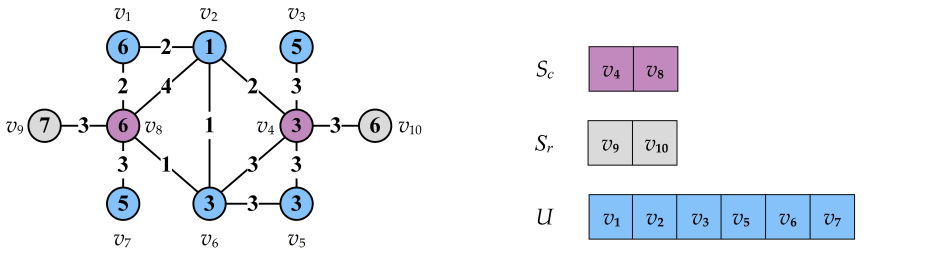


Fig. 2. Illustration of crucial (violet), redundant (grey) and unknown (blue) vertices.

$$\Omega = \{(x_1, x_2, \ldots, x_{|V_0|}) | \sum_{\{v_i, v_j\} \in E_0} x_j \geq 1, x_i \in \{0,1\}, 1 \leq i, j \leq |V_0|\} \quad (4)$$

Definition 4: For a WTDP instance, its *feasible and infeasible solution space* Ω^+ includes all feasible and infeasible solutions, i.e., all non-empty subsets of V_0, which is described as below.

$$\Omega^+ = \{(x_1, x_2, \ldots, x_{|V_0|}) | \sum_{i=1}^{|V_0|} x_i > 0, x_i \in \{0,1\}, 1 \leq i \leq |V_0|\} \quad (5)$$

Definition 1 presents three types of vertices that are used in the reduction procedure of the KILS algorithm. Definitions 2-4 are needed to illustrate the second phase of the local search procedure.

4 The knowledge-based iterated local search algorithm

In this section, we first introduce the general procedure of the proposed KILS algorithm and then present each of its components.

4.1 General approach

Algorithm 1 and Figure 3 show the general architecture of the KILS algorithm. At the beginning of the algorithm, a reduction procedure is used to preprocess the given input graph (line 7). Then, a number of iterations follow, in each of which the learning-based initialization procedure (line 11)

generates an initial solution S and the solution-based iterated local search procedure (line 12) improves the solution S until the stopping condition is reached. The recorded overall best solution (S^*) is updated whenever necessary, and is output at the end of the KILS algorithm.

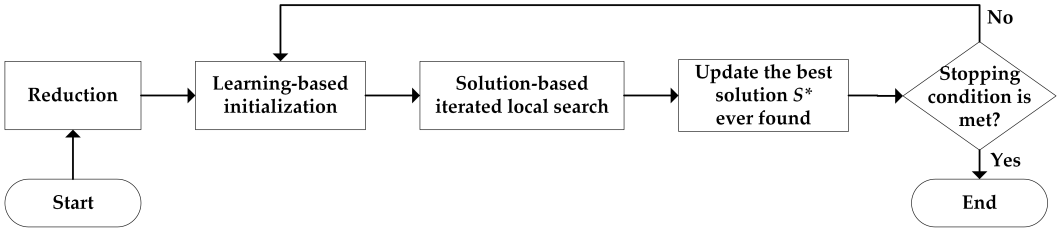


Fig. 3. Flowchart of the proposed KILS algorithm.

The proposed KILS algorithm necessitates some parameters, including hash vectors H_k and hash functions h_k ($k = 1, ..., 6$) to determine tabu status in solution-based iterated local search (Section 4.3.3); greediness ratio ϵ to balance the quality and randomness of initial solutions (Section 4.2); historical information vector P to generate informed initial solutions (Section 4.2); perturbation probability vector Q to obtain promising new starting solutions (Section 4.3.2); penalty coefficient α to control the importance given to the penalty function (Section 4.3.1.2).

Algorithm 1 The main framework of the KILS algorithm

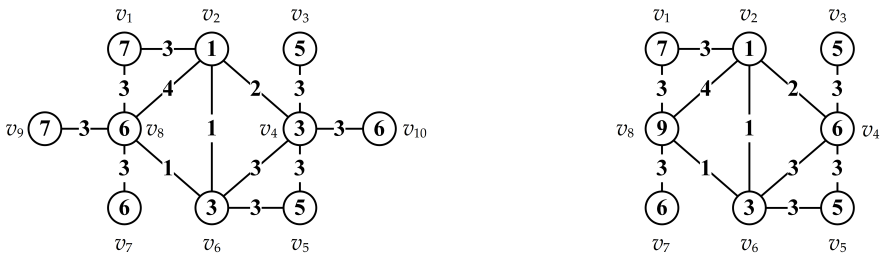
Input: Given graph $G_0 = (V_0, E_0, w_0, c_0)$; hash vectors H_k with a length of L ($k = 1, \ldots, 6$); hash functions h_k ($k = 1, \ldots, 6$); greediness ratio ϵ.
Output: The overall best feasible solution S^* ever found.

1: **for** $l \leftarrow 0$ to $L - 1$ **do**
2: $H_1[l] \leftarrow 0$; $H_2[l] \leftarrow 0$; $H_3[l] \leftarrow 0$; /*Initialization of hash vectors */
3: $H_4[l] \leftarrow 0$; $H_5[l] \leftarrow 0$; $H_6[l] \leftarrow 0$;
4: **end for**
5: Initialize the historical information vector P /*See Section 4.2*/
6: Initialize the perturbation probability vector Q /*See Section 4.3.2*/
7: $G \leftarrow Reduction(G_0)$
8: $S^* \leftarrow \emptyset$ /*Initialize the overall best feasible solution S^**/
9: **while** stopping condition is not met **do**
10: $\alpha \leftarrow 2$ /*The penalty coefficient α is initially set as 2*/
11: $S \leftarrow Learning\_Based\_Initialization(G, P, \epsilon)$ /* Generate an initial solution S*/
12: $(S, P, Q) \leftarrow Solution\_Based\_Iterated\_Local\_Search(G, S, H_k, h_k, \alpha, P, Q)$ /*Improve the initial solution S, $k = 1, \ldots, 6$*/
13: **if** $f(S) < f(S^*)$ **then**
14: $S^* \leftarrow S$ /*Update the overall best feasible solution S^* ever found*/
15: **end if**
16: **end while**
17: **return** S^*

The reduction procedure is designed to reduce the search space (line 7, Algorithm 1). Specifically, before the learning-based initialization procedure,

we identify the set of crucial vertices S_c and the set of redundant vertices S_r according to the rules presented in Appendix A. Then we simplify the given graph as follows: 1) all vertices in S_c are put in S; 2) all vertices in S_r and their connected edges are deleted; 3) the weights of the edges between the deleted vertices and their neighbor vertices are added to the latter, to ensure that the objective function remains unchanged after the reduction. Thus, we reduce the input graph $G_0 = (V_0, E_0, w_0, c_0)$ to the reduced graph $G = (V, E, w, c)$, where $V = V_0 \setminus S_r$. Throughout the whole KILS algorithm, none of the vertices in S_c will be traversed or moved during the subsequent neighborhood search.

To explain this process, we consider the graph in Figure 2 with 10 vertices $V_0 = \{v_1, \ldots, v_{10}\}$. Figure 4(a) depicts the input graph $G_0 = (V_0, E_0, w_0, c_0)$ with the set of crucial vertices $S_c = \{v_4, v_8\}$, the set of redundant vertices $S_r = \{v_9, v_{10}\}$ according to the Appendix A. After reducing the input graph $G_0 = (V_0, E_0, w_0, c_0)$, we obtain the reduced graph $G = (V, E, w, c)$ by adding the weight of edge $\{v_4, v_{10}\}$ to v_4 and the weight of edge $\{v_8, v_9\}$ to v_8 as shown in Figure 4(b), where $V = \{v_1, \ldots, v_8\}$, $w(v_4) = w_0(v_4) + c_0(v_4, v_{10})$, and $w(v_8) = w_0(v_8) + c_0(v_8, v_9)$.



(a) An input graph $G_0 = (V_0, E_0, w_0, c_0)$ (b) A reduced graph $G = (V, E, w, c)$

Fig. 4. Illustration of reduction procedure.

4.2 Learning-based initialization

Learning-based initialization aims to generate an initial feasible solution for the reduced graph (Algorithm 2). It initializes the current solution S with V, and iteratively removes one vertex from S according to the historical information vector $P = \{P[1], P[2], \ldots, P[|V|]\}$ (lines 3-15, Algorithm 2). $P[i]$ records the probability of v_i being removed from S, and is initially set to 0.5, $\forall i \in \{1, 2, \ldots, |V|\}$ (line 5, Algorithm 1). Specifically, when removing one vertex from S each time: 1) the vertex $v_i \in CL$ with the highest value $P[i]$ in P is chosen with a probability of ϵ, or a random vertex $v_i \in CL$ is selected (lines 5-9, Algorithm 2), where CL represents the set of unchosen vertices; 2) the chosen v_i is removed from CL (line 10, Algorithm 2); 3) v_i is removed from S if $S \setminus \{v_i\}$ is still a feasible solution and $P[i]$ is higher

than a random probability (lines 12-14, Algorithm 2), or v_i is ignored. The initialization procedure terminates when CL becomes empty.

Algorithm 2 Learning-based initialization

Input: Reduced graph $G = (V, E, w, c)$; historical information vector P; greediness ratio ϵ
Output: A feasible solution S

1: $S \leftarrow V$ /*Initialize the current solution S */
2: $CL \leftarrow S$ /*Initialize the set of unchosen vertices CL*/
3: **while** $CL \neq \emptyset$ **do**
4: /*Choose a vertex v_i */
5: **if** $rand(0,1) < \epsilon$ **then**
6: Choose a vertex $v_i \in CL$ with the largest value $P[i]$ in vector P
7: **else**
8: Randomly choose a vertex $v_i \in CL$
9: **end if**
10: $CL \leftarrow CL \setminus \{v_i\}$ /*Remove the chosen vertex v_i from CL*/
11: /*Remove the chosen vertex v_i from S*/
12: **if** $S \setminus \{v_i\}$ is a feasible solution and $rand(0,1) < P[i]$ **then**
13: $S \leftarrow S \setminus \{v_i\}$
14: **end if**
15: **end while**
16: **return** S

To explain this process, we consider the reduced graph $G = (V, E, w, c)$ in Figure 4(b) (with vertices $V = \{v_1, v_2, \ldots, v_8\}$). Figure 5 provides an example in which $CL = S = V$ initially. First, according to the greediness ratio ϵ and vector P, the vertex v_2 with the largest value $P[2]$ in vector P is chosen and removed from CL and S. Second, v_6, v_1 and v_5 are successively chosen and removed from CL and S. Then, the vertices v_3, v_4, v_7 and v_8 are successively removed from CL, but can not be removed from S. Finally, a feasible solution $S = \{v_3, v_4, v_7, v_8\}$ is obtained.

This initialization procedure thus provides a feasible solution of reasonable quality, which will be further improved by the subsequent solution-based iterated local search procedure.

4.3 Solution-based iterated local search procedure

Beginning with the solution that comes from the learning-based initialization procedure (Section 4.2), the KILS algorithm designs the solution-based iterated local search procedure (SILS) (Algorithm 3) to improve the solution. Specifically, SILS alternates the two-phase local search (line 4) and the learning-based perturbation (line 13) until its best solution cannot be improved for ω consecutive iterations (ω stands for the depth of SILS). More-

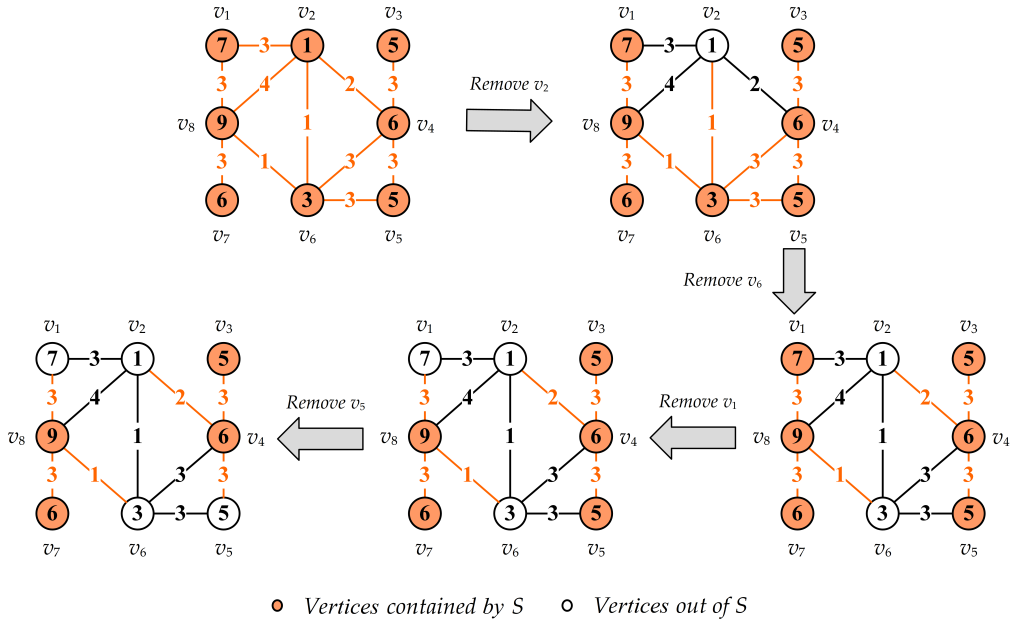


Fig. 5. Generate an initial feasible solution by iteratively removing vertices with the historical information vector (the vertices and edges involved by the objective function of S are in orange and the other vertices and edges are not colored).

over, the historical information vector P, perturbation probability vector Q, the best feasible solution S^{best} found during SILS, and the consecutive non-improvement counter $NoImprove$ are updated (lines 6-9), if the best feasible solution S^b found during the two-phase local search is better than S^{best}.

4.3.1 Two-phase local search

As a general method, two-phase search has been successfully adopted to solve various difficult optimization problems such as the multidimensional knapsack problem [15, 27], quadratic knapsack problem [2], and colored traveling salesman problem [8, 9].

The two-phase local search as described in Algorithm 4 aims to improve the input solution. If the input solution is feasible, the first phase, also called the solution-based descent phase, performs an intensive exploitation of candidate solutions to quickly find a local optimal solution (lines 2-4); otherwise, this phase is skipped. Then, the second phase, also called the solution-based feasible and infeasible phase, is invoked to explore larger search areas starting from the local optimal solution (line 5).

4.3.1.1 Solution-based descent phase The solution-based descent phase explores the feasible search space Ω only and tries to find a local optimal feasible solution rapidly. This phase iteratively replaces the current solution

Algorithm 3 The solution-based iterated local search

Input: Reduced graph $G = (V, E, w, c)$; the current solution S; hash vectors H_k with a length of L, $k = 1, \ldots, 6$; hash functions h_k, $k = 1, \ldots, 6$; penalty coefficient α; probability vectors P and Q.

Output: The best feasible solution S^{best} found during SILS; updated probability vectors P.

$S^{best} \leftarrow \emptyset$ /*Initialize the best feasible solution S^{best} found during SILS*/
$NoImprove \leftarrow 0$ /*Indicate the consecutive iterations where S^{best} is not updated*/
while $NoImprove < \omega$ **do**
  $(S, S^b, \alpha) \leftarrow TwoPhase\_LocalSearch(S, H_k, h_k, \alpha)$ /*Update the current solution S and the best feasible solution S^b found during two-phase local search*/
  if $f(S^b) < f(S^{best})$ **then**
    Update historical information vector P with Equations (17)-(18)
    Update perturbation probability vector Q with Equation (19)
    $S^{best} \leftarrow S^b$ /*Update the best feasible solution S^{best} found during SILS*/
    $NoImprove \leftarrow 0$
  else
    $NoImprove \leftarrow NoImprove + 1$
  end if
  $S \leftarrow Learning\_Based\_Perturbation(G, S, Q)$
end while
return (S^{best}, P, Q)

Algorithm 4 Two-phase local search

Input: The current solution S; hash vectors H_k with a length of L, $k = 1, \ldots, 6$; hash functions h_k, $k = 1, \ldots, 6$; penalty coefficient α.

Output: The current solution S; the best feasible solution S^b found during two-phase local search; penalty coefficient α.

$S^b \leftarrow \emptyset$ /*Initialize the best feasible solution S^b found during two-phase local search*/
if $F(S) = f(S)$ **then**
  $(S, H_k) \leftarrow Solution\_Based\_Descent(S, H_k, h_k)$ /*1st-phase: find a local optimal feasible solution, $k = 1, 2, 3$, Section 4.3.1.1*/
end if
$(S, S^b, H_k, \alpha) \leftarrow Solution\_Based\_Feasible\_Infeasible(S, S^b, H_k, h_k, \alpha)$ /*2nd-phase: find a local optimal feasible or infeasible solution, $k = 4, 5, 6$, Section 4.3.1.2*/
return (S, S^b, α)

S with a best admissible neighborhood solution S' (lines 5-10, Algorithm 5). We design a remove-insert operator to generate neighborhood solutions, and hash vectors $H_k, k = 1, 2, 3$ to manage the admissibility of each neighborhood solution (Section 4.3.3). This process is repeated until no improvement is possible.

Algorithm 5 The solution-based descent search

Input: The current solution S; hash vectors H_k, $k = 1, 2, 3$; hash functions h_k, $k = 1, 2, 3$.

Output: The current solution S; hash vectors H_k, $k = 1, 2, 3$.

```
Calculate the neighborhood N_1(S)        /*N_1(S) is defined in Equation (6)*/
NoImprove ← 0                            /*Indicate whether S has been improved*/
while NoImprove = 0 do
    NoImprove ← 1
    Select a best admissible neighborhood solution S' ∈ N_1(S) in terms of objective function f with a remove-insert operator that satisfies H_1(h_1(S')) ∧ H_2(h_2(S')) ∧ H_3(h_3(S')) = 0
    if f(S') < f(S) then
        S ← S'                           /*Update the current solution S */
        H_1(h_1(S)) ← 1; H_2(h_2(S)) ← 1; H_3(h_3(S)) ← 1 /*Update the hash vectors with S*/
        NoImprove ← 0
    end if
end while
return (S, H_k)                          /*k = 1, 2, 3*/
```

Remove-insert operator: This operator first removes a vertex v_i from S. Then, for each v_i's neighbor v_l that has no neighbor vertex in S, its neighbor $v_j \in V \setminus S$ that has the minimum incremental objective value (Equation (10)) will be inserted to S (ties are broken randomly). Let A be the set of eligible vertices for insertion after removing v_i, and let $remove\text{-}insert(v_i, A)$ denote the operation that removes vertex v_i from S and inserts the vertices from A. The neighborhood N_1 induced by this move operator contains all possible feasible solutions obtained by applying "remove-insert" to S.

$$N_1(S) = \{S \oplus remove\text{-}insert(v_i, A) |\ v_i \in S \text{ and } v_i \notin S_c\} \tag{6}$$

The size of N_1 is limited by $O(|S| \times Max_{deg}^2)$, where Max_{deg} is the maximum vertex degree in the graph G.

Fast Neighborhood Evaluation Technique: The remove-insert operator can be regarded as a combination of a remove operator followed by several insert operators. $\Delta f(remove\text{-}insert(v_i, A))$ denotes the incremental objective value after applying $remove\text{-}insert(v_i, A)$ and can be calculated by

$$\Delta f(remove\text{-}insert(v_i, A)) = \Delta f(remove(v_i)) + \sum_{v_j \in A} \Delta f(insert(v_j)) \tag{7}$$

where $remove(v_i)$ represents the operation that removes vertex v_i from the

current solution S and $insert(v_j)$ represents the operation that inserts vertex v_j into S.

To fasten the calculation of $\Delta f(remove(v_i))$, two $|V|$-vectors M_1 and M_2 are adopted. Specifically, M_1 and M_2 are maintained in which the entry $M_1[v_i]$ $(v_i \in V)$ is the edge with the minimum weight in $E_S(v_i)$ and $M_2[v_i]$ $(v_i \in V)$ is the edge with the minimum weight in $E_S(v_i)$ except $M_1[v_i]$, i.e., $M_1[v_i] = \arg\min\{c(v_i, v_j)|\ \{v_i, v_j\} \in E_S(v_i)\}$ and $M_2[v_i] = \arg\min\{c(v_i, v_j)|\ \{v_i, v_j\} \in E_S(v_i) \setminus \{M_1[v_i]\}\}$. With these data structures, we can efficiently calculate $\Delta f(remove(v_i))$ as follows.

$$\begin{aligned}\Delta f(remove(v_i)) = -w(v_i) - \sum_{\{v_i,v_j\}\in E_S(v_i)} c(v_i, v_j) + \\ c(M_1[v_i]) + \sum_{\{v_i,v_j\}\in E_{V\setminus S}(v_i)} \gamma_{remove}(v_i, v_j)\end{aligned} \tag{8}$$

where $E_S(v_i) = \{\{v_i, v_j\}|\{v_i, v_j\} \in E, v_j \in S\}$ is the set of edges between vertex v_i and vertices in S; $E_{V\setminus S}(v_i) = \{\{v_i, v_j\}|\{v_i, v_j\} \in E, v_j \in V \setminus S\}$ is the set of edges between vertex v_i and vertices in $V \setminus S$, and the $\gamma_{remove}(v_i, v_j)$ for each $\{v_i, v_j\} \in E_{V\setminus S}(v_i)$ can be given by

$$\gamma_{remove}(v_i, v_j) = \begin{cases} -c(v_i, v_j), & if\ E_S(v_j) = \{\{v_i, v_j\}\}; \\ -c(v_i, v_j) + c(M_2[v_j]), & if\ E_S(v_j) \neq \{\{v_i, v_j\}\},\ \{v_i, v_j\} = M_1[v_j]; \\ 0, & otherwise. \end{cases} \tag{9}$$

Then, vectors M_1 and M_2 are accordingly updated as follows: 1) $M_1[v_j] \leftarrow M_2[v_j]$, $M_2[v_j] \leftarrow \arg\min\{c(v_j, v_m)|\{v_j, v_m\} \in E_S(v_j) \setminus \{M_2[v_j]\}\}$ for each neighbor vertex v_j of v_i satisfying $M_1[v_j] = \{v_i, v_j\}$; 2) $M_2[v_t] \leftarrow \arg\min\{c(v_t, v_m)|\{v_t, v_m\} \in E_S(v_t) \setminus \{M_2[v_t]\}\}$ for each neighbor vertex v_t of v_i satisfying $M_2[v_t] = \{v_i, v_t\}$. Finally, the edge $\{v_i, v_n\}$ in $E_S(v_n)$ will be moved to $E_{V\setminus S}(v_n)$ for each neighbor vertex v_n of v_i. Clearly, updating M_1 and M_2 can be done in $O(Max^2_{deg})$.

Similarly, the incremental objective value of inserting $v_j \in A$ into the solution S is computed by

$$\Delta f(insert(v_j)) = w(v_j) + \sum_{\{v_j,v_m\}\in E_S(v_j)} c(v_j,v_m) - c(M_1[v_j]) + \sum_{\{v_j,v_m\}\in E_{V\setminus S}(v_j)} \gamma_{insert}(v_j,v_m) \tag{10}$$

where $\gamma_{insert}(v_j,v_m)$, $(v_j,v_m) \in E_{V\setminus S}(v_j)$, is given by

$$\gamma_{insert}(v_j,v_m) = \begin{cases} c(v_j,v_m), & if\ E_S(v_m) = \emptyset; \\ c(v_j,v_m) - c(M_1[v_m]), & if\ E_S(v_m) \neq \emptyset,\ c(v_j,v_m) < c(M_1[v_m]); \\ 0, & otherwise. \end{cases} \tag{11}$$

When a remove-insert move is performed, the affected data structures M_1 and M_2 can be firstly updated according to the updating rule of the "remove" move. Then, vectors M_1 and M_2 will be accordingly updated as below if an insert operator (i.e., $insert(v_j)$) is performed: 1) $M_2[v_m] \leftarrow M_1[v_m]$, $M_1[v_m] \leftarrow \{v_j, v_m\}$, for each neighbor vertex v_m of vertex v_j satisfying $c(v_j,v_m) < c(M_1[v_m])$; 2) $M_2[v_t] \leftarrow \{v_j, v_t\}$, for each neighbor vertex v_t of vertex v_j satisfying $c(M_1[v_t]) \leq c(v_j,v_t) < c(M_2[v_t])$. The edge $\{v_j, v_n\}$ in $E_{V\setminus S}(v_n)$ will be moved into $E_S(v_n)$ for each neighbor vertex v_n of v_j. Vectors M_1 and M_2 can be updated in $O(Max^2_{deg})$.

Figure 6 illustrates the solution-based descent phase operating on the given graph. Suppose that $S = \{v_3, v_4, v_7, v_8\}$ is the local optimal feasible solution with the objective value of 41 obtained from the learning-based initialization algorithm. When removing v_7 and inserting $A = \{v_6\}$, an improved neighborhood solution $S \oplus remove\text{-}insert(v_7, \{v_6\})$ can be obtained. The incremental objective value is the sum of $\Delta f(remove(v_7)) = -6$ and the $\sum_{u\in A} \Delta f(insert(u)) = \Delta f(insert(v_6)) = 5$, that is, -1, and thus the objective value of the improved solution is 40.

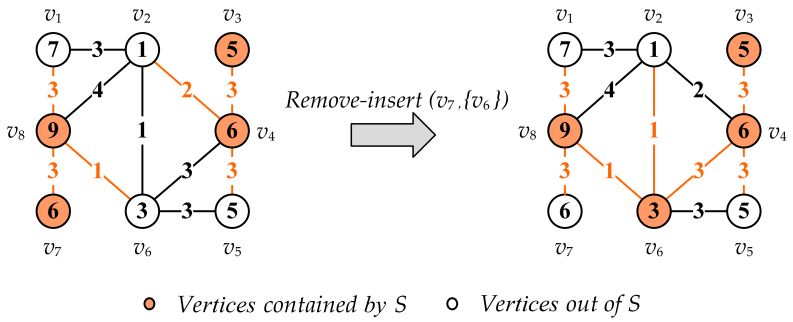


Fig. 6. Illustration of the solution-based descent search procedure (the vertices and edges involved by the objective function $f(S)$ are in orange and the other vertices and edges are not colored).

4.3.1.2 Solution-based feasible and infeasible phase

When the solution-based descent search cannot find still better solution within the feasible space Ω, or when the current solution of two-phase local search from the perturbation procedure is infeasible, the KILS algorithm invokes the second phase to explore an enlarged space Ω^+ including both feasible and infeasible solutions (line 5, Algorithm 4).

This phase also iteratively substitutes the current solution S with the best admissible neighborhood solution S' in terms of the extended evaluation function F (see Equation (13) below), which combines the objective function f with an adaptive penalty function ρ. Whether each solution is admissible is also managed by hash vectors H_k $(k = 4, 5, 6)$ (lines 5-12, Algorithm 6). The second phase stops when the solution cannot be improved. To generate neighborhood solutions, we design a flip operator.

Algorithm 6 The solution-based feasible and infeasible search

Input: The current solution S; the best feasible solution S^b found during two-phase local search; hash vectors H_4, H_5, H_6; hash functions h_4, h_5, h_6; penalty coefficient α.

Output: The current solution S; updated best feasible solution S^b found during two-phase local search; hash vectors H_k, $k = 4, 5, 6$; penalty coefficient α.

1: Calculate the neighborhood $N_2(S)$ /*$N_2(S)$ is defined in Equation (12)*/
2: $NoImprove \leftarrow 0$ /*Indicate whether S has been improved*/
3: **while** $NoImprove = 0$ **do**
4: $NoImprove \leftarrow 1$
5: Select a best admissible neighborhood solution $S' \in N_2(S)$ in terms of extended objective function F with a flip operator that satisfies $H_4(h_4(S')) \wedge H_5(h_5(S')) \wedge H_6(h_6(S')) = 0$
6: **if** $F(S') < F(S)$ **then**
7: $S \leftarrow S'$ /*Update the current solution S*/
8: Calculate the $N_2(S)$
9: $NoImprove \leftarrow 0$
10: $H_4(h_4(S)) \leftarrow 1; H_5(h_5(S)) \leftarrow 1; H_6(h_6(S)) \leftarrow 1$/*Update the hash vectors with S*/
11: $(S^b, \alpha) \leftarrow Adaptive\_Update(S, S^b, \alpha)$ /*Update the penalty coefficient α adaptively and the best feasible solution S^b found during two-phase local search*/
12: **end if**
13: **end while**
14: **return** (S, S^b, H_k, α) /*$k = 4, 5, 6$*/

Flip operator: This operator removes a vertex from S or inserts a vertex into S. Let $flip(v_i)$ denote such a move. Then, the induced neighborhood N_2 contains all feasible and infeasible solutions obtained by applying "flip" to S.

$$N_2(S) = \{S \oplus flip(v_i) \mid v_i \in V \text{ and } v_i \notin S_c\} \tag{12}$$

The size of N_2 is restricted by $O(|V|)$. To evaluate the quality of the solutions, the following extended penalty-based fitness function F is used.

Extended fitness function: The extended fitness function F (to be minimized) is defined as a linear combination of the basic fitness function f (Equation (1)) and a penalty function:

$$F(S) = f(S) + \alpha \sum_{v_i \in V} \rho(v_i) \tag{13}$$

where α (α is initially set as 2) is the penalty coefficient to control the importance of the penalty function; $\rho(v_i)$ is the penalty of a vertex v_i, which equals the sum of the vertex weight and the weights of all edges in $E_{V \setminus S}(v_i)$ if the vertex has no neighbor vertex in S; otherwise $\rho(v_i)$ is 0. $\rho(v_i)$ can be calculated as by

$$\rho(v_i) = \begin{cases} w(v_i) + \sum\limits_{\{v_i, v_j\} \in E_{V \setminus S}(v_i)} c(v_i, v_j), & E_S(v_i) = \emptyset; \\ 0, & \text{otherwise}. \end{cases} \tag{14}$$

Fast Neighborhood Evaluation Technique: The incremental value of each $flip(v_i)$ in terms of the extended fitness variation $\Delta F(flip(v_i))$ (see Equation (13)) can be easily computed as per:

$$\begin{aligned} \Delta F(flip(v_i)) &= F(S \oplus flip(v_i)) - F(S) \\ &= \begin{cases} \alpha \sum\limits_{v_j \in C_1(v_i)} \rho(v_j) + \Delta f(remove(v_i)), & v_i \in S \text{ and } v_i \notin S_c; \\ -\alpha \sum\limits_{v_j \in C_2(v_i)} \rho(v_j) + \Delta f(insert(v_i)), & v_i \in V \setminus S. \end{cases} \end{aligned} \tag{15}$$

where $C_1(v_i)$ is the set of vertices that only neighboring to v_i in S, i.e., $C_1(v_i) = \{v_j | \{v_i, v_j\} \in E, E_S(v_j) = \{v_i\}\}$, and $C_2(v_i)$ is the set of neigh-

bor vertices of v_i that has no neighbors in S, i.e., $C_2(v_i) = \{v_j | \{v_i, v_j\} \in E, E_S(v_j) = \emptyset\}$.

Figure 7 presents a neighborhood solution by executing the flip operator. Let $S = \{v_3, v_4, v_6, v_8\}$ be the best solution found during the descent search with an objective value of 40. When flipping v_3, a best improved neighborhood solution $S \oplus flip(v_3)$ can be obtained. The incremental extended objective value is $\Delta F(flip(v_3)) = -5$ and the objective value of the improved solution is 35.

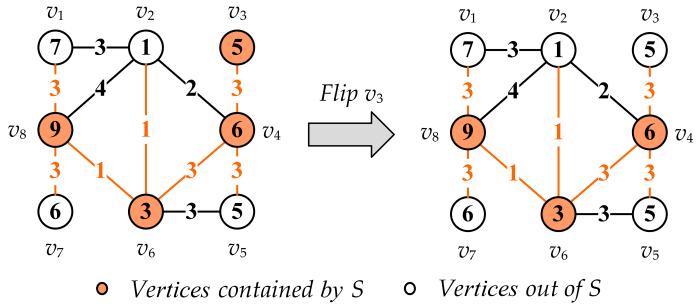


Fig. 7. Illustration of the solution-based feasible and infeasible search procedure (the vertices and edges involved by the objective function $f(S)$ are in orange and the other vertices and edges are not colored).

Adaptive Update Technique: According to whether the current solution S is a feasible solution, the adaptive update technique adjusts α to affect the search pathway of the next iteration of the search procedure. Specifically, if $F(S) \neq f(S)$ (line 1, Algorithm 7), which means the current solution S is an infeasible solution, we increase α to $\alpha + \alpha_{step}$ to lead the search toward feasible solutions (line 2, Algorithm 7). α_{step} is a parameter that is set to 2 according to the previous experiment. Inversely, we decrease α to $\alpha - \alpha_{step}$ to raise the chance of visiting infeasible solutions (line 4, Algorithm 7), and update the best feasible solution S^b found during two-phase local search if the current solution S is better than S^b (line 5, Algorithm 7).

4.3.2 Learning-based perturbation procedure

The learning-based perturbation has been successfully applied in iterated local search algorithms for solving the graph coloring problem [28], budgeted maximum coverage problem [6], and k-vertex-critical subgraphs problem [26]. Algorithm 8 gives the general framework of our learning-based perturbation, which is decomposed into two parts.

(1) Vertices of S are sorted in a non-increasing sequence and vertices of $V \setminus S$ are sorted in a non-descending sequence according to their probability values in Q, respectively.

Algorithm 7 Adaptive update technique for penalty coefficient

Input: The current solution S; the best feasible solution S^b found during two-phase local search; penalty coefficient α.
Output: Updated best feasible solution S^b found during two-phase local search; updated penalty coefficient α.

if $F(S) \neq f(S)$ **then**
  $\alpha \leftarrow \alpha + \alpha_{step}$ /*Increase the penalty term to lead the search toward feasible solutions*/
else
  $\alpha \leftarrow \alpha - \alpha_{step}$ /*Decrease the penalty term to raise the chance of visiting infeasible solutions*/
  $S^b = \arg\min\{f(S), f(S^b)\}$ /*Update the best feasible solution S^b found during two-phase local search*/
end if
if $\alpha \leq 0$ **then**
  $\alpha \leftarrow \alpha_{step}$
end if
return (S^b, α)

Algorithm 8 Learning-based perturbation

Input: Reduced graph $G = (V, E, w, c)$; the current solution S; the perturbation probability vector Q.
Output: A perturbed solution S.

$D_1 \leftarrow S$, $D_2 \leftarrow V \setminus S$
$D_1 \leftarrow$ sort D_1 in non-increasing order of the value in Q
$D_2 \leftarrow$ sort D_2 in non-descending order of the value in Q
for $r \leftarrow 1$ to 2 **do**
  $D_r \leftarrow$ the first η_r vertices from D_r
  for each $v_i \in D_r$ **do**
    if $1 - Q[i] > rand(0,1)$ and $r = 1$ **then**
      $S \leftarrow S \setminus \{v_i\}$ /*Remove vertex v_i from S according to $1 - Q[i]$*/
    else if $Q[i] > rand(0,1)$ and $r = 2$ **then**
      $S \leftarrow S \cup \{v_i\}$ /*Insert vertex v_i into S according to $Q[i]$*/
    end if
  end for
end for
return S

(2) The first η_r $(r = 1, 2)$ vertices in D_r are flipped according to their corresponding probabilities in Q, where η_1 is a parameter, which is the limit of the to-be flipped vertices of S when this vertex is in S, and η_2 is a parameter, which is the limit of the to-be flipped vertices out of S when this vertex is in $V \setminus S$. D_1 equals to S, and D_2 equals to $V \setminus S$.

Note that the vertices in S_c are not concerned by the perturbation since they are known to be part of any optimal total dominating set.

In the perturbation procedure, S does not follow the constraint that S must be a feasible solution. Therefore, the perturbed solutions can be feasible or infeasible.

Figure 8 shows an example of the learning-based perturbation for the local optimal solution $S = \{v_4, v_6, v_8\}$ from the two-phase local search. First, the vertices in S are sorted in a non-increasing order and $V \setminus S$ are sorted in a non-descending order according to their probabilities in Q, respectively. Then the first $\eta_1 = 1$ vertex v_6 in S and the first $\eta_2 = 2$ vertices $\{v_3, v_7\}$ in $V \setminus S$ are flipped with their probabilities in Q. This leads to the perturbed solution $S = \{v_3, v_4, v_7, v_8\}$.

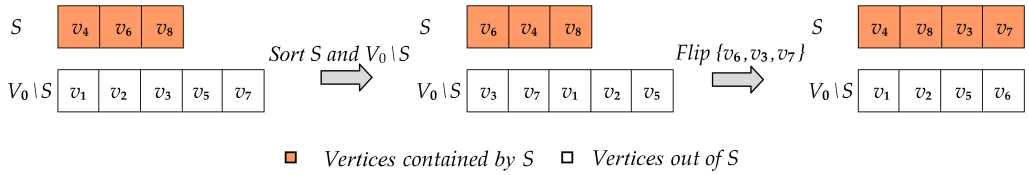


Fig. 8. Illustration of the learning-based perturbation procedure (the vertices contained in S are in orange and the other vertices are not colored).

Algorithm 3 illustrates that the proposed SILS algorithm repetitively calls the two-phase local search and the perturbation procedure until a termination criterion is met. The solution acquired through the perturbation procedure is employed as the new starting solution of the next round of the SILS.

4.3.3 Solution-based management strategy

It is important to prevent the search from revisiting a previously visited solution. To realize this, we adopt a tabu list [5] based on the hashing technique to determine whether a neighborhood solution was already visited. This technique has proven to be quite successful for solving several challenging optimization problems including the dispersion problems [14, 16], multidemand multidimensional knapsack problem [15], and dynamic bipartite drawing problem [25]. To our knowledge, this has not been investigated for solving the total dominating set problem.

Specifically, we employ six hash vectors H_k, $k = 1, \ldots, 6$, of length L (a parameter) associated with six hash function h_k. H_1 to H_3 are used during the first search phase (descent search) while H_4 to H_6 serve for the second search phase (feasible and infeasible search).

These hash vectors $H_k, k = 1, \ldots, 6$ are initially set to 0. Then, the corresponding positions of the hash vectors (H_1 to H_3 for the first phase and H_4 to H_6 for the second phase) are updated to 1 when a new neighborhood

solution S' is accepted to substitute the current solution S (lines 5-10, Algorithm 5, or lines 5-12, Algorithm 6). If this neighborhood solution S' has been previously encountered, i.e., $H_1[h_1(S')] \wedge H_2[h_2(S')] \wedge H_3[h_3(S')] = 1$ (or $H_4[h_5(S')] \wedge H_5[h_5(S')] \wedge H_6[h_6(S')] = 1$), S' will not be considered by the algorithm. The six hash functions h_k, $k = 1, \ldots, 6$, are defined as below.

$$h_k(S) = (\sum_{i=1}^{|V|} \lfloor i^{\xi_k} \rfloor \times x_i) \ mod \ L \tag{16}$$

where ξ_k refers to the parameters of hash functions, and L is the length of the hash vectors. We set $\xi_1 = 1.3, \xi_2 = 1.9, \xi_3 = 2.1, \xi_4 = 1.3, \xi_5 = 1.9, \xi_6 = 2.1$, and $L = 10^8$ according to preliminary experiments. At the beginning of the first (or second) phase, the hash value $h_k(S)$ of the current solution S is calculated. With the current solution S and its hash value $h_k(S)$, the hash value $h_k(S')$ of the neighborhood solution $S' = S \oplus remove\text{-}insert(v_i, A)$ (or $S' = S \oplus flip(v_i)$) is given by $h_k(S) + \sum_{v_j \in A} \lfloor j^{\xi_k} \rfloor - \lfloor i^{\xi_k} \rfloor$ (or $h_k(S) \pm \lfloor i^{\xi_k} \rfloor$).

4.4 Historical information update

As mentioned before, KILS creates initial solutions by using the historical information vector P (Section 4.2) and chooses perturbed vertices using the perturbation probability vector Q (Section 4.3.2). Both P and Q are updated each time the best feasible solution S^{best} found during SILS is improved.

1) Update the historical information vector P. The inspiration for updating $P = \{P[1], P[2], \ldots P[|V|]\}$ comes from the learning automata [22], whose principle is to reduce the probability $P[i] (1 \leq i \leq |V|)$ of each vertex v_i in S^b (recall that S^b is the best feasible solution found during the two-phase local search) while increasing the probability of each vertex v_i in $V \setminus S^b$. Specifically, we modify the probability value of each vertex as follows.

$$P[i] = \begin{cases} (1-\theta_1) \times P[i], & if \ v_i \in S^{best} \ and \ v_i \in S^b \\ (1-\theta_2) \times P[i], & if \ v_i \in V \setminus S^{best} \ and \ v_i \in S^b \\ \theta_1 + (1-\theta_1) \times P[i], & if \ v_i \in V \setminus S^{best} \ and \ v_i \in V \setminus S^b \\ \theta_2 + (1-\theta_2) \times P[i], & if \ v_i \in S^{best} \ and \ v_i \in V \setminus S^b \end{cases} \tag{17}$$

where reward factors $\theta_1 (0 < \theta_1 < 1)$ and $\theta_2 (0 < \theta_2 < 1)$ are respectively

set to 0.2 and 0.3 according to the previous experiment (see Section 5.2.1).

In addition, we use a probability smoothing technique to modify the historical information. When a probability in P reaches the specified probability threshold, a smoothing factor is applied to increase or decrease the probability to neglect some previous information as follows.

$$P[i] = \begin{cases} \beta_1 \times P[i], \, if \; P[i] > \tau, v_i \in V \\ \beta_2 \times P[i], \, if \; P[i] < 1 - \tau, v_i \in V \end{cases} \tag{18}$$

where the smoothing factors β_1 and β_2 are respectively set to 0.5 and 10, and the smoothing threshold τ is set to 0.95 experimentally (see Section 5.2.1).

2) Update the perturbation probability vector Q. We use a probability learning method (Equation (19)) to update the perturbation probability vector $Q = \{Q[1], Q[2], \ldots, Q[|V|]\}$ according to the principle of increasing the probability value $Q[i]$ $(1 \leq i \leq |V|)$ of each vertex v_i in the current solution S while decreasing that of each vertex v_i in $V \setminus S$. This is done as follows.

$$Q[i] = \begin{cases} \theta_3 + (1 - \theta_3) \times Q[i], \, if \; v_i \in S \\ (1 - \theta_3) \times Q[i], \qquad if \; v_i \in V \setminus S \end{cases} \tag{19}$$

where the reward factor θ_3 $(0 < \theta_3 < 1)$ is set to 0.1 experimentally (see Section 5.2.1). In addition, we smooth the probability vector Q in the same way as we smooth the historical information vector P.

4.5 Computational complexity

The time complexity of the learning-based initialization procedure for graph $G = (V, E, w, c)$ is $O(K_0 \times |V| \times Max_{deg})$, where K_0 is the quantity of iterations of initialization and Max_{deg} is the maximum vertex degree in V. The complexity of the two-phase local search and the perturbation procedure is $O(K_1 \times |V| \times Max_{deg}^2)$, where K_1 is the maximum iteration of solution-based descent phase. Then, the complexity of the solution-based iterated local search is $O(K_2 \times K_1 \times |V| \times Max_{deg}^2)$, where K_2 is the maximum iteration of the solution-based iterated local search. Finally, the overall time

complexity of the main algorithm is $O(K \times K_2 \times K_1 \times |V| \times Max_{deg}^2)$, where K is the maximum iteration of KILS.

5 Experimental results and comparisons

We present in this section detailed experimental results and assessment of the proposed KILS algorithm based on 315 conventional benchmark instances used in the literature [1,12] and 27 real-world benchmark instances that were initially tested in [12].

5.1 Benchmark instances

The set of the 315 conventional WTDP instances in the literature were generated according to the Erdös-Rényi model [1, 12, 20] and the set of 27 realistic instances were from real-world social networks [12]. These instances are organized into four groups.

(1) **MA (45 small instances)**: The first group from [20] were generated by adopting the "gnp_random_graph ($|V|$,p)" function in "networkx" package [7]. Names of these instances are in the form of MA-$|V|$-p-a-b-id, where MA is the name of the instance group, $|V| \in \{20, 50, 100\}$ is the number of vertices in the graph, and $p \in \{0.2, 0.5, 0.8\}$ is the edge density. The vertex weights and edge weights are chosen from $\{1, \ldots, a\}$ and $\{1, \ldots, b\}$, respectively, where $a = b = 5$. The vertex size and density are combined in 9 different ways, and each combination results in 5 different instances labeled with an index $id \in \{1, \ldots, 5\}$. The optimal solutions for these 45 instances in this group were reported in [1]. These instances are available at https://msinnl.github.io/pages/instancescodes.html.

(2) **AMS (135 medium instances)**: The second group from [1] are also generated by using the "gnp_random_graph ($|V|$,p)" function. These instances are named as AMS-$|V|$-p-a-b-id, with {75, 100, 125} vertices and a density $p \in \{0.2, 0.5, 0.8\}$. Vertex weights and edge weights are sampled from $\{1, \ldots, a\}$ and $\{1, \ldots, b\}$, respectively, where (a, b) $\in$ {(50, 10), (25, 25), (10, 50)}. We distinguish 5 graphs that are sampled for each of the 27 configurations $\{(1, 50, 1, 10), (1, 25, 1, 25), (1, 10, 1, 50)\} \times \{75, 100, 125\} \times \{0.2, 0.5, 0.8\}$ identified by id, leading to 135 AMS instances. The optimal solutions of 115 instances are given in [12], while for the remaining 20 instances, no optimal solution is known. These instances are available at https://msinnl.github.io/pages/instancescodes.html.

(3) **NEW (135 large instances)**: The instances of the third group, named by NEW-$|V|$-*p-a-b-id* were introduced in [12]. They were randomly generated in the same way as the AMS set in [1]. These large instances have $\{250, 500, 1000\}$ vertices and their optimal solutions are unknown. These instances are available at https://github.com/StefanKapunac/wtdp_public.

(4) **SNAP (27 real-world instances)**: The fourth group is composed of 27 real-world instances from the Stanford large network dataset collection [18]. These instances represent parts of real-world social networks such as Facebook and Twitter, which are characterized by the number of vertices within $[1912, 82168]$ and the number of edges within $[31299, 1342310]$. Since there are no vertex weights or edge weights for these instances, we set the vertex weights and edge weights sampled from $\{1, \ldots, a\}$ and $\{1, \ldots, b\}$, respectively, where (a, b) $\in \{(5, 5), (25, 25), (50, 50)\}$. To ensure that they are simple graphs we remove the self-loops and multiple edges from these instances. The optimal solutions for these instances are unknown. These instances are available at https://github.com/Miserable-ccf/WTDP.

5.2 Experimental settings

Our KILS algorithm was written in C++ [1] and compiled by the g++ compiler with the -O3 option. All experiments are performed on an Intel Core 2 Duo T7700 processor (2.4 GHz and 2 GB RAM) under the CentOS release 7.9.2009.

5.2.1 Parameters

The KILS algorithm requires several parameters (see Table 2): the greediness ratio ϵ of initialization, the step size α_{step} of penalty coefficient α, the search depth ω of SILS, the limit η_1 of the to-be flipped vertices of S, the limit η_2 of the to-be flipped vertices out of S, reward factors θ_1, θ_2 and θ_3 of the probability vectors, smoothing factors β_1 and β_2, and smoothing threshold τ. We identify the suitable values of the parameters by running the automatic parameter tuning package called "IRACE" [19] on 45 representative instances of group NEW. We perform 20 runs within a calibration budget of 1800 seconds. Table 2 provides the candidate and final values of these parameters. The final parameter values can be considered as the default parameter settings, which are consistently used through all experiments reported in this paper.

[1] The binary code of the algorithm will be made available upon the publication of this work.

Table 2
Settings of parameters.

Parameters	Description	Candidate value	Final value
ϵ	The greediness ratio of initialization	$\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7\}$	0.6
α_{step}	The step size of the penalty coefficient α	$\{1, 2, 3, 4, 5\}$	2
ω	The search depth of SILS	$\{\frac{10^6}{\|V\|}, \frac{10^5}{\|V\|}, \frac{10^4}{\|V\|}\}$	$\frac{10^6}{\|V\|}$
η_1	limit of to-be flipped vertices of S	$\{0.1\|S\|, 0.5\|S\|, 0.9\|S\|\}$ (for $\|V\| \leq 3000$)	$0.5\|S\|$
		$\{0.01\|S\|, 0.02\|S\|, 0.006\|S\|\}$ (for $\|V\| > 3000$)	$0.01\|S\|$
η_2	limit of to-be flipped vertices out of S	$\{0.1\|V \setminus S\|, 0.5\|V \setminus S\|, 0.9\|V \setminus S\|\}$ (for $\|V\| \leq 3000$)	$0.5\|V \setminus S\|$
		$\{0.006\|V \setminus S\|, 0.01\|V \setminus S\|, 0.02\|V \setminus S\|\}$ (for $\|V\| > 3000$)	$0.01\|V \setminus S\|$
$\theta 1$	First reward factor of the probability vector P	$\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7\}$	0.2
$\theta 2$	Second reward factor of the probability vector P	$\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7\}$	0.3
$\theta 3$	Reward factor of the probability vector Q	$\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7\}$	0.1
$\beta 1$	First smoothing factor	$\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7\}$	0.5
$\beta 2$	Second smoothing factor	$\{1, 5, 7, 10\}$	10
τ	Smoothing threshold	$\{0.9, 0.93, 0.95, 0.97\}$	0.95

5.2.2 Reference algorithms

The following three latest algorithms are used for our comparative study: the mixed integer programming model solved by CPLEX (re-MIP) [12], the genetic algorithm (re-GA) [12], and the variable neighborhood search algorithm (VNS) [12]. re-MIP is the re-implementation in [12] of the mixed integer programming model proposed in [1]. re-GA is the re-implementation in [12] of the genetic algorithm presented in [1]. The codes of re-MIP and re-GA are available at https://github.com/StefanKapunac/wtdp_public.

For the first three groups of instances, the reference algorithms were run in [12] on a computer with an Intel Core i9-9900KF processor (3.6 GHz and 6 GB RAM) with a time limit of 1800 seconds. Since our computer (Inter Core T7700 2.4GHz and 2 GB RAM) is slower, running the reference algorithms on our computer with the same time limit of 1800 seconds will produce results equal to or worse than those reported in [12]. Therefore, we use the best results of re-MIP, re-GA, and VNS from [12] for our comparative study. For the fourth group of instances, we ran the source codes of the reference algorithms (re-MIP, re-GA and VNS) on our computer to solve these instances with a time limit of 18000 seconds.

1) The CPLEX solver (re-MIP) [12]: re-MIP reaches 45/45, 115/135, 0/135 and 5/27 optimal solutions for the MA, AMS, NEW, and SNAP instances.

2) Heuristic algorithm re-GA [12]: re-GA reports 41/45 optimal solutions, 105/135 (including 89 optimal solutions), 64/135 and 12/27 best-known values for the MA, AMS, NEW, and SNAP instances.

3) Heuristic algorithm VNS [12]: VNS obtains 45/45 optimal solutions, 132/135 (including 115 optimal solutions), 107/135, 9/27 best-known values for the

MA, AMS, NEW, and SNAP instances.

5.2.3 Stopping conditions

Following [1, 12], we adopt a time limit of 1800s for the first three groups of instances and 18000s for the fourth group of instances, which is a more restrictive condition compared to the reference algorithms because the machine used to run the reference algorithms (Intel Core i9-9900KF processor with 3.6 GHz and a memory limit of 6 GB RAM) is 1.5 times faster than our computer (Intel Core 2 Duo T7700 processor with 2.4 GHz and a memory limit of 2 GB RAM) according to the SPEC evaluation [2] .

The number of independent runs for VNS and re-GA is not indicated in [1, 12]. Given the stochastic nature of our algorithm, we independently run our algorithm 20 times to solve each instance with a different random seed for each run.

5.3 Comparison with reference algorithms

This section compares our algorithm with the state-of-the-art algorithms on the four groups of 342 benchmark instances.

Table 3
Summary of comparative results between the KILS algorithm and three reference algorithms.

Algorithm pair	Instance group/Nb_of_instances	Indicator	#*Wins*	#*Ties*	#*Losses*	*p*-values
KILS vs re-MIP	MA/45	f_{best} (*time*)	0 (-)	45 (-)	0 (-)	1 (-)
	AMS/135	f_{best} (*time*)	0 (-)	135 (-)	0 (-)	1 (-)
	NEW/135	f_{best} (*time*)	133 (-)	2 (-)	0 (-)	- (-)
	SNAP/27	f_{best} (*time*)	22 (-)	5 (-)	0 (5)	- (-)
KILS vs re-GA	MA/45	f_{best} (*time*)	4 (45)	41 (0)	0 (0)	0.125 (5.18E-09)
	AMS/135	f_{best} (*time*)	30 (135)	105 (0)	0 (0)	1.65E-06 (6.71E-24)
	NEW/135	f_{best} (*time*)	97 (135)	38 (0)	0 (0)	1.21E-17 (6.71E-24)
	SNAP/27	f_{best} (*time*)	27 (27)	0 (0)	0 (0)	5.61E-06 (4.90E-05)
KILS vs VNS	MA/45	f_{best} (*time*)	0 (45)	45 (0)	0 (0)	1 (5.18E-09)
	AMS/135	f_{best} (*time*)	3 (135)	132 (0)	0 (0)	0.25 (6.71E-24)
	NEW/135	f_{best} (*time*)	79 (135)	56 (0)	0 (0)	1.14E-14 (1.05E-23)
	SNAP/27	f_{best} (*time*)	27 (22)	0 (0)	0 (5)	5.61E-06 (0.017)

[1] Note: "-" means the corresponding result is not available in the literature.

Table 3 shows a summary of the comparison between the KILS algorithm and each reference algorithm. Column 1 indicates the compared pairwise

[2] https://www.spec.org

algorithms. Column 2 tells the names of instance groups with the number of instances. Column 3 indicates the quality indicator in terms of the best objective value (f_{best}) and the average computation time ($time$) to obtain the best result of each algorithm over 20 runs, where"-" means the corresponding result and time is not available in the literature. Columns 4-6 provide the number of instances on which KILS achieves a better, equal or worse result compared with re-MIP, re-GA and VNS respectively (#$Wins$, #$Ties$ and #$Losses$). Column 7 gives the p-values of the non-parametric Wilcoxon signed-rank test which indicates a significant difference between two sets of compared results if less than 0.05, and no difference otherwise. Meanwhile, the detailed results for each instance group are reported in Tables B.1 - B.4 of the appendix.

We can make the following observations from Table 3. For the all instances (MA, AMS, NEW and SNAP), the KILS algorithm performs remarkably well by attaining 93 new upper bounds and 249 equal results (including 165 known optimal results). Besides, it consumes less time on 326 instances than all reference algorithms, considering that the reference algorithms use a faster computer than ours.

Compared with the most recent exact algorithm re-MIP, KILS behaves better in terms of f_{best} by reporting 155 better results and 187 equal results for the 342 instances. Besides, KILS outperforms re-GA by achieving 158 better results and 184 equal results. Finally, KILS dominates VNS by obtaining 109 better results and 233 equal results for the 342 instances.

6 Analysis

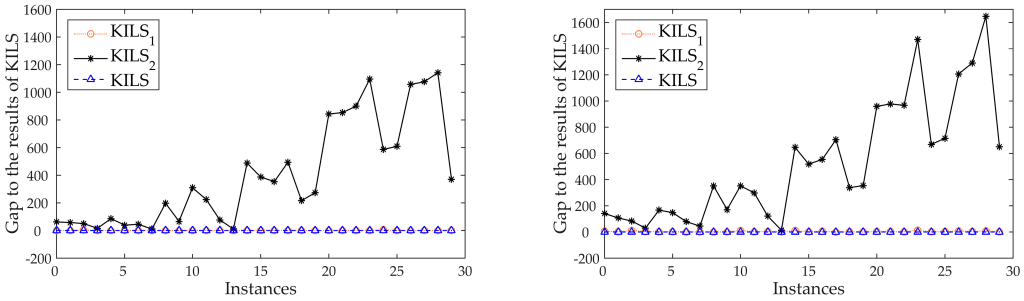
This section studies the contributions of each critical component of the proposed algorithm. The experiments for this study are based on a selection of 30 most challenging instances, which are solved 20 times by each investigated algorithm variant within a limit of 1800s per run.

6.1 Effectiveness of the two-phase local search

We create two variants $KILS_1$ and $KILS_2$ by respectively disabling the first phase (solution-based descent phase) and the second phase (solution-based feasible and infeasible phase) to justify the significance of these two phases of the local search.

Figure 9 shows their best results f_{best} and average results f_{avg}. The X-axis

indicates the instances numbered according to the order they appear in Table B.3. The Y-axis indicates the gap between f_{best} (or f_{avg}) and the new upper bound NUB_{best} (or NUB_{avg}) from KILS, i.e., $gap = f_{best} - NUB_{best}$ for Figure 9(a) and $gap = f_{avg} - NUB_{avg}$ for Figure 9(b).



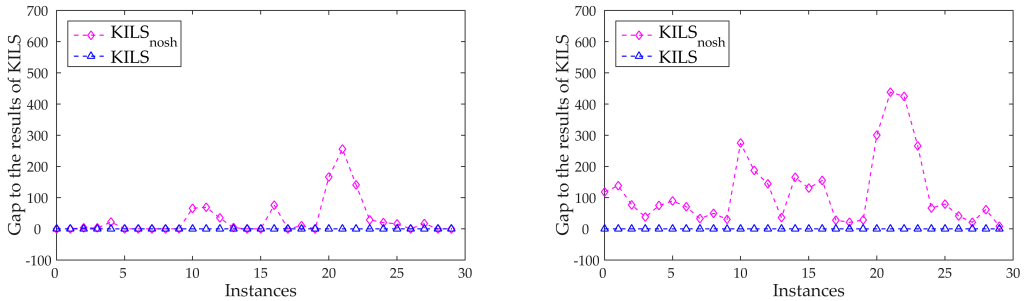
(a) Best results f_{best} (b) Average results f_{avg}

Fig. 9. Comparisons of KILS (in blue) with its variants $KILS_1$ (in orange) and $KILS_2$ (in black).

We observe from Figure 9 that KILS exhibits a superior performance in terms of the best results and the average results. Specifically, compared with the variant $KILS_1$, KILS obtains 2 (28) better results and 28 (2) equal results in terms of f_{best} (f_{avg}). KILS dominates $KILS_2$ by reporting 30 (30) better results in terms of f_{best} (f_{avg}) on all instances. Moreover, small p-values (3.79E-06, 1.73E-06 and 1.73E-06) confirm the statistically significant differences between KILS and the two variants.

6.2 *Effectiveness of solution-based hash strategy*

We implement a variant $KILS_{nosh}$ of the KILS algorithm by only removing the solution-based hash strategy, and compare it with KILS to justify the hash strategy as an essential component of the KILS algorithm.



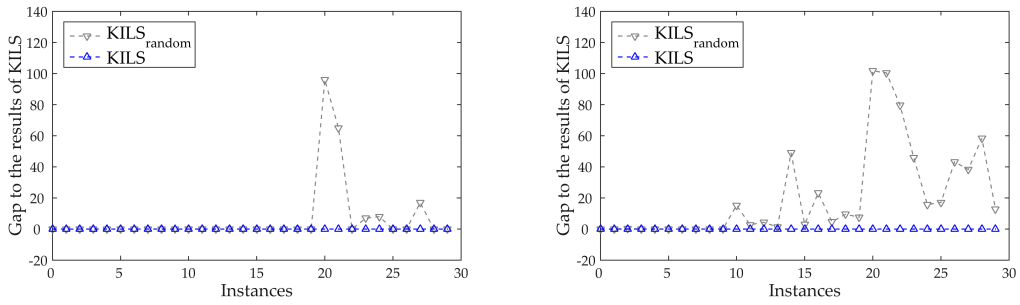
(a) Best results f_{best} (b) Average results f_{avg}

Fig. 10. Comparisons of KILS (in blue) with its variant $KILS_{nosh}$ (in magenta).

We observe from Figure 10 that KILS dominates KILS$_{nosh}$ by reporting 16 (30) better results and 14 (0) equal results in terms of f_{best} (f_{avg}). Besides, the small p-values (4.37E-04 and 1.73E-06) indicate that the performance differences between the compared results are statistically significant. This experiment demonstrates the usefulness of the solution-based strategy for solving the WTDP under the two-phase local search.

6.3 *Effectiveness of learning-based strategy*

To analyze the joint impacts of the learning-based initialization (Section 4.2) and perturbation (Section 4.3.2), we implement a variant $KILS_{random}$, where the initial solution is replaced by a random solution and the learning-based perturbation is replaced by a random perturbation. Specifically, the random initial solution is still generated following the process of Algorithm 2, but all values in P are fixed to 0.5; the random perturbation is still followed the process of Algorithm 8, but all values in Q are fixed to 0.5.



(a) Best results f_{best} (b) Average results f_{avg}

Fig. 11. Comparisons of KILS (in blue) with its variant KILS$_{random}$ (in grey).

Figure 11 shows that KILS significantly outperforms the random perturbation strategy (KILS$_{random}$). Specifically, compared with KILS$_{random}$, KILS obtains better f_{best} values for 5 out of the 30 instances, better f_{avg} values for 22 instances, and equal results for the remaining instances respectively. Moreover, judging from Table 4, the p-value (4.01E-05) confirms the statistically significant differences between the results of KILS and KILS$_{random}$.

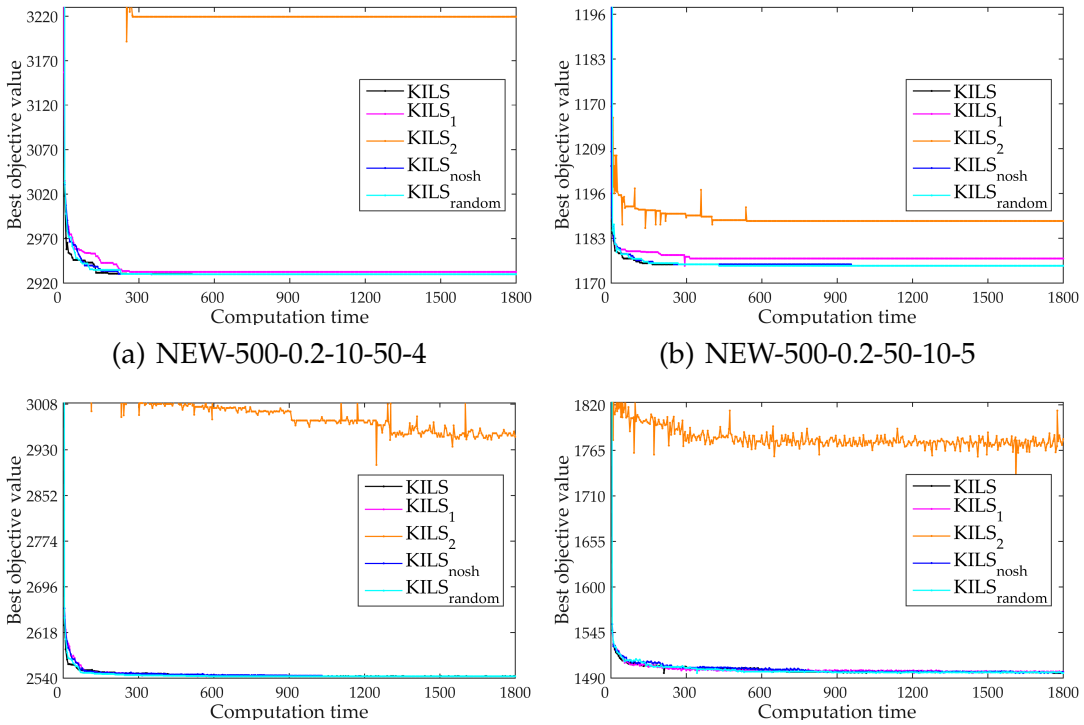
6.4 *Convergence analysis*

To investigate the most crucial component of KILS in time decrease, we conduct an additional convergence analysis of KILS and its four aforementioned variants (KILS$_1$, KILS$_2$, KILS$_{nosh}$ and KILS$_{random}$). This experiment is

Table 4
Summary of comparative results between the KILS algorithm and four variants $KILS_1$, $KILS_2$, $KILS_{nosh}$ and $KILS_{random}$.

Algorithm pair	Nb of instances	Indicator	#Wins	#Ties	#Losses	p-values
KILS vs $KILS_1$	30	f_{best} (f_{avg})	2 (28)	28 (2)	0 (0)	0.50 (3.79E-06)
KILS vs $KILS_2$	30	f_{best} (f_{avg})	30 (30)	0 (0)	0 (0)	1.73E-06 (1.73E-06)
KILS vs $KILS_{nosh}$	30	f_{best} (f_{avg})	16 (30)	14 (0)	0 (0)	4.37E-04 (1.73E-06)
KILS vs $KILS_{random}$	30	f_{best} (f_{avg})	5 (22)	25 (8)	0 (0)	0.06 (4.01E-05)

carried out on four difficult instances NEW-500-0.2-10-50-4, NEW-500-0.2-50-10-5, NEW-500-0.5-10-50-3 and NEW-500-0.8-25-25-5. Each algorithm is executed 20 times to solve each instance within a time limit of 1800 seconds per run. The convergence graphs (i.e., running profiles) are depicted in Figure 12, where the X-axis represents the computation time in seconds and the Y-axis indicates the best objective value achieved by each algorithm.



(a) NEW-500-0.2-10-50-4 (b) NEW-500-0.2-50-10-5

(c) NEW-500-0.5-10-50-3 (d) NEW-500-0.8-25-25-5

Fig. 12. Convergence profiles of KILS and four variants ($KILS_1$, $KILS_2$, $KILS_{nosh}$ and $KILS_{random}$)

Figure 12 confirms the superiority of KILS over other algorithms as evidenced by the fact that the curve of KILS consistently lies below those of other algorithms, although the performance difference between the compared algorithms is marginal. It can also be observed that the curve of $KILS_2$ strictly runs above the curve of other algorithms, indicating the importance of using the second phase (the feasible and infeasible search phase) of the

local search procedure.

The feasible and infeasible search phase oscillates between feasible and infeasible regions, which can locate high-quality solutions that are difficult to obtain if the search is restricted to feasible regions only. This strategy proves to be crucial in helping the algorithm to effectively find high-quality solutions to the constrained WTDP problem.

7 Conclusions and prospects

The weighted total domination problem is a general model which is useful for a variety of applications. The proposed knowledge-based iterated local search algorithm includes a reduction procedure (to reduce the input graph), a learning-based initialization (to generate an initial solution), and a solution-based iterated local search procedure (to find high-quality solutions).

Extensive empirical tests and comparative results on 342 benchmark instances show a remarkable performance of the proposed algorithm. Specifically, it discovers 93 new upper bounds and attains 249 current best results (including the 165 known optimal results). Moreover, the proposed algorithm is computationally efficient as it only needs a fraction of the time (about $< 10\%$) required by the state-of-the-art algorithm in groups MA and AMS to attain the best known solutions.

For future research, we can advance two directions. First, population-based approaches would be investigated, where the proposed algorithm can be advantageously used as a critical intensification component. Second, given its excellent performance for the weighted total domination problem, it would be interesting to adapt the proposed algorithm to other related domination problems [23, 24].

Acknowledgments

We are grateful to the reviewers for their useful comments and suggestions, which helped us to improve the paper. This work is partially supported by the National Natural Science Foundation of China [GrantNo. 72122006, No. 62101125], and Fundamental Research Funds for the Central Universities [Grant No. 2242022R40067, No. 2242022K30007].

References

[1] E. Álvarez-Miranda, M. Sinnl, Exact and heuristic algorithms for the weighted total domination problem, Computers & Operations Research 127 (2021) 105157.

[2] Y. N. Chen, J. K. Hao, An iterated "hyperplane exploration" approach for the quadratic knapsack problem, Computers & Operations Research 77 (2017) 226–239.

[3] E. J. Cockayne, R. Dawes, S. T. Hedetniemi, Total domination in graphs, Networks 10 (3) (1980) 211–219.

[4] Q. Z. Fang, H. K. Kim, D. S. Lee, Total dominating set games, in: International Workshop on Internet and Network Economics, 2005, pp. 520–530.

[5] F. Glover, M. Laguna, Tabu Search, 1997, Springer New York, NY.

[6] F. Glover, Z. P. Lü, J. K. Hao, Diversification-driven tabu search for unconstrained binary quadratic problems, 4OR-A Quarterly Journal of Operations Research 8 (3) (2010) 239–253.

[7] A. Hagberg, P. Swart, A. D. Schult, Exploring network structure, dynamics, and function using networkx, Proceedings of the 7th Python in Science Conference 46 (2008) 113–130.

[8] P. F. He, J. K. Hao, Iterated two-phase local search for the colored traveling salesmen problem, Engineering Applications of Artificial Intelligence 97 (2021) 104018.

[9] P. F. He, J. K. Hao, Q. H. Wu, Grouping memetic search for the colored traveling salesmen problem, Information Sciences 570 (2021) 689–707.

[10] M. A. Henning, Restricted total domination in graphs, Discrete Mathematics 289 (1-3) (2004) 25–44.

[11] M. A. Henning, Signed total domination in graphs, Discrete Mathematics 278 (1-3) (2004) 109–125.

[12] S. Kapunac, A. Kartelj, M. Djukanović, Variable neighborhood search for weighted total domination problem and its application in social network information spreading, Applied Soft Computing (2023) 110387.

[13] V. Kulli, B. Chaluvaraju, M. Kumara, Graphs with equal secure total domination and inverse secure total domination numbers, Journal of Information and Optimization Sciences 39 (2) (2018) 467–473.

[14] X. J. Lai, J. K. Hao, F. Glover, D. Yue, Intensification-driven tabu search for the minimum differential dispersion problem, Knowledge-Based Systems 167 (2019) 68–86.

[15] X. J. Lai, J. K. Hao, D. Yue, Two-stage solution-based tabu search for the multidemand multidimensional knapsack problem, European Journal of Operational Research 274 (1) (2019) 35–48.

[16] X. J. Lai, D. Yue, J. K. Hao, F. Glover, Solution-based tabu search for the maximum min-sum dispersion problem, Information Sciences 441 (2018) 79–94.

[17] R. Laskar, J. Pfaff, S. M. Hedetniemi, S. T. Hedetniemi, On the algorithmic complexity of total domination, SIAM Journal on Algebraic Discrete Methods 5 (3) (1984) 420–425.

[18] J. Leskovec, A. Krevl, Snap datasets: Stanford large network dataset collection (2014).

[19] M. López-Ibáñez, J. Dubois-Lacoste, L. P. Cáceres, M. Birattari, T. Stützle, The irace package: Iterated racing for automatic algorithm configuration, Operations Research Perspectives 3 (2016) 43–58.

[20] Y. D. Ma, Q. Q. Cai, S. Y. Yao, Integer linear programming models for the weighted total domination problem, Applied Mathematics and Computation 358 (2019) 146–150.

[21] S. Muthukumar, J. Anitha, Total domination in silicate networks, Journal of Information and Computational Science 10 (11) (2020) 331–337.

[22] K. S. Narendra, M. A. L. Thathachar, Learning automata - an introduction, Prentice Hall, 1989.

[23] D. D. Niu, B. Liu, M. H. Yin, Y. P. Zhou, A new local search algorithm with greedy crossover restart for the dominating tree problem, Expert Systems with Applications 229 (Part A) (2023) 120353.

[24] D. D. Niu, X. L. Nie, L. L. Zhang, H. M. Zhang, M. H. Yin, A greedy randomized adaptive search procedure (GRASP) for minimum weakly connected dominating set problem, Expert Systems with Applications 215 (2023) 119338.

[25] B. Peng, D. H. Liu, Z. P. Lü, R. Martí, J. W. Ding, Adaptive memory programming for the dynamic bipartite drawing problem, Information Sciences 517 (2020) 183–197.

[26] W. Sun, J. K. Hao, A. Caminada, Iterated backtrack removal search for finding k-vertex-critical subgraphs, Journal of Heuristics 25 (4-5) (2019) 565–590.

[27] M. Vasquez, J. K. Hao, A hybrid approach for the 0-1 multidimensional knapsack problem, in: International Joint Conference on Artificial Intelligence, 2001, pp. 328–333.

[28] Y. M. Zhou, B. Duval, J. K. Hao, Improving probability learning based local search for graph coloring, Applied Soft Computing 65 (2018) 542–553.

A Reduction strategy

The two reduction rules used by the proposed algorithm to reduce the given graph $G_0 = (V_0, E_0, w_0, c_0)$ are defined as follows.

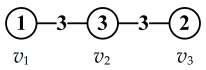
Rule 1: If vertex v_i has a degree of 1, its neighbor vertex v_j is identified as a crucial vertex, and included in the crucial vertex set S_c.

Proof of Rule 1: For each vertex v_i of degree of 1, if its neighbor vertex v_j is not a crucial vertex, then at least there is one total dominating set S' that does not contain v_j. For S', v_i has no neighbor vertex in S', which is in contradiction with the condition that S' is a total dominating set. Therefore, v_j must be a crucial vertex.

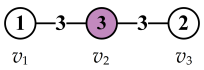
Rule 2: For each crucial vertex v_j that has more than one neighbor vertex whose degree is 1, all its neighbor vertices, except the one and the only one vertex v_i with the minimum vertex weight, will be identified as redundant vertices and contained by the redundant vertex set S_r.

Proof of Rule 2: For each crucial vertex v_j that has more than one neighbor vertex with a degree of 1, there is one of its neighbor vertices v_m, except the unique vertex v_i with the minimum vertex weight, such that v_m is not a redundant vertex. Then according to the definition of redundant vertex, v_m at least belongs to one optimal solution. For any optimal solution S' that contains v_m, the solution $S' \setminus \{v_m\} \cup \{v_i\}$ is a total dominating set since the only neighbor vertex v_j of v_m in S' is also neighbor of v_i, and $f(S' \setminus \{v_m\} \cup \{v_i\}) < f(S')$ since $w_0(v_m) < w_0(v_i)$. Then S' is not an optimal solution, which conflicts with the condition that S' is an optimal total dominating set.

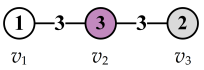
For example, Figure A.1(a) is a given graph G_0 with 3 vertices $\{v_1, v_2, v_3\}$ and 2 edges. As shown in Figure A.1(b), because both vertices v_1 and v_3 have a degree of 1, their neighbor vertex v_2 is a crucial vertex and is contained in S_c according to Rule 1. After Rule 1 identifies all crucial vertices, i.e., $S_c = \{v_2\}$ (in violet), Rule 2 is applied. Vertex v_1 has the minimum vertex weight among all v_2's neighbors whose degree is 1. Therefore, v_3 is in the redundant set S_r. After all vertices in S_c are traversed, $S_r = \{v_3\}$ (in grey).



(a) A given graph G_0



(b) A crucial set $S_c = \{v_2\}$



(c) A redundant set $S_r = \{v_3\}$

Fig. A.1. Illustration of reduction operation for a given graph G_0 (crucial vertices and the edges involved by them are colored in violet, redundant vertices are in grey).

B Detailed comparative results between KILS and the state-of-the-art methods

This appendix presents detailed comparative results of re-MIP, re-GA, VNS [12] and KILS on all benchmark instances of the four groups MA, AMS, NEW, and SNAP (Tables B.1, B.2, B.3, and B.4). The first two columns give the names of instances and the corresponding best-known values ($BKVs$) in the literature, respectively. The remaining columns report the best result (f_{best}), the average time ($time$) in seconds to reach the best result of re-MIP, re-GA, VNS and KILS over 20 runs, and the percentage gap ($pg\% = \frac{f_{best}-BKVs}{BKVs}$) between the f_{best} and $BKVs$ (A negative value indicates an improved best-known result). The row (Average) indicates the average value of each column. Entries "-" mean the corresponding result is not available in the literature. Entries with "n/a" indicate that no solution is found by re-MIP within the time limit. The best results among the compared values are highlighted in boldface. The optimal results are indicated by "*".

It can be seen from Tables B.1, B.2, B.3, and B.4 that KILS outperforms the reference algorithms for all these instances. Specifically, compared with the reference algorithms (re-MIP, re-GA and VNS), KILS obtains respectively (155, 158 and 109) better results and (187, 184 and 233) equal results, respectively. Besides, KILS obtains the best average results of f_{best} in each instance group compared with re-MIP, re-GA and VNS. Moreover, KILS spends much less time for 326 instances than the re-MIP, re-GA and VNS to reach their best results.

Table B.1
Comparative results of KILS with re-MIP, re-GA and VNS on MA instances with a time limit of 1800 seconds.

Instance	*BKVs*	re-MIP [12]		re-GA [12]			VNS [12]			KILS		
		f_{best}	*time*	f_{best}	*time*	*pg%*	f_{best}	*time*	*pg%*	f_{best}	*time*	*pg%*
MA-20-0.2-5-5-1	**63***	**63***	-	**63***	1.56	0.00%	**63***	2.35	0.00%	**63***	**0.13**	0.00%
MA-20-0.2-5-5-2	**58***	**58***	-	**58***	1.80	0.00%	**58***	1.45	0.00%	**58***	**0.13**	0.00%
MA-20-0.2-5-5-3	**58***	**58***	-	**58***	1.51	0.00%	**58***	1.42	0.00%	**58***	**0.14**	0.00%
MA-20-0.2-5-5-4	**51***	**51***	-	**51***	2.00	0.00%	**51***	1.45	0.00%	**51***	**0.14**	0.00%
MA-20-0.2-5-5-5	**55***	**55***	-	**55***	1.80	0.00%	**55***	1.44	0.00%	**55***	**0.15**	0.00%
MA-20-0.5-5-5-1	**44***	**44***	-	**44***	1.74	0.00%	**44***	1.71	0.00%	**44***	**0.14**	0.00%
MA-20-0.5-5-5-2	**47***	**47***	-	**47***	2.03	0.00%	**47***	1.56	0.00%	**47***	**0.15**	0.00%
MA-20-0.5-5-5-3	**46***	**46***	-	**46***	1.96	0.00%	**46***	1.56	0.00%	**46***	**0.13**	0.00%
MA-20-0.5-5-5-4	**40***	**40***	-	**40***	1.41	0.00%	**40***	1.52	0.00%	**40***	**0.14**	0.00%
MA-20-0.5-5-5-5	**41***	**41***	-	**41***	1.89	0.00%	**41***	1.54	0.00%	**41***	**0.13**	0.00%
MA-20-0.8-5-5-1	**37***	**37***	-	**37***	2.08	0.00%	**37***	1.37	0.00%	**37***	**0.13**	0.00%
MA-20-0.8-5-5-2	**35***	**35***	-	**35***	1.93	0.00%	**35***	1.65	0.00%	**35***	**0.13**	0.00%
MA-20-0.8-5-5-3	**40***	**40***	-	**40***	1.86	0.00%	**40***	1.63	0.00%	**40***	**0.13**	0.00%
MA-20-0.8-5-5-4	**34***	**34***	-	**34***	2.21	0.00%	**34***	1.41	0.00%	**34***	**0.14**	0.00%
MA-20-0.8-5-5-5	**34***	**34***	-	**34***	1.98	0.00%	**34***	1.78	0.00%	**34***	**0.14**	0.00%
MA-50-0.2-5-5-1	**111***	**111***	-	**111***	6.52	0.00%	**111***	5.77	0.00%	**111***	**0.15**	0.00%
MA-50-0.2-5-5-2	**106***	**106***	-	**106***	7.02	0.00%	**106***	4.51	0.00%	**106***	**0.14**	0.00%
MA-50-0.2-5-5-3	**111***	**111***	-	**111***	7.40	0.00%	**111***	4.53	0.00%	**111***	**0.15**	0.00%
MA-50-0.2-5-5-4	**101***	**101***	-	**101***	7.73	0.00%	**101***	4.60	0.00%	**101***	**0.15**	0.00%
MA-50-0.2-5-5-5	**108***	**108***	-	110	6.51	1.85%	**108***	4.69	0.00%	**108***	**0.14**	0.00%
MA-50-0.5-5-5-1	**82***	**82***	-	**82***	8.13	0.00%	**82***	5.08	0.00%	**82***	**0.13**	0.00%
MA-50-0.5-5-5-2	**85***	**85***	-	**85***	6.11	0.00%	**85***	4.79	0.00%	**85***	**0.14**	0.00%
MA-50-0.5-5-5-3	**84***	**84***	-	85	8.35	1.19%	**84***	5.54	0.00%	**84***	**0.15**	0.00%
MA-50-0.5-5-5-4	**82***	**82***	-	**82***	8.42	0.00%	**82***	5.39	0.00%	**82***	**0.15**	0.00%
MA-50-0.5-5-5-5	**82***	**82***	-	**82***	7.15	0.00%	**82***	5.43	0.00%	**82***	**0.14**	0.00%
MA-50-0.8-5-5-1	**77***	**77***	-	**77***	7.81	0.00%	**77***	5.70	0.00%	**77***	**0.15**	0.00%
MA-50-0.8-5-5-2	**72***	**72***	-	**72***	8.02	0.00%	**72***	5.73	0.00%	**72***	**0.13**	0.00%
MA-50-0.8-5-5-3	**74***	**74***	-	**74***	7.43	0.00%	**74***	5.70	0.00%	**74***	**0.14**	0.00%
MA-50-0.8-5-5-4	**76***	**76***	-	**76***	7.07	0.00%	**76***	5.52	0.00%	**76***	**0.14**	0.00%
MA-50-0.8-5-5-5	**79***	**79***	-	**79***	8.48	0.00%	**79***	5.65	0.00%	**79***	**0.15**	0.00%
MA-100-0.2-5-5-1	**175***	**175***	-	177	19.18	1.14%	**175***	16.30	0.00%	**175***	**0.32**	0.00%
MA-100-0.2-5-5-2	**174***	**174***	-	**174***	22.63	0.00%	**174***	15.42	0.00%	**174***	**0.22**	0.00%
MA-100-0.2-5-5-3	**177***	**177***	-	**177***	24.85	0.00%	**177***	15.16	0.00%	**177***	**0.28**	0.00%
MA-100-0.2-5-5-4	**169***	**169***	-	**169***	24.28	0.00%	**169***	15.59	0.00%	**169***	**0.17**	0.00%
MA-100-0.2-5-5-5	**167***	**167***	-	168	22.93	0.60%	**167***	15.49	0.00%	**167***	**0.33**	0.00%
MA-100-0.5-5-5-1	**147***	**147***	-	**147***	28.93	0.00%	**147***	18.94	0.00%	**147***	**0.83**	0.00%
MA-100-0.5-5-5-2	**144***	**144***	-	**144***	25.12	0.00%	**144***	19.75	0.00%	**144***	**0.16**	0.00%
MA-100-0.5-5-5-3	**147***	**147***	-	**147***	23.05	0.00%	**147***	19.48	0.00%	**147***	**0.32**	0.00%
MA-100-0.5-5-5-4	**146***	**146***	-	**146***	27.62	0.00%	**146***	20.46	0.00%	**146***	**0.18**	0.00%
MA-100-0.5-5-5-5	**139***	**139***	-	**139***	25.02	0.00%	**139***	20.85	0.00%	**139***	**0.24**	0.00%
MA-100-0.8-5-5-1	**136***	**136***	-	**136***	29.24	0.00%	**136***	23.51	0.00%	**136***	**0.19**	0.00%
MA-100-0.8-5-5-2	**140***	**140***	-	**140***	27.93	0.00%	**140***	21.04	0.00%	**140***	**0.20**	0.00%
MA-100-0.8-5-5-3	**141***	**141***	-	**141***	25.46	0.00%	**141***	22.45	0.00%	**141***	**0.61**	0.00%
MA-100-0.8-5-5-4	**141***	**141***	-	**141***	27.18	0.00%	**141***	22.85	0.00%	**141***	**0.18**	0.00%
MA-100-0.8-5-5-5	**134***	**134***	-	**134***	28.11	0.00%	**134***	22.17	0.00%	**134***	**0.19**	0.00%
Average	**95.33**	**95.33**	-	95.47	11.59	0.11%	**95.33**	8.71	0.00%	**95.33**	**0.19**	0.00%

[1] Note: The results of re-GA are obtained by re-compiling these algorithms on our computer since the algorithm has not been applied on these instances.

Table B.2
Comparative results of KILS with re-MIP, re-GA and VNS on AMS instances with a time limit of 1800 seconds.

Instance	*BKVs*	re-MIP [12]		re-GA [12]			VNS [12]			KILS		
		f_{best}	*time*	f_{best}	*time*	*pg%*	f_{best}	*time*	*pg%*	f_{best}	*time*	*pg%*
AMS-75-0.2-10-50-1	**686***	**686***	-	**686***	5.00	0.00%	**686***	10.85	0.00%	**686***	**0.17**	0.00%
AMS-75-0.2-10-50-2	**770***	**770***	-	794	6.00	3.10%	**770***	10.03	0.00%	**770***	**0.27**	0.00%
AMS-75-0.2-10-50-3	**661***	**661***	-	**661***	6.00	0.00%	**661***	10.13	0.00%	**661***	**0.15**	0.00%
AMS-75-0.2-10-50-4	**703***	**703***	-	740	7.00	5.30%	**703***	10.71	0.00%	**703***	**0.25**	0.00%
AMS-75-0.2-10-50-5	**758***	**758***	-	779	6.00	2.80%	**758***	9.79	0.00%	**758***	**0.19**	0.00%
AMS-75-0.2-25-25-1	**498***	**498***	-	504	6.00	1.20%	**498***	10.24	0.00%	**498***	**0.20**	0.00%
AMS-75-0.2-25-25-2	**546***	**546***	-	**546***	6.00	0.00%	**546***	9.63	0.00%	**546***	**0.16**	0.00%
AMS-75-0.2-25-25-3	**518***	**518***	-	**518***	5.00	0.00%	**518***	9.50	0.00%	**518***	**0.16**	0.00%
AMS-75-0.2-25-25-4	**498***	**498***	-	**498***	6.00	0.00%	**498***	9.72	0.00%	**498***	**0.29**	0.00%
AMS-75-0.2-25-25-5	**513***	**513***	-	**513***	6.00	0.00%	**513***	9.89	0.00%	**513***	**0.14**	0.00%
AMS-75-0.2-50-10-1	**339***	**339***	-	**339***	6.00	0.00%	**339***	9.12	0.00%	**339***	**0.13**	0.00%
AMS-75-0.2-50-10-2	**382***	**382***	-	**382***	5.00	0.00%	**382***	8.30	0.00%	**382***	**0.15**	0.00%
AMS-75-0.2-50-10-3	**335***	**335***	-	341	5.00	1.80%	**335***	9.11	0.00%	**335***	**0.14**	0.00%
AMS-75-0.2-50-10-4	**333***	**333***	-	**333***	6.00	0.00%	**333***	8.82	0.00%	**333***	**0.14**	0.00%
AMS-75-0.2-50-10-5	**347***	**347***	-	**347***	6.00	0.00%	**347***	9.03	0.00%	**347***	**0.14**	0.00%
AMS-75-0.5-10-50-1	**581***	**581***	-	**581***	13.00	0.00%	**581***	13.09	0.00%	**581***	**0.28**	0.00%
AMS-75-0.5-10-50-2	**602***	**602***	-	**602***	11.00	0.00%	**602***	12.14	0.00%	**602***	**0.18**	0.00%
AMS-75-0.5-10-50-3	**545***	**545***	-	**545***	10.00	0.00%	**545***	13.34	0.00%	**545***	**0.16**	0.00%
AMS-75-0.5-10-50-4	**540***	**540***	-	**540***	10.00	0.00%	**540***	12.69	0.00%	**540***	**0.20**	0.00%
AMS-75-0.5-10-50-5	**519***	**519***	-	**519***	10.00	0.00%	**519***	12.69	0.00%	**519***	**0.17**	0.00%
AMS-75-0.5-25-25-1	**387***	**387***	-	**387***	10.00	0.00%	**387***	12.16	0.00%	**387***	**0.17**	0.00%
AMS-75-0.5-25-25-2	**384***	**384***	-	**384***	10.00	0.00%	**384***	11.54	0.00%	**384***	**0.16**	0.00%
AMS-75-0.5-25-25-3	**362***	**362***	-	**362***	10.00	0.00%	**362***	11.63	0.00%	**362***	**0.16**	0.00%
AMS-75-0.5-25-25-4	**366***	**366***	-	371	9.00	1.40%	**366***	11.25	0.00%	**366***	**0.21**	0.00%
AMS-75-0.5-25-25-5	**331***	**331***	-	**331***	10.00	0.00%	**331***	12.16	0.00%	**331***	**0.16**	0.00%
AMS-75-0.5-50-10-1	**240***	**240***	-	**240***	9.00	0.00%	**240***	10.45	0.00%	**240***	**0.14**	0.00%
AMS-75-0.5-50-10-2	**238***	**238***	-	**238***	9.00	0.00%	**238***	10.48	0.00%	**238***	**0.14**	0.00%
AMS-75-0.5-50-10-3	**215***	**215***	-	**215***	9.00	0.00%	**215***	10.49	0.00%	**215***	**0.13**	0.00%
AMS-75-0.5-50-10-4	**235***	**235***	-	**235***	9.00	0.00%	**235***	10.60	0.00%	**235***	**0.14**	0.00%
AMS-75-0.5-50-10-5	**206***	**206***	-	**206***	8.00	0.00%	**206***	10.95	0.00%	**206***	**0.15**	0.00%
AMS-75-0.8-10-50-1	**571***	**571***	-	**571***	16.00	0.00%	**571***	13.82	0.00%	**571***	**0.18**	0.00%
AMS-75-0.8-10-50-2	**520***	**520***	-	**520***	15.00	0.00%	**520***	14.18	0.00%	**520***	**0.18**	0.00%
AMS-75-0.8-10-50-3	**543***	**543***	-	**543***	15.00	0.00%	**543***	14.43	0.00%	**543***	**0.21**	0.00%
AMS-75-0.8-10-50-4	**571***	**571***	-	**571***	15.00	0.00%	**571***	13.93	0.00%	**571***	**0.26**	0.00%
AMS-75-0.8-10-50-5	**509***	**509***	-	**509***	17.00	0.00%	**509***	14.39	0.00%	**509***	**0.16**	0.00%
AMS-75-0.8-25-25-1	**357***	**357***	-	**357***	15.00	0.00%	**357***	13.27	0.00%	**357***	**0.17**	0.00%
AMS-75-0.8-25-25-2	**338***	**338***	-	**338***	15.00	0.00%	**338***	13.24	0.00%	**338***	**0.16**	0.00%
AMS-75-0.8-25-25-3	**323***	**323***	-	**323***	13.00	0.00%	**323***	12.97	0.00%	**323***	**0.16**	0.00%
AMS-75-0.8-25-25-4	**345***	**345***	-	**345***	13.00	0.00%	**345***	13.70	0.00%	**345***	**0.16**	0.00%
AMS-75-0.8-25-25-5	**311***	**311***	-	**311***	15.00	0.00%	**311***	13.47	0.00%	**311***	**0.17**	0.00%
AMS-75-0.8-50-10-1	**182***	**182***	-	**182***	14.00	0.00%	**182***	12.51	0.00%	**182***	**0.14**	0.00%
AMS-75-0.8-50-10-2	**188***	**188***	-	**188***	11.00	0.00%	**188***	12.31	0.00%	**188***	**0.14**	0.00%
AMS-75-0.8-50-10-3	**191***	**191***	-	**191***	11.00	0.00%	**191***	11.95	0.00%	**191***	**0.14**	0.00%
AMS-75-0.8-50-10-4	**196***	**196***	-	**196***	12.00	0.00%	**196***	12.05	0.00%	**196***	**0.14**	0.00%
AMS-75-0.8-50-10-5	**192***	**192***	-	**192***	15.00	0.00%	**192***	12.19	0.00%	**192***	**0.14**	0.00%

(Continued)

Table B.2
(continued)

Instance	*BKVs*	re-MIP [12]		re-GA [12]			VNS [12]			KILS		
		f_{best}	*time*	f_{best}	*time*	*pg%*	f_{best}	*time*	*pg%*	f_{best}	*time*	*pg%*
AMS-100-0.2-10-50-1	**873***	**873***	**-**	**873***	12.00	0.00%	**873***	19.33	0.00%	**873***	**0.64**	0.00%
AMS-100-0.2-10-50-2	**944***	**944***	**-**	**944***	13.00	0.00%	**944***	17.70	0.00%	**944***	**0.34**	0.00%
AMS-100-0.2-10-50-3	**878***	**878***	**-**	**878***	11.00	0.00%	**878***	18.01	0.00%	**878***	**0.49**	0.00%
AMS-100-0.2-10-50-4	**837***	**837***	**-**	**837***	11.00	0.00%	**837***	18.30	0.00%	**837***	**0.33**	0.00%
AMS-100-0.2-10-50-5	**840***	**840***	**-**	870	12.00	3.60%	**840***	17.79	0.00%	**840***	**0.48**	0.00%
AMS-100-0.2-25-25-1	**591***	**591***	**-**	**591***	12.00	0.00%	**591***	18.15	0.00%	**591***	**0.18**	0.00%
AMS-100-0.2-25-25-2	**653***	**653***	**-**	655	11.00	0.30%	**653***	15.76	0.00%	**653***	**0.26**	0.00%
AMS-100-0.2-25-25-3	**612***	**612***	**-**	616	12.00	0.70%	615	16.47	0.50%	**612***	**0.63**	0.00%
AMS-100-0.2-25-25-4	**552***	**552***	**-**	**552***	11.00	0.00%	**552***	15.92	0.00%	**552***	**0.23**	0.00%
AMS-100-0.2-25-25-5	**606***	**606***	**-**	607	12.00	0.20%	**606***	16.79	0.00%	**606***	**0.55**	0.00%
AMS-100-0.2-50-10-1	**418***	**418***	**-**	420	12.00	0.50%	**418***	15.18	0.00%	**418***	**0.19**	0.00%
AMS-100-0.2-50-10-2	**447***	**447***	**-**	456	11.00	2.00%	**447***	14.26	0.00%	**447***	**0.19**	0.00%
AMS-100-0.2-50-10-3	**419***	**419***	**-**	**419***	11.00	0.00%	**419***	15.21	0.00%	**419***	**0.16**	0.00%
AMS-100-0.2-50-10-4	**403***	**403***	**-**	410	12.00	1.70%	**403***	14.97	0.00%	**403***	**0.18**	0.00%
AMS-100-0.2-50-10-5	**375***	**375***	**-**	379	13.00	1.10%	**375***	15.74	0.00%	**375***	**0.20**	0.00%
AMS-100-0.5-10-50-1	**743***	**743***	**-**	749	26.00	0.80%	**743***	22.39	0.00%	**743***	**1.43**	0.00%
AMS-100-0.5-10-50-2	**698***	**698***	**-**	700	25.00	0.30%	**698***	21.47	0.00%	**698***	**0.88**	0.00%
AMS-100-0.5-10-50-3	**699***	**699***	**-**	718	24.00	2.70%	**699***	22.06	0.00%	**699***	**1.89**	0.00%
AMS-100-0.5-10-50-4	**726***	**726***	**-**	**726***	26.00	0.00%	**726***	22.15	0.00%	**726***	**0.37**	0.00%
AMS-100-0.5-10-50-5	**702***	**702***	**-**	**702***	25.00	0.00%	**702***	22.44	0.00%	**702***	**0.28**	0.00%
AMS-100-0.5-25-25-1	**461***	**461***	**-**	**461***	25.00	0.00%	**461***	20.31	0.00%	**461***	**0.19**	0.00%
AMS-100-0.5-25-25-2	**437***	**437***	**-**	**437***	19.00	0.00%	**437***	21.27	0.00%	**437***	**0.18**	0.00%
AMS-100-0.5-25-25-3	**434***	**434***	**-**	**434***	22.00	0.00%	**434***	22.22	0.00%	**434***	**0.29**	0.00%
AMS-100-0.5-25-25-4	**482***	**482***	**-**	**482***	25.00	0.00%	**482***	20.52	0.00%	**482***	**0.43**	0.00%
AMS-100-0.5-25-25-5	**456***	**456***	**-**	457	23.00	0.20%	**456***	20.36	0.00%	**456***	**0.30**	0.00%
AMS-100-0.5-50-10-1	**260***	**260***	**-**	**260***	22.00	0.00%	**260***	18.15	0.00%	**260***	**0.16**	0.00%
AMS-100-0.5-50-10-2	**271***	**271***	**-**	**271***	21.00	0.00%	**271***	18.86	0.00%	**271***	**0.16**	0.00%
AMS-100-0.5-50-10-3	**283***	**283***	**-**	**283***	21.00	0.00%	**283***	20.55	0.00%	**283***	**0.17**	0.00%
AMS-100-0.5-50-10-4	**291***	**291***	**-**	**291***	22.00	0.00%	**291***	18.30	0.00%	**291***	**0.16**	0.00%
AMS-100-0.5-50-10-5	**269***	**269***	**-**	**269***	21.00	0.00%	**269***	18.48	0.00%	**269***	**0.17**	0.00%
AMS-100-0.8-10-50-1	**730**	**730**	**-**	**730**	39.00	0.00%	**730**	25.15	0.00%	**730**	**0.73**	0.00%
AMS-100-0.8-10-50-2	**683***	**683***	**-**	**683***	37.00	0.00%	**683***	23.51	0.00%	**683***	**0.52**	0.00%
AMS-100-0.8-10-50-3	**718***	**718***	**-**	**718***	37.00	0.00%	**718***	24.80	0.00%	**718***	**0.23**	0.00%
AMS-100-0.8-10-50-4	**709***	**709***	**-**	**709***	41.00	0.00%	**709***	28.08	0.00%	**709***	**0.70**	0.00%
AMS-100-0.8-10-50-5	**700***	**700***	**-**	704	39.00	0.60%	**700***	26.15	0.00%	**700***	**0.36**	0.00%
AMS-100-0.8-25-25-1	**442***	**442***	**-**	**442***	40.00	0.00%	**442***	22.49	0.00%	**442***	**0.27**	0.00%
AMS-100-0.8-25-25-2	**430***	**430***	**-**	**430***	32.00	0.00%	**430***	23.40	0.00%	**430***	**0.22**	0.00%
AMS-100-0.8-25-25-3	**426***	**426***	**-**	**426***	36.00	0.00%	**426***	23.26	0.00%	**426***	**0.19**	0.00%
AMS-100-0.8-25-25-4	**428***	**428***	**-**	**428***	35.00	0.00%	**428***	22.73	0.00%	**428***	**0.24**	0.00%
AMS-100-0.8-25-25-5	**432***	**432***	**-**	**432***	42.00	0.00%	**432***	23.04	0.00%	**432***	**0.39**	0.00%
AMS-100-0.8-50-10-1	**259***	**259***	**-**	**259***	32.00	0.00%	**259***	21.01	0.00%	**259***	**0.18**	0.00%
AMS-100-0.8-50-10-2	**246***	**246***	**-**	**246***	9.00	0.00%	**246***	20.64	0.00%	**246***	**0.16**	0.00%
AMS-100-0.8-50-10-3	**238***	**238***	**-**	**238***	34.00	0.00%	**238***	21.63	0.00%	**238***	**0.16**	0.00%
AMS-100-0.8-50-10-4	**253***	**253***	**-**	**253***	34.00	0.00%	**253***	22.23	0.00%	**253***	**0.18**	0.00%
AMS-100-0.8-50-10-5	**248***	**248***	**-**	**248***	31.00	0.00%	**248***	33.60	0.00%	**248***	**0.17**	0.00%

(Continued)

Table B.2
(continued)

Instance	*BKVs*	re-MIP [12]		re-GA [12]			VNS [12]			KILS		
		f_{best}	*time*	f_{best}	*time*	*pg%*	f_{best}	*time*	*pg%*	f_{best}	*time*	*pg%*
AMS-125-0.2-10-50-1	**1026**	**1026**	-	**1026**	24.00	0.00%	**1026**	28.80	0.00%	**1026**	**0.69**	0.00%
AMS-125-0.2-10-50-2	**1038**	**1038**	-	**1038**	22.00	0.00%	**1038**	28.10	0.00%	**1038**	**0.52**	0.00%
AMS-125-0.2-10-50-3	**935***	**935***	-	947	23.00	1.28%	**935***	24.50	1.28%	**935***	**0.39**	0.00%
AMS-125-0.2-10-50-4	**1050**	**1050**	-	1051	21.00	0.10%	1064	26.10	0.10%	**1050**	**0.48**	0.00%
AMS-125-0.2-10-50-5	**974**	**974**	-	975	25.00	0.10%	**974**	25.70	0.10%	**974**	**0.53**	0.00%
AMS-125-0.2-25-25-1	**720***	**720***	-	**720***	26.00	0.00%	**720***	25.70	0.00%	**720***	**0.39**	0.00%
AMS-125-0.2-25-25-2	**746***	**746***	-	748	24.00	0.27%	**746***	22.80	0.27%	**746***	**0.47**	0.00%
AMS-125-0.2-25-25-3	**715***	**715***	-	717	21.00	0.28%	**715***	26.90	0.28%	**715***	**0.40**	0.00%
AMS-125-0.2-25-25-4	**701***	**701***	-	705	22.00	0.57%	**701***	25.40	0.57%	**701***	**0.73**	0.00%
AMS-125-0.2-25-25-5	**684***	**684***	-	697	23.00	1.90%	685	23.40	1.90%	**684***	**1.61**	0.00%
AMS-125-0.2-50-10-1	**455***	**455***	-	**455***	21.00	0.00%	**455***	23.20	0.00%	**455***	**0.14**	0.00%
AMS-125-0.2-50-10-2	**477***	**477***	-	**477***	23.00	0.00%	**477***	22.40	0.00%	**477***	**0.14**	0.00%
AMS-125-0.2-50-10-3	**490***	**490***	-	**490***	21.00	0.00%	**490***	22.00	0.00%	**490***	**0.17**	0.00%
AMS-125-0.2-50-10-4	**467***	**467***	-	**467***	23.00	0.00%	**467***	22.50	0.00%	**467***	**0.23**	0.00%
AMS-125-0.2-50-10-5	**457***	**457***	-	459	24.00	0.44%	**457***	23.10	0.44%	**457***	**0.17**	0.00%
AMS-125-0.5-10-50-1	**817**	**817**	-	**817**	41.00	0.00%	**817**	33.80	0.00%	**817**	**0.27**	0.00%
AMS-125-0.5-10-50-2	**815**	**815**	-	**815**	45.00	0.00%	**815**	34.40	0.00%	**815**	**0.23**	0.00%
AMS-125-0.5-10-50-3	**836**	**836**	-	872	45.00	4.31%	**836**	35.80	4.31%	**836**	**0.63**	0.00%
AMS-125-0.5-10-50-4	**867**	**867**	-	**867**	55.00	0.00%	**867**	33.70	0.00%	**867**	**0.35**	0.00%
AMS-125-0.5-10-50-5	**867**	**867**	-	**867**	55.00	0.00%	**867**	28.60	0.00%	**867**	**1.23**	0.00%
AMS-125-0.5-25-25-1	**566**	**566**	-	**566**	48.00	0.00%	**566**	32.00	0.00%	**566**	**0.34**	0.00%
AMS-125-0.5-25-25-2	**533***	**533***	-	**533***	48.00	0.00%	**533***	28.60	0.00%	**533***	**0.58**	0.00%
AMS-125-0.5-25-25-3	**538***	**538***	-	**538***	49.00	0.00%	**538***	30.40	0.00%	**538***	**0.41**	0.00%
AMS-125-0.5-25-25-4	**552**	**552**	-	**552**	53.00	0.00%	**552**	31.10	0.00%	**552**	**0.41**	0.00%
AMS-125-0.5-25-25-5	**545**	**545**	-	548	48.00	0.55%	**545**	31.70	0.55%	**545**	**0.89**	0.00%
AMS-125-0.5-50-10-1	**334***	**334***	-	**334***	40.00	0.00%	**334***	29.30	0.00%	**334***	**0.22**	0.00%
AMS-125-0.5-50-10-2	**330***	**330***	-	**330***	38.00	0.00%	**330***	28.50	0.00%	**330***	**0.15**	0.00%
AMS-125-0.5-50-10-3	**315***	**315***	-	**315***	49.00	0.00%	**315***	26.50	0.00%	**315***	**0.14**	0.00%
AMS-125-0.5-50-10-4	**316***	**316***	-	**316***	51.00	0.00%	**316***	29.20	0.00%	**316***	**0.15**	0.00%
AMS-125-0.5-50-10-5	**311***	**311***	-	**311***	40.00	0.00%	**311***	28.90	0.00%	**311***	**0.20**	0.00%
AMS-125-0.8-10-50-1	**793**	**793**	-	**793**	78.00	0.00%	**793**	39.50	0.00%	**793**	**0.20**	0.00%
AMS-125-0.8-10-50-2	**845**	**845**	-	**845**	72.00	0.00%	**845**	40.40	0.00%	**845**	**0.54**	0.00%
AMS-125-0.8-10-50-3	**787**	**787**	-	**787**	74.00	0.00%	**787**	41.80	0.00%	**787**	**0.24**	0.00%
AMS-125-0.8-10-50-4	**777**	**777**	-	**777**	83.00	0.00%	**777**	40.80	0.00%	**777**	**0.29**	0.00%
AMS-125-0.8-10-50-5	**813**	**813**	-	**813**	77.00	0.00%	**813**	40.70	0.00%	**813**	**0.66**	0.00%
AMS-125-0.8-25-25-1	**508***	**508***	-	510	69.00	0.39%	**508***	37.40	0.39%	**508***	**0.54**	0.00%
AMS-125-0.8-25-25-2	**498***	**498***	-	**498***	65.00	0.00%	**498***	38.30	0.00%	**498***	**0.36**	0.00%
AMS-125-0.8-25-25-3	**513**	**513**	-	**513**	77.00	0.00%	**513**	37.00	0.00%	**513**	**0.36**	0.00%
AMS-125-0.8-25-25-4	**493***	**493***	-	**493***	75.00	0.00%	**493***	38.90	0.00%	**493***	**0.20**	0.00%
AMS-125-0.8-25-25-5	**504**	**504**	-	**504**	76.00	0.00%	**504**	38.30	0.00%	**504**	**0.89**	0.00%
AMS-125-0.8-50-10-1	**307***	**307***	-	**307***	64.00	0.00%	**307***	33.60	0.00%	**307***	**0.17**	0.00%
AMS-125-0.8-50-10-2	**296***	**296***	-	**296***	57.00	0.00%	**296***	37.60	0.00%	**296***	**0.16**	0.00%
AMS-125-0.8-50-10-3	**294***	**294***	-	**294***	71.00	0.00%	**294***	33.60	0.00%	**294***	**0.18**	0.00%
AMS-125-0.8-50-10-4	**270***	**270***	-	**270***	86.00	0.00%	**270***	34.80	0.00%	**270***	**0.15**	0.00%
AMS-125-0.8-50-10-5	**278***	**278***	-	**278***	77.00	0.00%	**278***	33.60	0.00%	**278***	**0.16**	0.00%
Average	**518.74**	**518.74**	-	520.73	26.93	0.30%	518.87	21.03	0.08%	**518.74**	**0.32**	0.00%

Table B.3
Comparative results of KILS with re-MIP, re-GA and VNS on NEW instances with a time limit of 1800 seconds.

Instance	*BKVs*	re-MIP [12]		re-GA [12]			VNS [12]			KILS		
		f_{best}	*time*	f_{best}	*time*	*pg%*	f_{best}	*time*	*pg%*	f_{best}	*time*	*pg%*
NEW-250-0.2-10-50-1	1662	1933	-	1662	122.44	0.00%	1703	111.20	2.47%	**1637**	**8.64**	-1.50%
NEW-250-0.2-10-50-2	1728	2029	-	1768	125.38	2.31%	1728	108.40	0.00%	**1718**	**11.46**	-0.58%
NEW-250-0.2-10-50-3	1754	1943	-	1754	143.58	0.00%	1769	96.60	0.86%	**1723**	**8.03**	-1.77%
NEW-250-0.2-10-50-4	1635	1799	-	1655	121.02	1.22%	1635	100.80	0.00%	**1632**	**48.11**	-0.18%
NEW-250-0.2-10-50-5	1737	1905	-	1739	135.64	0.12%	1737	103.60	0.00%	**1698**	**8.91**	-2.25%
NEW-250-0.2-25-25-1	**1144**	1244	-	1157	141.54	1.14%	**1144**	102.70	0.00%	**1144**	**5.53**	0.00%
NEW-250-0.2-25-25-2	**1135**	1210	-	1139	112.92	0.35%	**1135**	101.90	0.00%	**1135**	**1.72**	0.00%
NEW-250-0.2-25-25-3	**1132**	1197	-	**1132**	120.07	0.00%	**1132**	109.20	0.00%	**1132**	**1.58**	0.00%
NEW-250-0.2-25-25-4	1162	1228	-	1162	103.53	0.00%	1162	107.70	0.00%	**1149**	**9.95**	-1.12%
NEW-250-0.2-25-25-5	1147	1285	-	1149	124.22	0.17%	1147	111.30	0.00%	**1144**	**12.61**	-0.26%
NEW-250-0.2-50-10-1	749	763	-	749	101.84	0.00%	749	108.20	0.00%	**743**	**0.96**	-0.80%
NEW-250-0.2-50-10-2	**708**	715	-	**708**	110.43	0.00%	**708**	102.40	0.00%	**708**	**0.72**	0.00%
NEW-250-0.2-50-10-3	**719**	720	-	**719**	103.70	0.00%	**719**	104.30	0.00%	**719**	**0.54**	0.00%
NEW-250-0.2-50-10-4	**680**	**680**	-	**680**	107.74	0.00%	**680**	117.90	0.00%	**680**	**0.31**	0.00%
NEW-250-0.2-50-10-5	**758**	765	-	764	127.89	0.79%	**758**	105.90	0.00%	**758**	**1.01**	0.00%
NEW-250-0.5-10-50-1	1431	1696	-	1431	175.70	0.00%	1431	195.20	0.00%	**1410**	**16.10**	-1.47%
NEW-250-0.5-10-50-2	**1468**	1834	-	1471	168.53	0.20%	**1468**	223.50	0.00%	**1468**	**13.91**	0.00%
NEW-250-0.5-10-50-3	**1417**	1502	-	**1417**	205.66	0.00%	**1417**	234.80	0.00%	**1417**	**10.51**	0.00%
NEW-250-0.5-10-50-4	**1492**	1716	-	1518	183.18	1.74%	**1492**	208.80	0.00%	**1492**	**3.87**	0.00%
NEW-250-0.5-10-50-5	**1474**	1798	-	**1474**	175.63	0.00%	1483	205.00	0.61%	**1474**	**12.95**	0.00%
NEW-250-0.5-25-25-1	**900**	1018	-	914	182.00	1.56%	**900**	204.40	0.00%	**900**	**4.87**	0.00%
NEW-250-0.5-25-25-2	**921**	1044	-	939	171.87	1.95%	**921**	195.40	0.00%	**921**	**5.53**	0.00%
NEW-250-0.5-25-25-3	**896**	1100	-	**896**	164.35	0.00%	**896**	195.00	0.00%	**896**	**5.42**	0.00%
NEW-250-0.5-25-25-4	**956**	1082	-	957	189.52	0.10%	**956**	237.60	0.00%	**956**	**22.23**	0.00%
NEW-250-0.5-25-25-5	**914**	1060	-	**914**	210.20	0.00%	915	219.40	0.11%	**914**	**9.73**	0.00%
NEW-250-0.5-50-10-1	**533**	535	-	**533**	161.83	0.00%	**533**	188.10	0.00%	**533**	**0.39**	0.00%
NEW-250-0.5-50-10-2	**554**	588	-	**554**	189.04	0.00%	**554**	203.20	0.00%	**554**	**0.96**	0.00%
NEW-250-0.5-50-10-3	**556**	557	-	**556**	166.84	0.00%	**556**	173.70	0.00%	**556**	**0.92**	0.00%
NEW-250-0.5-50-10-4	**558**	569	-	**558**	206.30	0.00%	**558**	181.70	0.00%	**558**	**0.45**	0.00%
NEW-250-0.5-50-10-5	**532**	**532**	-	**532**	143.61	0.00%	**532**	186.10	0.00%	**532**	**0.31**	0.00%
NEW-250-0.8-10-50-1	**1410**	1627	-	1435	275.55	1.77%	**1410**	304.70	0.00%	**1410**	**12.91**	0.00%
NEW-250-0.8-10-50-2	1401	1547	-	1401	272.07	0.00%	1401	275.30	0.00%	**1396**	**5.68**	-0.36%
NEW-250-0.8-10-50-3	**1444**	1748	-	**1444**	313.13	0.00%	**1444**	313.80	0.00%	**1444**	**6.70**	0.00%
NEW-250-0.8-10-50-4	**1420**	1738	-	**1420**	293.97	0.00%	**1420**	308.70	0.00%	**1420**	**4.89**	0.00%
NEW-250-0.8-10-50-5	**1430**	1668	-	**1430**	269.13	0.00%	**1430**	290.00	0.00%	**1430**	**5.73**	0.00%
NEW-250-0.8-25-25-1	**919**	1006	-	928	295.71	0.98%	**919**	295.20	0.00%	**919**	**1.93**	0.00%
NEW-250-0.8-25-25-2	**835**	931	-	**835**	286.39	0.00%	**835**	293.30	0.00%	**835**	**1.24**	0.00%
NEW-250-0.8-25-25-3	**914**	1072	-	**914**	295.34	0.00%	919	271.50	0.55%	**914**	**17.61**	0.00%
NEW-250-0.8-25-25-4	**846**	903	-	**846**	268.14	0.00%	**846**	289.60	0.00%	**846**	**1.45**	0.00%
NEW-250-0.8-25-25-5	**853**	911	-	**853**	280.14	0.00%	**853**	285.60	0.00%	**853**	**9.44**	0.00%
NEW-250-0.8-50-10-1	**489**	490	-	**489**	295.28	0.00%	**489**	274.80	0.00%	**489**	**3.53**	0.00%
NEW-250-0.8-50-10-2	**507**	522	-	**507**	254.11	0.00%	**507**	271.50	0.00%	**507**	**1.52**	0.00%
NEW-250-0.8-50-10-3	**498**	507	-	499	323.87	0.20%	**498**	274.80	0.00%	**498**	**1.08**	0.00%
NEW-250-0.8-50-10-4	**489**	519	-	494	287.05	1.02%	**489**	289.00	0.00%	**489**	**2.12**	0.00%
NEW-250-0.8-50-10-5	**482**	508	-	**482**	339.30	0.00%	**482**	253.00	0.00%	**482**	**0.39**	0.00%

(Continued)

Table B.3
(continued)

Instance	*BKVs*	re-MIP [12]		re-GA [12]			VNS [12]			KILS		
		f_{best}	*time*	f_{best}	*time*	*pg%*	f_{best}	*time*	*pg%*	f_{best}	*time*	*pg%*
NEW-500-0.2-10-50-1	**2893**	3837	-	**2893**	728.03	0.00%	2968	650.84	2.47%	**2893**	**125.22**	0.00%
NEW-500-0.2-10-50-2	2937	3736	-	2937	806.45	0.00%	2947	766.95	0.00%	**2848**	**751.86**	-3.03%
NEW-500-0.2-10-50-3	3082	3839	-	3090	940.17	0.26%	3082	737.20	0.86%	**2978**	**174.51**	-3.37%
NEW-500-0.2-10-50-4	2964	3705	-	2964	904.48	0.00%	3063	584.93	0.00%	**2930**	**138.43**	-1.15%
NEW-500-0.2-10-50-5	2928	3965	-	2942	1069.15	0.48%	2928	574.73	0.00%	**2869**	**376.30**	-2.02%
NEW-500-0.2-25-25-1	1884	2383	-	1918	667.69	1.80%	1884	835.16	0.00%	**1861**	**265.76**	-1.22%
NEW-500-0.2-25-25-2	1931	2474	-	1931	803.81	0.00%	1941	603.05	0.00%	**1916**	**512.44**	-0.78%
NEW-500-0.2-25-25-3	1898	2225	-	1914	913.94	0.84%	1898	664.47	0.00%	**1867**	**160.48**	-1.63%
NEW-500-0.2-25-25-4	**1861**	2368	-	**1861**	774.70	0.00%	1937	603.66	0.00%	**1861**	**334.69**	0.00%
NEW-500-0.2-25-25-5	1861	2285	-	1904	926.24	2.31%	1861	875.06	0.00%	**1855**	**352.25**	-0.32%
NEW-500-0.2-50-10-1	**1128**	1256	-	1133	853.24	0.44%	**1128**	667.66	0.00%	**1128**	**15.18**	0.00%
NEW-500-0.2-50-10-2	**1143**	1263	-	1159	831.21	1.40%	**1143**	719.89	0.00%	**1143**	**31.61**	0.00%
NEW-500-0.2-50-10-3	**1108**	1288	-	**1108**	627.85	0.00%	**1108**	633.61	0.00%	**1108**	**6.78**	0.00%
NEW-500-0.2-50-10-4	**1145**	1246	-	1149	708.18	0.35%	**1145**	715.77	0.00%	**1145**	**9.10**	0.00%
NEW-500-0.2-50-10-5	1186	1301	-	1186	821.28	0.00%	1188	747.80	0.00%	**1175**	**140.38**	-0.93%
NEW-500-0.5-10-50-1	2590	3864	-	2614	1031.47	0.93%	2590	1560.09	0.00%	**2522**	**650.08**	-2.63%
NEW-500-0.5-10-50-2	2542	4551	-	2631	1089.09	3.50%	2542	1624.32	0.00%	**2534**	**279.52**	-0.31%
NEW-500-0.5-10-50-3	2547	4759	-	2552	1011.45	0.20%	2547	1432.79	0.00%	**2543**	**487.11**	-0.16%
NEW-500-0.5-10-50-4	2584	3220	-	2584	1112.37	0.00%	2596	1140.71	0.00%	**2551**	**621.63**	-1.28%
NEW-500-0.5-10-50-5	2539	3691	-	2539	978.20	0.00%	2539	821.39	0.61%	**2522**	**794.30**	-0.67%
NEW-500-0.5-25-25-1	1543	2164	-	1566	1064.91	1.49%	1543	1059.86	0.00%	**1542**	**372.22**	-0.06%
NEW-500-0.5-25-25-2	1574	2137	-	1574	1266.94	0.00%	1594	1069.64	0.00%	**1559**	**197.52**	-0.95%
NEW-500-0.5-25-25-3	1547	2121	-	1568	1054.71	1.36%	1547	1093.08	0.00%	**1546**	**93.74**	-0.06%
NEW-500-0.5-25-25-4	**1564**	2210	-	**1564**	1029.50	0.00%	1566	986.54	0.00%	**1564**	**115.81**	0.00%
NEW-500-0.5-25-25-5	1567	2343	-	1567	1083.46	0.00%	1570	998.63	0.11%	**1566**	**969.76**	-0.06%
NEW-500-0.5-50-10-1	**919**	1036	-	**919**	961.30	0.00%	**919**	785.72	0.00%	**919**	**10.82**	0.00%
NEW-500-0.5-50-10-2	**911**	1067	-	933	874.65	2.41%	**911**	761.01	0.00%	**911**	**11.58**	0.00%
NEW-500-0.5-50-10-3	**906**	1020	-	**906**	850.46	0.00%	**906**	808.49	0.00%	**906**	**32.05**	0.00%
NEW-500-0.5-50-10-4	**904**	1091	-	**904**	995.52	0.00%	**904**	851.96	0.00%	**904**	**41.49**	0.00%
NEW-500-0.5-50-10-5	**932**	1063	-	938	945.76	0.64%	**932**	986.16	0.00%	**932**	**20.38**	0.00%
NEW-500-0.8-10-50-1	**2500**	3385	-	**2500**	1479.92	0.00%	**2500**	1465.99	0.00%	**2500**	**109.22**	0.00%
NEW-500-0.8-10-50-2	2497	3492	-	2497	1524.66	0.00%	2499	1631.88	0.00%	**2483**	**99.39**	-0.56%
NEW-500-0.8-10-50-3	**2466**	3384	-	2513	1729.63	1.91%	**2466**	1588.32	0.00%	**2466**	**430.55**	0.00%
NEW-500-0.8-10-50-4	2511	5055	-	2544	1769.65	1.31%	2511	1552.92	0.00%	**2509**	**134.28**	-0.08%
NEW-500-0.8-10-50-5	**2497**	4581	-	2538	1790.36	1.64%	**2497**	1450.32	0.00%	**2497**	**204.15**	0.00%
NEW-500-0.8-25-25-1	1490	2004	-	1490	1635.55	0.00%	1490	1326.62	0.00%	**1480**	**855.17**	-0.67%
NEW-500-0.8-25-25-2	1505	1972	-	1509	1394.27	0.27%	1505	1523.79	0.00%	**1500**	**133.69**	-0.33%
NEW-500-0.8-25-25-3	**1470**	2048	-	**1470**	1693.75	0.00%	1471	1332.70	0.55%	**1470**	**107.36**	0.00%
NEW-500-0.8-25-25-4	**1489**	1986	-	**1489**	1532.75	0.00%	**1489**	1369.60	0.00%	**1489**	**78.90**	0.00%
NEW-500-0.8-25-25-5	**1496**	2055	-	1507	1469.24	0.74%	**1496**	1534.04	0.00%	**1496**	**527.41**	0.00%
NEW-500-0.8-50-10-1	**871**	1021	-	**871**	1368.94	0.00%	**871**	1411.70	0.00%	**871**	**23.34**	0.00%
NEW-500-0.8-50-10-2	**853**	968	-	**853**	1611.79	0.00%	**853**	1406.58	0.00%	**853**	**13.42**	0.00%
NEW-500-0.8-50-10-3	**855**	1002	-	**855**	1607.03	0.00%	**855**	1426.59	0.00%	**855**	**17.19**	0.00%
NEW-500-0.8-50-10-4	869	985	-	871	1510.68	0.23%	869	1254.65	0.00%	**867**	**48.05**	-0.23%
NEW-500-0.8-50-10-5	**872**	933	-	**872**	1313.77	0.00%	**872**	1421.05	0.00%	**872**	**21.61**	0.00%

(Continued)

Table B.3
(continued)

Instance	*BKVs*	re-MIP [12]		re-GA [12]			VNS [12]			KILS		
		f_{best}	*time*	f_{best}	*time*	*pg%*	f_{best}	*time*	*pg%*	f_{best}	*time*	*pg%*
NEW-1000-0.2-10-50-1	4990	10851	-	4990	1844.62	0.00%	5106	1802.92	2.32%	**4875**	**396.92**	-2.30%
NEW-1000-0.2-10-50-2	5233	9687	-	5233	1911.96	0.00%	5316	1806.68	1.59%	**5049**	**1172.23**	-3.52%
NEW-1000-0.2-10-50-3	5044	9995	-	5196	2030.31	3.01%	5044	1800.12	0.00%	**4962**	**665.97**	-1.63%
NEW-1000-0.2-10-50-4	5079	9760	-	5079	1901.64	0.00%	5321	1803.50	4.76%	**4981**	**742.05**	-1.93%
NEW-1000-0.2-10-50-5	5098	10617	-	5098	1817.69	0.00%	5260	1802.36	3.18%	**4988**	**1556.61**	-2.16%
NEW-1000-0.2-25-25-1	3167	5340	-	3167	1903.54	0.00%	3233	1803.17	2.08%	**3122**	**636.26**	-1.42%
NEW-1000-0.2-25-25-2	3163	5845	-	3179	1954.87	0.51%	3163	1804.58	0.00%	**3127**	**768.29**	-1.14%
NEW-1000-0.2-25-25-3	3148	5003	-	3159	1802.92	0.35%	3148	1800.66	0.00%	**3106**	**892.91**	-1.33%
NEW-1000-0.2-25-25-4	3227	4863	-	3227	1839.07	0.00%	3268	1802.43	1.27%	**3147**	**736.78**	-2.48%
NEW-1000-0.2-25-25-5	3234	5349	-	3285	1848.57	1.58%	3234	1800.96	0.00%	**3151**	**981.56**	-2.57%
NEW-1000-0.2-50-10-1	**1907**	2419	-	**1907**	1876.57	0.00%	1930	1807.56	1.21%	**1907**	**598.98**	0.00%
NEW-1000-0.2-50-10-2	1882	2275	-	1882	1826.26	0.00%	1890	1803.21	0.43%	**1876**	**533.31**	-0.32%
NEW-1000-0.2-50-10-3	1899	2401	-	1899	1851.74	0.00%	1921	1820.84	1.16%	**1890**	**104.14**	-0.47%
NEW-1000-0.2-50-10-4	1914	2281	-	1914	1842.00	0.00%	1916	1801.58	0.10%	**1908**	**350.41**	-0.31%
NEW-1000-0.2-50-10-5	1934	2302	-	1934	1883.87	0.00%	1936	1807.26	0.10%	**1917**	**569.95**	-0.88%
NEW-1000-0.5-10-50-1	4483	n/a	-	4869	1817.82	8.61%	4483	1817.88	0.00%	**4363**	**349.51**	-2.68%
NEW-1000-0.5-10-50-2	4462	n/a	-	4807	1814.33	7.73%	4462	1802.64	0.00%	**4405**	**1085.69**	-1.28%
NEW-1000-0.5-10-50-3	4567	n/a	-	5015	1819.77	9.81%	4567	1809.09	0.00%	**4428**	**223.83**	-3.04%
NEW-1000-0.5-10-50-4	4480	n/a	-	4967	1804.76	10.87%	4480	1847.11	0.00%	**4386**	**876.22**	-2.10%
NEW-1000-0.5-10-50-5	4469	n/a	-	4862	1805.86	8.79%	4469	1812.15	0.00%	**4364**	**751.38**	-2.35%
NEW-1000-0.5-25-25-1	2743	n/a	-	2961	1822.59	7.95%	2743	1826.69	0.00%	**2712**	**1565.19**	-1.13%
NEW-1000-0.5-25-25-2	2771	n/a	-	2905	1811.76	4.84%	2771	1814.80	0.00%	**2646**	**741.23**	-4.51%
NEW-1000-0.5-25-25-3	2783	n/a	-	2923	1815.74	5.03%	2783	1813.17	0.00%	**2698**	**813.65**	-3.05%
NEW-1000-0.5-25-25-4	2757	n/a	-	2942	1811.96	6.71%	2757	1800.03	0.00%	**2694**	**573.76**	-2.29%
NEW-1000-0.5-25-25-5	2752	n/a	-	2946	1818.25	7.05%	2752	1800.10	0.00%	**2662**	**1053.23**	-3.27%
NEW-1000-0.5-50-10-1	**1612**	n/a	-	1699	1810.50	5.40%	**1612**	1801.31	0.00%	**1612**	**623.61**	0.00%
NEW-1000-0.5-50-10-2	1631	n/a	-	1697	1813.97	4.05%	1631	1807.96	0.00%	**1604**	**489.38**	-1.66%
NEW-1000-0.5-50-10-3	**1625**	n/a	-	1711	1819.20	5.29%	**1625**	1801.26	0.00%	**1625**	**676.47**	0.00%
NEW-1000-0.5-50-10-4	1629	n/a	-	1710	1805.03	4.97%	1629	1835.95	0.00%	**1616**	**589.00**	-0.80%
NEW-1000-0.5-50-10-5	**1597**	n/a	-	1694	1813.95	6.07%	**1597**	1825.22	0.00%	**1597**	**221.53**	0.00%
NEW-1000-0.8-10-50-1	4334	n/a	-	4678	1827.11	7.94%	4334	1806.33	0.00%	**4284**	**691.70**	-1.15%
NEW-1000-0.8-10-50-2	4352	n/a	-	4630	1807.23	6.39%	4352	1800.31	0.00%	**4242**	**578.95**	-2.53%
NEW-1000-0.8-10-50-3	4358	n/a	-	4698	1800.78	7.80%	4358	1841.15	0.00%	**4310**	**639.24**	-1.10%
NEW-1000-0.8-10-50-4	4375	n/a	-	4660	1817.18	6.51%	4375	1818.54	0.00%	**4295**	**422.35**	-1.83%
NEW-1000-0.8-10-50-5	4373	n/a	-	4587	1814.03	4.89%	4373	1857.44	0.00%	**4228**	**455.45**	-3.32%
NEW-1000-0.8-25-25-1	**2580**	n/a	-	2723	1802.05	5.54%	**2580**	1885.02	0.00%	**2580**	**417.76**	0.00%
NEW-1000-0.8-25-25-2	2605	n/a	-	2736	1811.44	5.03%	2605	1848.56	0.00%	**2557**	**505.91**	-1.84%
NEW-1000-0.8-25-25-3	2577	n/a	-	2742	1824.19	6.40%	2577	1818.58	0.00%	**2576**	**1293.46**	-0.04%
NEW-1000-0.8-25-25-4	2574	n/a	-	2668	1803.35	3.65%	2574	1815.11	0.00%	**2544**	**1119.54**	-1.17%
NEW-1000-0.8-25-25-5	2613	n/a	-	2726	1840.62	4.32%	2613	1800.48	0.00%	**2543**	**619.85**	-2.68%
NEW-1000-0.8-50-10-1	**1525**	n/a	-	1597	1820.47	4.72%	**1525**	1803.74	0.00%	**1525**	**485.52**	0.00%
NEW-1000-0.8-50-10-2	1549	n/a	-	1637	1819.92	5.68%	1549	1800.89	0.00%	**1542**	**825.45**	-0.45%
NEW-1000-0.8-50-10-3	**1526**	n/a	-	1594	1800.46	4.46%	**1526**	1815.45	0.00%	**1526**	**354.75**	0.00%
NEW-1000-0.8-50-10-4	1526	n/a	-	1579	1801.10	3.47%	1526	1800.99	0.00%	**1507**	**520.77**	-1.25%
NEW-1000-0.8-50-10-5	1541	n/a	-	1596	1806.17	3.57%	1541	1801.81	0.00%	**1532**	**765.95**	-0.58%
Average	1951.37	n/a	-	2000.84	1055.86	1.70%	1959.90	1026.59	0.20%	**1927.76**	**314.35**	-0.75%

Table B.4
Comparative results of KILS with re-MIP, re-GA and VNS on SNAP instances with a time limit of 18000 seconds.

Instance	*BKVs*	re-MIP [12]		re-GA [12]			VNS [12]			KILS		
		f_{best}	*time*	f_{best}	*time*	*pg%*	f_{best}	*time*	*pg%*	f_{best}	*time*	*pg%*
deezer HR	586365	n/a	-	586365	18000.17	0.00%	620957	18001.20	5.90%	**536771**	**17973.00**	-8.46%
deezer HU	555498	n/a	-	555498	18000.35	0.00%	556022	16186.68	0.09%	**508548**	**16021.60**	-8.45%
deezer RO	556582	654858	18001.23	556582	18000.09	0.00%	578040	11007.70	3.86%	**534449**	17909.90	-3.98%
Slashdot0811	6050730	n/a	-	6050730	18000.30	0.00%	6179223	18025.35	2.12%	**5936600**	**17980.40**	-1.89%
Slashdot0902	6015347	n/a	-	6015347	18000.10	0.00%	6247455	18000.99	3.86%	**5895170**	**17972.60**	-2.00%
Wiki-Vote	**650140***	**650140***	4.67	654677	18000.03	0.70%	651159	18012.22	0.16%	**650140***	**1140.14**	0.00%
facebook artist	740523	n/a	-	740523	18000.33	0.00%	912157	18019.65	23.18%	**698510**	**17842.30**	-5.67%
facebook athletes	169723	n/a	-	178107	18000.09	4.94%	169723	6178.07	0.00%	**166569**	17958.80	-1.86%
facebook company	199157	n/a	-	199157	18000.04	0.00%	203123	2560.45	1.99%	**192749**	16667.00	-3.22%
deezer europe	381800	381800	18000.02	391494	18000.06	2.54%	396971	6877.88	3.97%	**378612**	17989.60	-0.83%
facebook combined	27205	n/a	-	27537	13109.16	1.22%	27205	14645.15	0.00%	**27137**	16619.30	-0.25%
facebook government	71259	n/a	-	77368	18017.82	8.57%	71259	2545.88	0.00%	**69030**	17482.60	-3.13%
lastfm asia	**29814***	**29814***	3.38	30334	18010.74	1.74%	30169	3232.05	1.19%	**29814***	14739.30	0.00%
musae DE	114292	n/a	-	119985	18000.09	4.98%	114292	18001.97	0.00%	**110880**	**16281.40**	-2.99%
musae ENGB	**25540***	**25540***	3.14	26118	18013.43	2.26%	25859	4401.84	1.25%	**25540***	14333.45	0.00%
musae ES	106135	n/a	-	111628	18001.85	5.18%	106135	7198.66	0.00%	**104865**	**5172.06**	-1.20%
musae FR	72864	n/a	-	78113	18000.04	7.20%	72864	16243.31	0.00%	**71538**	17651.00	-1.82%
musae PTBR	**44605***	**44605***	16.19	45593	18001.42	2.21%	44886	2569.50	0.63%	**44605***	**195.84**	0.00%
musae RU	**14577***	**14577***	4.18	15083	18006.74	3.47%	14710	4839.94	0.91%	**14577***	16327.80	0.00%
musae facebook	285549	n/a	-	285549	18000.02	0.00%	286852	6384.36	0.46%	**271000**	17994.70	-5.10%
musae git	1425237	n/a	-	1425237	18000.10	0.00%	1508949	12529.38	5.87%	**1421929**	17769.10	-0.23%
facebook new sites	351544	n/a	-	351544	18000.12	0.00%	353735	8607.34	0.62%	**330643**	17860.20	-5.95%
facebook politician	66757	n/a	-	70258	18003.53	5.24%	66757	5039.59	0.00%	**66290**	5765.64	-0.70%
facebook public figure	181074	n/a	-	181074	18000.07	0.00%	181428	2458.28	0.20%	**174529**	14184.40	-3.61%
soc-Epinions1	6413824	n/a	-	6413824	18000.17	0.00%	6566233	18011.00	2.38%	**6365250**	**17952.80**	-0.76%
facebook tvshow	47173	n/a	-	48355	18005.77	2.51%	47173	1876.68	0.00%	**46990**	4969.87	-0.39%
twitter combined	721324	n/a	-	3876001	18000.89	437.35%	721324	18003.00	0.00%	**708565**	**17975.50**	-1.77%
Average	959431.04	n/a	-	1078225	17821.24	18.15%	990913.33	10350.30	2.17%	**940048.15**	14545.57	-2.38%

[1] Note: The results of re-MIP, re-GA and VNS are obtained by re-compiling these algorithms on our computer since these algorithms have not been applied on these instances.