TOP
https://doi.org/10.1007/s11750-025-00695-1

ORIGINAL PAPER

Heuristics for the weighted total domination problem

Alejandra Casado[1] · **Jesús Sánchez-Oro**[1] · **Anna Martínez-Gavara**[2]

Received: 7 July 2024 / Accepted: 13 January 2025

Abstract
The weighted total domination problem (WTDP) belongs to the family of dominating set problems. Given an edge- and vertex- weighted graph, the WTDP consists in selecting a total dominating set D, such that the sum of vertices and edges weights of the subgraph induced by D plus, for each vertex not in D, the minimum weight of its edge to a vertex in D is minimized. A total dominating set D is a subset of the graph's vertices, such that every vertex, including those in D, is at least adjacent to one vertex in D. This problem arises in many real-life applications closely related to covering and independent set problems; however, it remains computationally challenging due to its $\mathcal{NP}$-hardness. This work presents a variable neighborhood search (VNS) procedure to tackle the WTDP, and investigates the advantages and disadvantages of a multi-start strategy within VNS methodology. In addition, we develop a biased greedy randomized adaptive search procedure (Biased GRASP) that keeps adding elements once a feasible solution is found to produce high-quality initial solutions. We perform extensive numerical analysis to look into the influences of the algorithmic components and to disclose the contribution of the elements and strategies of our method. Finally, the empirical analysis shows that our proposal outperforms the state-of-art results, and the statistical analysis confirms the superiority of our proposal to find the best total dominating sets.

Keywords Weighted total domination problem · Graph domination · Biased grasp · Variable neighborhood search · Metaheuristics

✉ Anna Martínez-Gavara
gavara@uv.es
https://www.uv.es/gavara/

Alejandra Casado
alejandra.casado@urjc.es
https://alejandracasado.github.io/

Jesús Sánchez-Oro
jesus.sanchezoro@urjc.es
https://jesussanchezoro.github.io/

1 Department Computer Science and Statistics, Universidad Rey Juan Carlos, C/Tulipan S/N, 28933 Móstoles, Madrid, Spain

2 Departament d'Estadística i Investigació Operativa, Universitat de València, C/Doctor Moliner, 50, 46100 Burjassot, Valencia, Spain

Published online: 10 February 2025

Mathematics Subject Classification 68W99 · 90B99

1 Introduction

There are many optimization problems in scientific literature that consist in selecting a subgraph of a given input graph with a topological relation to the unselected vertices. Covering, independent, and dominating set problems are representative examples and constitute a relevant branch within graph theory (Beasley 1987; Beasley and Chu 1996; Haynes et al. 1998; Caprara et al. 2000; Tutte and Tutte 2001). In particular, domination in graphs is an important research area in graph optimization. It is difficult to say when the study of domination in graphs began, but we can say that the publications of Ore (1962), Berge (1962), and Cockayne and Hedetniemi (1977) are among the first references. The literature on domination graph theory is vast from both theoretical and practical perspectives, so we refer the reader to comprehensive surveys and books, such as Haynes et al. (1998), Henning (2009), Haynes et al. (2021), and Haynes et al. (2022b) to mention a few.

There are various types of dominating set problems based on the properties being considered. For instance, concerning the type of connectivity, we can find the independent dominating set, the connected dominating set, or the total dominating set (Guha and Khuller 1998; Goddard and Henning 2013; Cockayne et al. 1980). Neighboring variants emerge if each node needs to have a neighbor or k-neighbors in the dominating set (Hwang and Chang 1991; Corcoran and Gagarin 2021). Moreover, if each node has a limit on the number of neighbors that it has the capacity to dominate, the problem is then called capacitated dominating set (Li et al. 2018). Weighted variants arise when the aim is to minimize weight instead of cardinality (Balakrishnan and Ranganathan 2012). These are just some examples of the multiple variants that can be found in the literature (Haynes et al. 1998; Levin 2020; Haynes et al. 2020). The most relevant real-life applications such as facility location or social networks are shown in Table 1.

In this research paper, we study the edge- and vertex-weighted version of the total domination problem proposed by Ma et al. (2019), in which the objective is to find a total dominating set with a minimum total weight. This problem is called the Weighted Total Domination Problem (WTDP). This is an extension of the Total Domination Problem (TDP) introduced by Cockayne et al. (1980). The TDP is an $\mathcal{NP}$-hard (Laskar et al. 1984) optimization problem that has been widely studied in graph theory (Haynes et al. 1998, 2002a; Henning 2009). It is easy to see that the WTDP is $\mathcal{NP}$-hard, since the TDP is a particular case of WTDP with the edge and vertex weights equal to 0 and 1, respectively. Before defining the WTDP in mathematical terms, we introduce terminology and basic concepts of graph theory.

1.1 Graph theory terminology and basic definitions

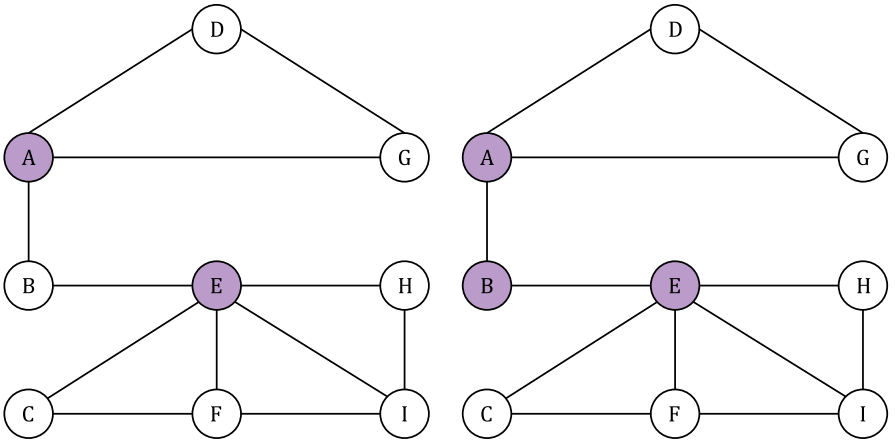
Let $G = (V, E)$ be an undirected graph, where V is the set of vertices and E is the set of edges. Following the same notation as in Haynes et al. (1998), we define the *neighborhood* of a vertex $v \in V$, $N(v)$, as the set of vertices adjacent to v, i.e.,

Table 1 Applications of dominating set and related problems

Context	References	Description
Chess problems	Berge (1962)	To find the minimum number of queens needed to cover or dominate every square on a chess board is one of the fundamental problems in graph theory
School bus routing	Sarubbi et al. (2016)	Location of the minimum number of bus stops for transporting children to and from school
Communication networks	Bai et al. (2020)	Construction of virtual backbone in wireless ad hoc networks
Radio stations	Haynes et al. (1998)	Location of the minimum number of radio stations, so that messages can be broadcast to all cities in the region
Electric power network	Haynes et al. (2002a)	Topology design for a power electricity networked system
Monitoring systems	Henning and Jafari Rad (2012)	Placement of monitoring devices such as fire alarms or surveillance cameras
Facility location	Corcoran and Gagarin (2021)	Location of one or more facilities in street networks
Social Networks	Campan et al. (2015); Wang et al. (2011)	Representation of the relationships between people to ensure a positive influence on the entire social network, or in order to communicate quickly within the network (e.g., in an emergency situation)

$N(v) = \{u \in V : e = (u, v) \in E\}$. We denote as $N[v] := N(v) \cup \{v\}$ the *closed neighborhood* of a vertex v. Note that each edge $e \in E$ connects two different vertices u and v from V, equally denoted by $e = (u, v)$ and by $e = (v, u)$. Given a subset of vertices S, we denote $N(S)$ as the set of neighboring vertices of the set S, i.e., $N(S) = \{v \in N(u),\ \forall u \in S\}$, and $N[S] = N(S) \cup S$. Furthermore, let $E(S)$ denote all the edges in E that have both endpoints in S, and let $G[S]$ be the subgraph induced in G by S. Throughout this paper, we will use vertex and node interchangeably.

Given an undirected graph G, a *dominating set* (DS) is a subset D of V, where each vertex in $V \backslash D$ is adjacent to some vertex from D. Furthermore, a *total dominating set* (TDS) of G with no isolated vertex is a subset $S \subset V$ of the vertices, such that every vertex is adjacent to a vertex in S, that is, $N(S) = V$. The existence of a total dominating set in a graph G requires that all vertices have at least one adjacent vertex, i.e., there are no isolated vertices. If there is no subset of S that can serve as a total dominating set of G, except for S itself, then S is considered to be a minimal total dominating set of G. A total dominating set S differs from a dominating set D in that its members must have adjacent vertices within S. On the other hand, in an ordinary dominating set D, its members may either be in the set or have adjacent vertices within



(a) Dominating set {A, E}. (b) Total dominating set {A, B, E}.

Fig. 1 Example of a dominating set and a total dominating set on a graph G

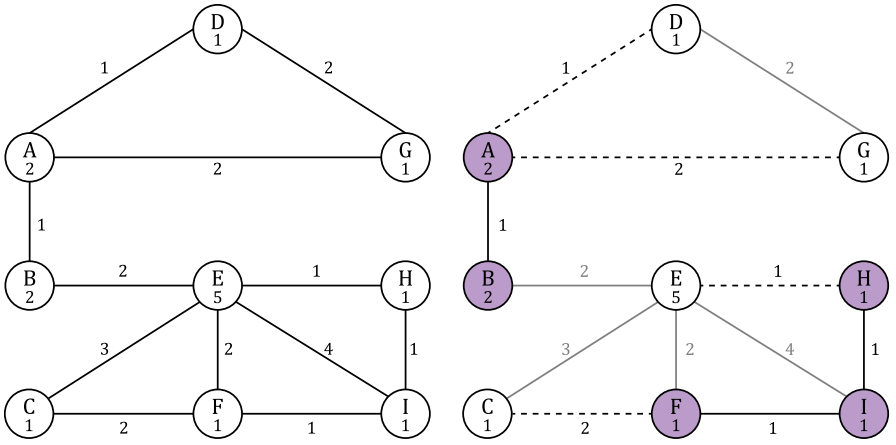
the set. The goal of the *total dominating set problem* is to find the total dominating set with minimum cardinality. Figure 1 shows an undirected graph $G = (V, E)$ with 9 vertices and 12 edges. The dominating set represented in Fig. 1a consists of the vertices with solid background A and E, whereas adding the vertex B a total dominating set is obtained; see Fig. 1b.

1.2 The weighted total domination problem

Consider an edge- and vertex-weighted graph $G = (V, \omega_V, E, \omega_E)$, where V and E are the set of vertices and edges, respectively. Let $\omega_V : V \to \mathbb{R}^+$ be the vertex weight function that maps all vertices onto the set of positive real numbers. Similarly, the edge weight function $\omega_E : E \to \mathbb{R}^+$ assigns a certain weight to each edge e of G. Specifically, $\omega_V(v)$ and $\omega_E(e)$ are the weights of vertex $v \in V$ and edge $e \in E$, respectively. For simplicity without loss of generality, we will use ω instead of ω_V and ω_E whenever there is no ambiguity between them. Then, the *weighted total domination problem* (WTDP) consists in finding a total dominating set S in G with minimum total weight, where the total weight is computed as the following cost function:

$$f(S) := \sum_{u \in S} \omega(u) + \sum_{e \in E(S)} \omega(e) + \sum_{v \in V \setminus S} min\{\omega(v, u) : u \in N(v) \cap S\}. \quad (1)$$

In Fig. 2a, we represent a double-weighted graph G by adding weights in the vertices and edges to the graph of Fig. 1. The label of vertices and their weights are included inside each node, and the edges weights are indicated besides the edges. The subset $S = \{A, B, F, H, I\}$ is the optimal solution of the WTDP of G. In Fig. 2b, vertices in the solution are filled with a solid background. The objective function value is obtained with the sum of the vertices' weights in S $(2 + 2 + 1 + 1 + 1)$, the sum of the edges of



(a) Original weighted graph. (b) Feasible solution.

Fig. 2 Example of a weighted total dominating set on a graph G

the subgraph induced in G by S marked with bold lines $(1 + 1 + 1)$, and the sum of the minimum edge weight between vertices not in S and vertices in S $(1 + 2 + 1 + 2)$, which are marked in bold dashed line. The total cost/weight is 16. Notice that those edges which are not involved in the objective function evaluation have been made less visible in the figure.

Most of the applications of the total domination problem or the weighted dominating set problem can be found in the context of the WTDP (Wang et al. 2011; Henning and Jafari Rad 2012; Zverovich 2021). For instance, the problem of placing a limited number of devices in a communication network, such that every site in the system (including the monitoring transmitters themselves) is adjacent to a monitor, can be modeled by a weighted total domination in graphs. The objective is to ensure that every site in the network can be directly linked to a site that has a transmitter. Furthermore, the placement of a device in each location and the rate of flow (transport service) incur in specific costs. Then, the minimum weight total dominating set problem aims to identify the smallest-weight dominating set in a double-weighted graph, without any restrictions on the size of the dominating set, i.e., number of devices. Note that communication networks usually involve high-operational costs and, therefore, the optimization problem considered here results significant cost reduction for the company providing this communication service.

Regarding the example depicted in Fig. 2, the graph models a network with six devices connected with 12 different links. To have a secure network, we need to locate some transmitters in the network in such a way that every device is directly connected to a transmitter. The weight of each node represents the cost of deploying a transmitter in that node and the weight of each link indicates the cost of transmitting information between the two endpoints of the link. Then, the solution depicted in Fig. 2b represents a network where the traffic flow is guaranteed minimizing the total cost of both deploying the transmitters and communicating them with the devices.

1.3 Literature review

As far as we know, there are only two research papers that deal with the WTDP. In 2019, Ma et al. (2019) proposed the WTDP, which is a combination of the TDP and the minimum weighted dominating set problem, and verify its computational complexity. Furthermore, the authors proposed three integer linear programming (ILP) formulations to solve it using an optimization solver like CPLEX. The benchmark set consists of 9 random graphs generated by the Erdös–Rényi model with different sizes, $|V| = 20, 50, 100$ (three instances per size). The three ILP models are able to find the optimal solution for instances with 20 and 50 vertices within a time limit of 1800 seconds; however, in the same time limit, none of the three models is able to solve the instances with 100 vertices.

In 2021, Álvarez-Miranda and Sinnl (2021) proposed two new Mixed-Integer Programming (MIP) models for the problem including valid inequalities. Moreover, the authors developed a greedy randomized adaptive search procedure (GRASP) and a genetic algorithm (GA) to solve up to 125 nodes. The authors generated 180 instances using the Erdös–Rényi model as in Ma et al. (2019) and considering the following number of vertices $|V| = 20, 50, 75, 100, 125$. An extensive computational experiment revealed that both MIP models are able to find almost all the optimal solutions of instances with $|V| = 100$ vertices within a time limit of 1800s, and some of the instances with $|V| = 125$. Finally, their GRASP and GA are able to find some of the optimal solutions within a short runtime.

These two previous research papers greatly contributed to the development of linear programming models that are able to compute optimal solutions in a reasonable time frame for small size instances. However, they require a significant amount of time in medium-size instances (up to 125) as it is expected due to the $\mathcal{NP}$-hardness of the WTDP. Moreover, the GRASP and GA methods proposed in the previous paper have difficulties in consistently producing the best solutions, and have not been tested in large-size instances. Therefore, there is a need for a strategy to overcome these limitations.

1.4 Contribution and outline

Literature review shows that WTDP is a computationally challenging problem that has been mainly studied from an exact perspective. The practical significance of this problem and its applicability to large networks make the use of heuristics specially suited for it. Our main contributions are summarized as follows:

(i) We propose a metaheuristic procedure based on VNS methodology to obtain high-quality solutions for large-size instances in a reasonable CPU time frame.
(ii) We study different strategies for the construction of high-quality initial solutions. We propose two heuristic algorithms with different greedy functions, and a biased-grasp construction (BGC) algorithm. In the BGC algorithm, the probability of selecting an element is biased toward those that not only have high quality but also contribute to the diversity of the solution. In addition, two versions are considered for each constructive method, the basic (that stops when a

feasible solution is found) and the extended design, which keeps adding elements to the solution.

(iii) We study efficient search strategies to reduce complexity and save computational time in the improvement phase of our proposal. We also study the comparison between a basic VNS design with a multi-start one.

(iv) We perform with statistical tests numerical experiments that find the best strategies and validate our proposals.

The following sections of this paper are organized as follows: Sect. 2 provides a detailed description of our metaheuristics for the WTDP. Section 3 presents computational experiments that first, Sect. 3.1 reveals the most effective strategies, and then, Sect. 3.2 shows the final results of our experimental study. The paper concludes in Sect. 4.

2 Algorithmic approach

As mentioned in Sect. 1.2, the WTDP is an $\mathcal{NP}$-hard problem, which makes exact algorithms not suitable when dealing with large and complex instances. In this research, a metaheuristic based on Variable Neighborhood Search (VNS) is proposed Brimberg et al. (2023). VNS requires starting the search from an initial solution, which can be generated either at random or with a more elaborate procedure. The latter is usually considered in the literature (Casado et al. 2023b) with the aim of providing VNS with a promising starting point, thus reducing the computational effort required to reach high-quality solutions. To that end, we propose three constructive procedures, where two of them are totally greedy methods, while the third method is based on the Greedy Randomized Adaptive Search Procedure (GRASP) framework. In particular, it considers the Biased GRASP version, which assigns a certain probability to each candidate element. The constructive methods will be thoroughly described in Sect. 2.1.

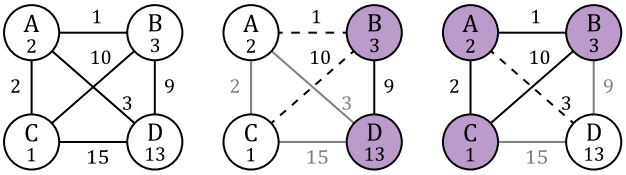
The constructed solutions can be locally improved, since they are not necessarily a local optimum with respect to any neighborhood. A local search method is described in Sect. 2.2 with the aim of reaching a local optimum with respect to the initially generated solution.

Finally, Sect. 2.3 presents the VNS algorithm which is designed for escaping from local optima by perturbing the solutions with a shake procedure. In this research, two different shake approaches are presented to evaluate the impact of diversification and intensification in the VNS framework.

2.1 Solution generation

This section is devoted to present three different constructive approaches to generating high-quality initial solutions for the WTDP. The first two approaches are purely greedy completely focused on quality, while the third approach adds diversity considering the inclusion of randomization to generate better solutions.

The three approaches follow the classical greedy scheme in which the procedure starts from an empty solution and iteratively adds elements until it becomes feasible.



(a) Original graph. (b) $S_1 = \{\texttt{B}, \texttt{D}\}$ (c) $S_2 = \{\texttt{A}, \texttt{B}, \texttt{C}\}$

Fig. 3 Example of a graph with four nodes and two possible solutions for the WTDP

However, regarding the WTDP constraints, it is important to remark that even when a solution is feasible, it is possible to add new elements and continue improving the incumbent solution. Analyzing Eq. (1) of the objective function, it can be divided into three different components: the sum of the weights of the selected nodes, the sum of the edge weights between every pair of selected nodes, and the sum of edge weights between a non-selected node and its adjacent selected node with the smallest edge weight. Including new nodes will increase the value of the two first components, but it may decrease the value of the third one. Therefore, if the decrease of the third component is, in absolute value, larger than the sum of the increase in the two first ones, the objective function will be improved. An example of this behavior can be seen in Fig. 3.

Figure 3a shows the input graph with 4 nodes. The capital letter and the number inside each node indicate its label and weight, respectively, and the number close to each edge denotes the weight of the edge. Solution $S_1 = \{\texttt{B}, \texttt{D}\}$, depicted in Fig. 3b, is conformed with two nodes, and the objective function value is evaluated as $f(S_1) = 16 + 9 + 11 = 36$. If we now analyze solution $S_2 = \{\texttt{A}, \texttt{B}, \texttt{C}\}$, presented in Fig. 3c, the result is $f(S_2) = 6 + 13 + 3 = 22$. Therefore, solution S_2 is better than S_1, even with an additional node in it.

Following this reasoning, a second variant of each constructive method is considered. In this variant, the method continues adding new elements to the solution until all the elements have been included in it. This strategy returns the best feasible solution found so far during the construction process. The main steps of the three constructive methods are described below.

Objective function greedy constructive

The first proposal is a greedy algorithm that selects a node at each iteration based on its objective function value. We refer to this approach as the Objective Function Greedy Constructive Method (OGC). The general scheme is presented in Algorithm 1.

The method starts by creating the solution, S, to which nodes will be iteratively added (step 1). Additionally, solution S_b is created (step 2). This solution is intended to maintain the best solution found, since S becomes feasible until the stopping criterion of the greedy procedure is met. Then, the set of unselected nodes U is initialized with all the nodes in the graph (step 3). The method iteratively selects a node to be included in the solution (steps 4–11) until reaching a stopping criterion.

Algorithm 1 `Greedy constructive`$(G = (V, E))$

1: $S \leftarrow \emptyset$
2: $S_b \leftarrow \emptyset$
3: $U \leftarrow V$
4: **while** $U \neq \emptyset$ **and** $f(S_b) \geq \sum_{e \in E(S)} \omega(e) + \sum_{s \in S} \omega(s)$ **do**
5: $\quad c \leftarrow \arg\min_{u \in U} g_{\texttt{OGC}}(u)$
6: $\quad S \leftarrow S \cup \{c\}$
7: $\quad U \leftarrow U \backslash \{c\}$
8: $\quad$ **if** $f(S) < f(S_b)$ **and** $N(S) = V$ **then**
9: $\quad\quad S_b \leftarrow S$
10: $\quad$ **end if**
11: **end while**
12: **return** S_b

In the context of WTDP, there are two main reasons to stop adding more nodes to a partial solution under construction (step 4). The first one is the simplest: there are no more candidate to add, i.e., all the nodes have already been included in the incumbent solution. The second one tries to reduce the computational effort by stopping the search when it is impossible to add a node without resulting in a solution of worse quality than the best solution found so far. In particular, if the sum of the weights of the selected nodes plus the sum of the internal edge weights is larger than or equal to the objective function value of the best solution found so far, then selecting any unselected node to be included in the incumbent solution will result in a solution with worse quality than S_b.

In each iteration, the node with the best value of a certain greedy criterion is selected (step 5). In this constructive method, the greedy criterion is directly the objective function value of the incumbent solution if the candidate node were included. More formally, $g_{\texttt{OGC}}(u) = f(S \cup \{u\})$. Once a node is selected, it is included in the solution under construction (step 6) and removed from the unselected nodes (step 7). At this point, it is necessary to check if solution S is feasible and better than the best solution found so far S_b (step 8). If so, the best solution found is updated (step 9).

Once the algorithm reaches the stopping criterion, it returns the best solution found during the search S_b (step 12).

Ratio-based greedy constructive

The second constructive procedure, named Ratio-based Greedy Constructive (`RGC`), follows the same scheme as `OGC` but vary the greedy function used in the selection of the next candidate. The main drawback of `OGC` is that it only evaluates the objective function value if the considered candidate is included in the solution. However, it does not necessarily reflect the actual contribution of the node to the objective function value. This is mainly because the external edges involved in the solution evaluation undergo through large variations when adding new nodes, so the selection of the next node in the next iteration will eventually result in a completely different selection of external edges, thus affecting to the objective function value. For instance, regarding solution S_1 depicted in Fig. 3b, the external edge (`B`, `C`) with weight $\omega(\texttt{B}, \texttt{C}) = 10$ is considered. However, if the node `A` were included in the solution, then node `C` would

be connected to the solution by the external edge (A, C), with weight $\omega(\mathtt{A}, \mathtt{C}) = 2$, thus replacing the external edge (B, C).

With the aim of overcoming this disadvantage, we propose to calculate a ratio between the node and edge weights for each candidate to analyze the relevance of including it in the solution. Notice that, as in OGC, the method will continue adding nodes until all the candidates are included in the solution. Therefore, there are two different situations to be considered during construction: unfeasible and feasible solution. The former refers to a solution which requires from additional nodes to be feasible, while the latter refers to a solution which is already feasible, but there are still more candidates that can be added. In short, given S the incumbent solution in one iteration of the construction algorithm, S is an unfeasible solution if $N(S) \neq V$, and it is feasible, otherwise. Then, the greedy function $g_{\mathrm{RGC}}(u)$ is defined differently depending on the situation.

Given a node $u \in U = V \backslash S$ for the first situation, i.e., S is an unfeasible solution, the ratio evaluates the sum of the weights of the nodes that will be dominated if u were included in the solution, divided by the weight of u plus the sum of the weights of the new internal edges if u is included in the solution. More formally

$$g^{u}_{\mathrm{RGC}}(u) = \frac{\sum_{v \in N(u) \cap N^c(S) \cap U} \omega(v)}{\omega(u) + \sum_{v \in N(u) \cap S} \omega(u, v)}, \tag{2}$$

where $N^c(S)$ represents the complement of the set of nodes adjacent to the solution S, i.e., $N^c(S) = V \backslash N(S)$. Therefore, $N(u) \cap N^c(S) \cap U$ is the set of unassigned nodes, adjacent to node u, and not dominated by the solution S. Notice that, in this ratio, the weights of the external edges are ignored, since their evaluation is computationally demanding and, additionally, the selection of future nodes will drastically modify the final external edges, as it was aforementioned.

The second situation occurs when the incumbent solution is already feasible. In that situation, if we use the same ratio, the numerator of the division is always equal to zero, since the inclusion of a node in a feasible solution will never dominate new nodes. Therefore, it is necessary to propose a modification of the ratio for feasible solutions. In this case, the modification maintains the weight of internal edges in the denominator. Then, the numerator is modified by including the sum over non-selected nodes adjacent to the candidate node, the weight of minimum edge weight to a selected node. In mathematical terms

$$g^{f}_{\mathrm{RGC}}(u) = \frac{\sum_{v \in N(u) \cap S^c} \omega(v, S)}{\omega(u) + \sum_{v \in N(u)} \omega(u, v)}, \tag{3}$$

where $\omega(v, S) := \min\{\omega(v, s) : s \in N(v) \cap S\}$.

In both cases, the larger the value, the better the candidate, so step 5 in Algorithm 1 is replaced by $c \leftarrow \arg\max_{u \in U} g_{\mathrm{RGC}}(u)$. Then, the greedy function $g_{\mathrm{RGC}}(u)$ takes the value of $g^{u}_{\mathrm{RGC}}(u)$ when the incumbent solution is not feasible, and $g^{f}_{\mathrm{RGC}}(u)$, otherwise.

Biased GRASP constructive

The two previous greedy algorithms lacks of diversity in the construction, since they select the next candidate following a totally greedy criterion. In this third constructive procedure, diversity becomes more relevant by considering randomization in the construction phase.

One of the most popular metaheuristics that combines randomization and greediness is the Greedy Randomized Adaptive Search (GRASP) methodology (Feo and Resende 1989). In the standard implementation of the GRASP construction phase, the next element to be introduced in the solution is chosen at random from the elements in the restricted candidate list. However, instead of the random function other probability distributions can be used to incorporate bias or preference information in the construction of the initial solutions (Resende and Ribeiro 2016). The bias information is used during the solution construction phase, with the aim to guide the construction process toward solutions that are likely to be of higher quality. As it is pointed out in Resende and Ribeiro (2016), these bias functions can be evaluated for all element in the candidate list (CL), or alternatively, their evaluation can be limited to the elements that belong to a restricted candidate list (RCL). There is a vast literature in heuristic methods that uses bias functions in the construction phase, so we refer interested readers to the following relevant research papers on this topic (Bresina 1996; Juan et al. 2013; Grasas et al. 2017; Buxey 1979; Faulin and Juan 2008). Specifically, we follow the biased randomization GRASP constructive framework formulated in Ferone et al. (2019), using an empirical non-uniform probability distribution instead of other well-known skewed (non-symmetric) distributions (e.g., geometric distribution).

Both the standard GRASP method and the bias variants are memoryless techniques. Generally, using bias functions, a weight is assigned to each candidate element based on a quality criterion (e.g., ranking, the objective function value) that does not incorporate information from previous iterations. There may be benefits to preserving such information to develop construction methods that make guided selection within the solution space. We propose adding a frequency memory function that modifies the evaluations of the greedy function to take into account previously recorded information (Resende and Ribeiro 2016; Fleurent and Glover 1999; Napoletano et al. 2019). To do so, each candidate vertex u has associated a ranking function value $g_{\text{BCG}}(u)$ computed as a combination of quality and diversity as follows:

$$g_{\text{BCG}}(u) = \alpha \cdot D + (1 - \alpha) \cdot Q, \tag{4}$$

where D and Q are metrics of diversity and quality, respectively, while $\alpha \in [0, 1]$ is an input parameter of the algorithm that balances both metrics. On the one hand, the quality metric used is calculated as the greedy function g_{RGC} considered in RGC. On the other hand, the diversity metric proposed is computed as the number of solutions in which the node has not been included. This diversity metric tries to prioritize the inclusion of nodes which have not appeared in a large number of solutions, which will eventually lead the method to explore a wider region of the search space. Then, the

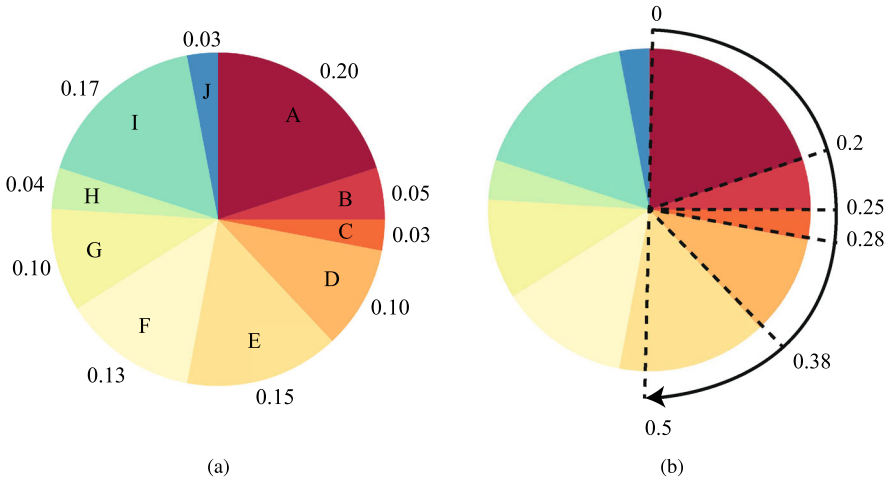


Fig. 4 Example of roulette selection considering 10 nodes

selection probability of a vertex u is set as

$$P(u) = \frac{g_{\text{BCG}}(u)}{\sum_{v \in CL} g_{\text{BCG}}(v)}. \tag{5}$$

The general framework of our biased GRASP constructive (BGC) method is similar to the RGC method but modifying the selection criterion. In particular, instead of selecting the most promising node according to the greedy function value, in BGC, the next candidate is selected from the *CL* according to the probability distribution. Our proposal differs from the classical framework in that it does not use a restrictive candidate list (RCL) to construct a feasible solution. Instead, our method selects the next vertex to be included in the current solution using a discrete non-uniform distribution based on the probabilities specified in Eq. (5) for all elements in the candidate list. Furthermore, as in the previous greedy constructive methods, BGC continues adding nodes until reaching the stopping criterion described in OGC.

From a computational perspective, the next candidate is selected using the roulette selection method, which effectively reduces computational time. In particular, a probability of being selected is assigned to each node, which is directly evaluated as the greedy function g_{BGC} normalized in the range [0, 1]. Then, a random number in the range [0, 1] is generated. The method then traverses the candidate list accumulating the probability of each node. The selected node will be the first node whose accumulated probability exceeds the random number generated. Notice that, although this selection includes a random selection, the most promising nodes have larger probabilities and, so, they are more likely to be selected than the least promising nodes. Let us illustrate this method with a graphical example depicted in Fig. 4, considering a social network with 10 nodes.

Figure 4a shows an example of the probability assigned to each node. As it can be seen in the figure, the nodes with larger probability are more likely to be selected than the ones with small probability. Then, a random number in the range [0, 1] is obtained,

being 0.51 in this particular example. Figure 4b then selects the first node whose accumulated probability is larger than or equal to 0.51. In particular, the selected node is E, since the accumulated probability is evaluated as $0.20+0.25+0.28+0.38+0.15 = 0.53$.

The inclusion of randomness in this procedure increases diversity, so it is interesting to generate more than a single solution to assure that a wide portion of the search space is explored. As it is customary in GRASP (Campos et al. 2014; Ferone et al. 2019), we perform 100 independent constructions.

2.2 Improvement method

The solution generated by any of the previously described constructive methods is not necessarily a local optimum. Therefore, it is interesting to apply a local improvement method to each generated solution.

The local search method proposed in this research is based on the exchange move operator, which consists of replacing one or more nodes from the solution with another nodes which are not included in it yet. In the particular case of the WTDP, the single exchange operator is considered, which removes a single node from the solution S, replacing it with another one not belonging to the solution ($U = V \setminus S$). More formally,

$$Exchange(S, s, u) \leftarrow (S \setminus \{s\}) \cup \{u\}$$

with $s \in S$ and $u \in U$.

Having defined the move operator, it is now required to indicate the neighborhood that the local improvement method will explore. In particular, the neighborhood is conformed with all the solutions that can be reached by performing a single exchange move. In mathematical terms,

$$N_e(S) = \{S' \leftarrow Exchange(S, s, u) \quad \forall s \in S \wedge \forall u \in U\}$$

.

Finally, it is necessary to define the strategy followed to traverse the proposed neighborhood. There are two main strategies to explore a neighborhood: best and first improvement. In both of them, the stopping criterion is usually the same: the method stops when no improvement is found.

In a best improvement strategy, the neighborhood is completely explored in each iteration, continuing the search with the best solution found in the neighborhood. This strategy is usually rather computationally demanding, since it requires to evaluate the complete neighborhood in each iteration of the local search to select the best neighbor solution.

On the contrary, first improvement strategy performs the first movement that leads to a better solution, even if that solution is not the best one of the neighborhood. This behavior allows the search to reduce the computational effort by avoiding the evaluation of the complete neighborhood.

Each strategy has different advantages and drawbacks. In the case of best improvement, although it always selects the best solution in the neighborhood, the movement

may quickly fall in a basin of attraction, which will eventually result in the stagnation in a local optimum. First improvement, in turn, does not select the best solution in the neighborhood but the first achieving and improve, thus eventually ignoring solutions that can be better than the selected one. However, if this selection is performed with certain randomness, the diversity of the search will increase, reducing the opportunities to fall in a basin of attraction. Additionally, each iteration of first improvement is usually faster than best improvement, since it does not require to evaluate the complete neighborhood.

The local search proposed for the WTDP follows the first improvement approach to reduce the computational effort, since its effectiveness compared with best improvement has been shown in several recent works (Hansen and Mladenović 2006; Casado et al. 2023a). In the context of first improvement, the order in which the neighborhood is explored may affect to the obtained solutions. The proposed local search follows a random exploration to favor diversity. Algorithm 2 presents the pseudocode of the local search method.

Algorithm 2 `Local search`(S)

```
1: improve ← TRUE
2: while improve do
3:    OF ← f(S)
4:    for s ∈ S do
5:       for u ∈ U do
6:          S ← Exchange(S, s, u)
7:          if f(S) < OF and N(S) = V then
8:             improve ← TRUE
9:             go to step 2
10:         end if
11:         S ← Exchange(S, u, s)
12:      end for
13:   end for
14:   improve ← FALSE
15: end while
```

The method receives as input parameter the solution S to be locally improve. Notice that this solution will be modified during the search and, therefore, the local search method does not return any value. The local search iterates, while an improvement is found (steps 2–15). In each iteration, the objective function value of the incumbent solution is stored (step 3). Then, the exploration of the neighborhood starts by randomly traversing the neighborhood (steps 4–13). For each pair of selected and unselected nodes s and u, respectively, the exchange move is performed (step 6). If the resulting solution is feasible and its objective function value is better than the previously stored value (step 7), then an improvement has been found (step 8), so the search is restarted (step 9). Otherwise, it is necessary to undo the exchange move (step 11). If all the neighborhood has been explored, then no improvement is found (step 14), indicating that the search must finish.

It is important to remark that this local search, hereinafter denoted as Exhaustive Local Search (`ELS`), requires to perform the movement, evaluate its quality and, in

case of a non-improvement move, undo the exchange movement, resulting in a very computationally demanding method. To reduce its complexity, we propose two local search variants which include different optimizations.

The first variant, named Predictive Local Search (PLS), tries to reduce the complexity by computing the objective function value obtained with the exchange move but without actually performing the exchange move. If the prediction indicates that the resulting solution would be feasible and better than the incumbent one, then the exchange move is effectively performed. The aim of this optimization is to reduce the number of exchange move performed, thus reducing the total computing time required by the local search method.

The second optimization, named Reduced Local Search (RLS), is designed to explore only those solutions that are feasible when performing an exchange move. In particular, when selecting a node s for the exchange move, not all the nodes $u \in U$ can be considered, since only the inclusion of some of them would result in a feasible solution. In particular, all the nodes adjacent to s which are dominated only by s must be connected to u to guarantee feasibility. Specifically, this optimization result in skipping the execution of steps 6–11 in those nodes that do not satisfy this constraint, thus avoiding the most computationally demanding part of the local search for a considerably large percentage of nodes. The exact reduction in computing time will be later discussed in Sect. 3. Notice that this optimization also includes the one presented in PLS; specifically, the objective function is evaluated without performing the exchange move.

2.3 Variable neighborhood search

Variable Neighborhood Search (VNS) is a metaheuristic framework which leverages systematic changes of neighborhood to escape from local optima. VNS was originally defined in Mladenović and Hansen (1997) and it is in constant evolution, leading to a wide variety of strategies such as Basic VNS, Reduced VNS, Variable Neighborhood Descent, or Variable Formulation Search, among others. We refer the reader to Hansen et al. (2019) for a recent survey on VNS methodology and applications.

The main differences among VNS variants rely on how the considered neighborhoods are explored: from a totally random exploration in the case of Reduced VNS to a completely deterministic search in Variable Neighborhood Descent. In this work, a Jump VNS (JVNS) is proposed, which combines the deterministic search of the local search method with the random exploration of the shake procedure, trying to balance intensification and diversification. Jump VNS (JVNS) was originally presented in Fleszar and Hindi (2004) as a variant of Basic VNS (BVNS). The main difference between JVNS and BVNS lies on the neighborhood change. Classical BVNS sequentially explores the neighborhoods from $k = 1$ to $k_{\max}$, increasing the neighbor in each step by one unit. However, in the context of WTDP, such a small neighborhood change will result in a rather similar solution, thus leading to the same local optimum. With the aim of increase efficiency by avoiding the exploration of the same solutions, JVNS modifies the size of the neighborhood change with an additional input parameter k_{step}.

Algorithm 3 shows the pseudocode of the proposed Jump VNS procedure (JVNS) for the WTDP.

Algorithm 3 JVNS(S, $k_{\max}$, k_{step}, Δ)

1: $S_b \leftarrow S$
2: **for** $i \in 1 \ldots \Delta$ **do**
3: $k \leftarrow k_{step}$
4: **while** $k \leq k_{\max}$ **do**
5: $S' \leftarrow Shake(S_b, k)$
6: $S'' \leftarrow LocalSearch(S')$
7: **if** $f(S'') < f(S_b)$ **then**
8: $S_b \leftarrow S''$
9: $k \leftarrow k_{step}$
10: **else**
11: $k \leftarrow k + k_{step}$
12: **end if**
13: **end while**
14: **end for**
15: **return** S_b

The method receives four input parameters: the initial solution, S; the maximum neighborhood to be explored, $k_{\max}$; the step performed when changing the neighborhood, k_{step}; and the number of VNS iterations executed, Δ. The values for all these parameters are determined in Sect. 3. The procedure starts initializing the best solution found with the initial solution S (step 1). Then, the algorithm performs a fixed number of complete JVNS iterations (step 2–14). Each iteration of JVNS starts initializing the neighborhood k to be explored. Then, the method starts the neighborhoods exploration until reaching the maximum considered neighborhood $k_{\max}$ (steps 4–13). The exploration of a given neighborhood starts with the *Shake* procedure, which is responsible of the diversification in the VNS methodology (step 5). In this research, two different shake procedures are proposed, which are detailedly described in Sect. 2.3. The method generates a solution S' in the current neighborhood k and, since S' is not necessarily a local optimum with respect to any neighborhood, the local search method described in Sect. 2.2 is applied to further improved S', resulting in S'' (step 6). Then, the neighborhood change method is applied, which decides the next neighborhood to be explored (steps 7–12). In particular, if the local optimum S'' outperforms the best solution found so far S_b (step 7), then it is updated (step 8), restarting the search from the first neighborhood (step 9). Otherwise, the method continues with the next considered neighborhood (step 11). Once all the iterations are performed, the method returns the best solution found during the search S_b (step 15).

Shake

The *Shake* procedure proposed in this work follows the classical approach in which random modifications are performed to the incumbent solution to generate a random neighbor solution. This modifications are usually performed by a move operator, which in this case is defined as removing k elements from the solution and then add new

elements until reaching the same stopping criterion as the one defined in Sect. 2.1, resulting in the best solution explored among the feasible ones. The elements removed are selected completely at random, and so the newly added nodes.

Multi-start JVNS

In VNS (and therefore, JVNS), the shake procedure introduces random changes to the current solution, allowing the algorithm to escape local optima and explore different regions of the solution space. The multi-start version of VNS extends this concept by incorporating multiple initial solutions. Instead of starting with a single solution, several diverse solutions are generated and treated as separate initial solutions, and the VNS procedure is run independently. The motivation behind multi-start VNS is to overcome the issue of being trapped in local optima by exploring various parts of the search space.

It is worth noting that a VNS algorithm with an improved perturbation phase can potentially achieve similar exploration capabilities as multi-start VNS. By designing a more effective perturbation strategy, the VNS algorithm can enhance its ability to diversify the search and discover high-quality solutions. For example, the recent study of Casado et al. (2023b) has proposed an intensified shake strategy with successful results.

Nonetheless, the choice between the standard VNS method and the multi-start approach remains a controversial subject. The effectiveness of each algorithm depends on various factors, including the specific problem, the instances being considered, and even the quality of the initial solutions generated. As a result, the selection between VNS and multi-start VNS continues to be debated upon within the optimization community. In the upcoming Sect. 3, we aim to shed light on this debate through experimental analysis.

3 Computational experiments

This section has two main objectives: select the best configuration of the proposed algorithm and then compare it with the best algorithm found in the literature for the WTDP. All the experiments have been performed in an AMD Ryzen 9 5950x (3.4 GHz) with 128GB RAM using Java 17 as programming language. The testbed of instances is divided into two different subsets. On the one hand, there are 180 instances directly derived from the state of the art for the WTDP, with the number of nodes ranging from 20 to 125. This set of instances is divided into two subsets in the literature, named as MA (45 instances) and NEW (135 instances). On the other hand, we propose a new set of 135 more challenging instances, named CSM, with size from 200 to 500 nodes to analyze the limits of the compared algorithms. Both the code of the proposed algorithms and the instances used are publicly available at https://grafo.etsii.urjc.es/wtdp.

The experiments are divided into two different stages: preliminary and final experimentation, Sects. 3.1 and 3.2, respectively. The former is intended to select the best configuration for the proposed algorithms, while the latter is designed to evaluate the

Table 2 Comparison of different values for α parameter in BGC

α	Avg.	Time (s)	Dev. (%)	#Best
0.0	612.84	0.13	0.36	22
0.1	665.28	0.13	7.70	2
0.2	681.24	0.13	10.65	2
0.3	686.68	0.13	12.23	0
0.4	707.16	0.15	15.76	0
0.5	710.08	0.15	16.41	0
0.6	730.76	0.15	19.16	0
0.7	735.04	0.16	19.58	0
0.8	730.60	0.16	20.06	0
0.9	725.24	0.16	19.08	0
RND	677.44	0.15	11.25	0

quality of the proposal when comparing it with the state-of-the-art algorithm for the WTDP. The following metrics are considered in all the experiments: Avg., the average objective function value; Time (s), the average computing time required by the algorithm measured in seconds; Dev. (%), the average deviation with respect to the best solution found in the experiment; and # Best, the number of times that the algorithm reaches the best solution found in the experiment.

3.1 Preliminary experimentation

This section presents the preliminary experiments performed to select the best configuration for the algorithms proposed in this research. With the aim of avoiding overfitting, these experiments have been performed over a subset of 25 representative instances extracted from the original set of 180 instances.

First of all, it is necessary to select the best value for the α parameter for the Biased GRASP constructive, named as BGC. The values tested are selected from $\alpha = 0.0$ to $\alpha = 0.9$, to evaluate the impact of diversification and intensification in the construction phase. Additionally, we have included a variant with $\alpha = RND$, which randomly selects the value for each construction. This variant has been successfully tested in traditional GRASP constructive procedures in some recent research Casado et al. (2022). Table 2 shows the results obtained in this experiment.

As it can be seen in the experiment, when considering the constructive procedure isolatedly, the quality of the solutions obtained is deteriorated with the increase in the α value, while maintaining similar computing times. In particular, the best solutions are found when considering $\alpha = 0.0$, with a deviation of 0.36% and 22 best solutions. It is worth mentioning that increasing the value to $\alpha = 0.1$ drastically worsens the quality of the solutions, obtaining an average deviation of 7.70% and only 2 best solutions. The results of this experiment suggest that the best option for biased GRASP constructive is considering that the probability of selecting a node depends on a totally greedy criterion. Therefore, for the remaining experiments, $\alpha = 0.0$ is considered.

Table 3 Comparison of the constructive procedure when considering the stopping criterion or the feasibility constraint

Constructive	Avg.	Time (s)	Dev. (%)	#Best
BGC (feasibility constraint)	640.80	0.06	5.09	8
BGC (stopping criterion)	612.84	0.13	1.43	19

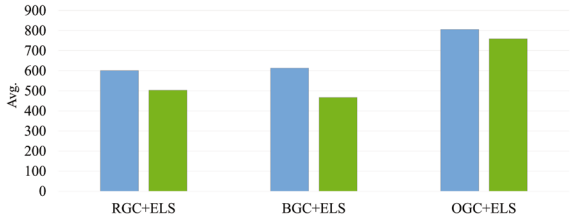


Fig. 5 Comparison of average objective function values (Avg.) obtained by constructive methods executed isolatedly (blue bars, on the left) and coupled with a local search method (green bars, on the right)

The next experiment is devoted to evaluate the impact of considering the stopping criterion described in Sect. 2.1 when comparing it with stopping the construction when reaching a feasible solution. To do so, the constructive selected in the previous experiment is evaluated using the stopping criterion and the feasibility criterion in Table 3.

As expected, the computing time is slightly larger when considering the stopping criterion, but it remains negligible. However, when analyzing the quality of the solutions, it can be seen that the constructive method that considers the stopping criterion is able to reach 19 out of 25 solutions, with a deviation of 1.43%. On the contrary, the one that stops when reaching a feasible solution achieves a deviation of 5.09% and only 8 best solutions, being considerably worse. Therefore, we select the constructive method based on the stopping criterion for the remaining experiments.

The next experiment is designed to analyze the performance of the constructive methods and then coupling them with a local search. Although the quality of one of the constructive methods may be better than the other ones when executing them isolatedly, the best constructive procedure may vary when considering the local search method. The rationale behind this is that those solutions which are initially worse but more diverse can eventually lead to better local optimum when applying a local optimizer. In this experiment, the Exhaustive Local Search (ELS) is considered as local optimizer for each constructive procedure. Notice that RGC and OGC are totally greedy procedures, so they only generate a single solution, while the randomization of BGC allows us to generate 100 independent solutions. Figure 5 shows the results obtained in this experiment. Blue bars (on the left of each algorithm) represent the results obtained by each constructive method and green bars (on the right) represent the results when considering the local search method.

Regarding the results provided by the constructive procedures, it can be seen that both RGC and BGC show similar performance, being RGC slightly better than BGC. Additionally, OGC seems to be the worst constructive method proposed, which highlights the relevance of providing an appropriate greedy function instead of directly evaluating the objective function value. If we now analyze the impact of the local

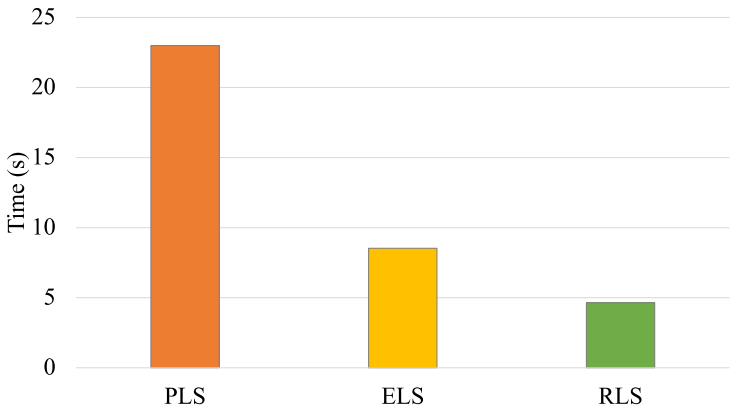


Fig. 6 Comparison of the performance of the three proposed local search methods using BGC as constructive procedure

search in each constructive procedure, it seems that it is not able to perform a relevant improvement in the OGC. However, with respect to RGC and BGC, the local search is able to further improve the obtained solutions. Finally, it is worth mentioning that BGC+ELS provides better quality solutions than RGC+ELS, emerging BGC as the best constructive procedure. The computing times are not included in the comparison, since the differences among algorithms are negligible.

In the last experiment, ELS was considered as the improvement procedure, but it is necessary to evaluate each one of the proposed local search methods: PLS, ELS, and RLS. This analysis is done by comparing only computing times, since the three proposed local search methods explore the same neighborhood in the same order, i.e., PLS and RLS are designed to increase the efficiency but not the efficacy of ELS. Figure 6 shows the average computing time required by the proposed local search when constructing and improving 100 solutions.

The most remarkable aspect of the figure is that PLS requires from more than twice the computing time required by ELS. This is mainly because the prediction performed by PLS to select the most promising moves is more time consuming than performing the move itself and then undo it if it is not an improvement. Therefore, PLS is discarded for the remaining experiments. If we now analyze RLS, it can be seem that it requires half of the computing time of ELS, resulting in an more efficient search. These results indicate that discarding those moves that lead to an unfeasible solution achieves a considerably improvement with respect to the required computing time. We then select RLS as the best local search method for the remaining experimentation.

The next experiment is devoted to select the maximum neighborhood to be explored in JVNS, i.e., the value of $k_{\max}$ parameter. The value of the jump step, k_{step}, is fixed to 0.1, which is 10% of the nodes already in the solution, and the values tested for $k_{\max}$ are $k_{\max} = \{0.2, 0.3, 0.4, 0.5\}$. Larger values are not tested, since it will result in a large modification of the incumbent solution, which may be equivalent to generate a completely new solution. Table 4 shows the results obtained in this experiment.

Table 4 Performance of JVNS when considering different values for the maximum neighborhood k_{max}

k_{max}	Avg.	Time (s)	Dev. (%)	#Best
0.2	464.56	19.59	0.22	22
0.3	464.04	30.66	0.16	23
0.4	463.88	32.13	0.02	24
0.5	463.88	46.48	0.02	24

Table 5 Performance of the multi-start variant of JVNS when considering different values of k_{max}

k_{max}	Avg.	Time (s)	Dev. (%)	#Best
0.2	464.08	34.19	0.31	19
0.3	463.04	47.91	0.11	21
0.4	463.04	57.39	0.11	21
0.5	463.20	73.21	0.15	20

Table 6 Comparison of JVNS with the multi-start approach

Algorithm	Avg.	Time (s)	Dev. (%)	#Best
Multi-start JVNS	463.04	47.91	0.12	21
Standard JVNS	463.88	37.32	0.18	23

Before analyzing the results, it is worth mentioning that the initial solution for JVNS is the best solution found among 100 solutions generated with Biased GRASP (BGC + RLS), and then 100 iterations of JVNS are performed, i.e., $\Delta = 0.1$. As expected, the computing time increases with the value of k_{max}. The quality of the solutions found stagnates when reaching $k_{max} = 0.4$, providing the same results with $k_{max} = 0.5$ but requiring more computing time. For that reason, we select $k_{max} = 0.4$ as the maximum neighborhood to be explored.

Having evaluated the results of JVNS, the next experiment evaluates a multi-start JVNS approach, where a single complete iteration of JVNS is performed for each solution generated with Biased GRASP. Table 5 shows the results obtained in this experiment.

In this experiment, the value of k_{max} is not as relevant as in the previous one, providing similar results for the different configurations in terms of quality. However, the computing time drastically increases with the k_{max} value and, therefore, we select $k_{max} = 0.3$, since it provides a good compromise between quality and computing time.

Having configured the best variant for JVNS and for the multi-start approach, the next experiment compares both algorithms. As in the previous experiments, the standard JVNS performs 100 iterations using as initial solution the best solution found in 100 iterations of Biased GRASP. On the contrary, the multi-start approach performs a single JVNS iteration for each one of the 100 solutions generated by Biased GRASP. Table 6 shows the results obtained in this experiment.

The results show that the multi-start approach is not significantly contributing to the quality of the final solution, providing similar results than the standard JVNS

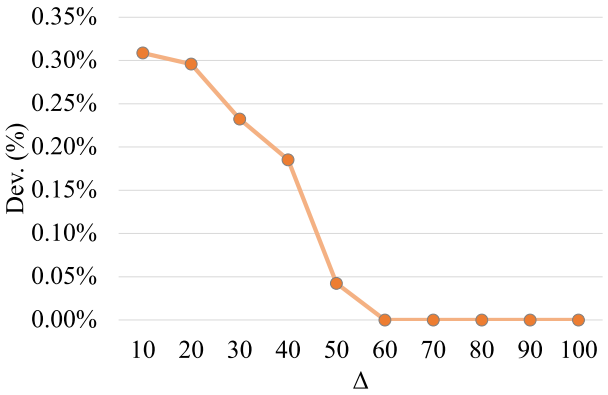


Fig. 7 Profile of the average deviation evolution in steps of 10 iterations for JVNS

Table 7 Comparison of constructive procedure, constructive procedure couple with local search and complete JVNS

Algorithm	Avg.	Time (s)	Dev. (%)	#Best
BGC	612.84	0.13	28.74	0
Biased GRASP	467.00	4.64	0.62	12
JVNS	463.88	24.43	0.00	25

approach. However, it requires considerably more computing time, so we have decided to maintain the standard JVNS for the remaining experiments.

It is important to analyze the evolution of JVNS with the increase in the number of iterations, since it may stagnate in a certain value, performing iterations in which the algorithm will not be able to outperform the quality of the incumbent solution. To select the best number of JVNS iterations, i.e., the value of parameter Δ, Fig. 7 shows a profile of the evolution of the average deviation obtained in steps of 10 iterations.

As it can be seen in the figure, the algorithm finds new improvements until reaching 60 iterations, where it is not able to find further improvements. This indicates that the last 40 iterations are only consuming computing time and, therefore, the maximum number of iterations can be reduced from 100 to 60. Then, for the remaining experiments, the value of Δ is set to 60.

Once the final version of JVNS has been configured, it is important to evaluate the contribution of each part to the complete algorithm. Table 7 shows the comparison among the constructive procedure BGC, the constructive procedure coupled with the local search Biased GRASP and, finally, the complete JVNS algorithm.

Regarding the computing time, the addition of new components to the algorithm also affects to the computational time, being JVNS the slowest algorithm and BGC the fastest one. A similar but inverse analysis can be done when considering the quality. The initial solutions provided by BGC remain at a considerably average deviation with respect to the best ones found in the experiment, but the inclusion of the local search drastically reduces the average deviation. However, JVNS is required to reach the best solutions of the experiment. Notice that, although Biased GRASP has a small deviation (0.62%), the number of best solutions found is smaller than half of the

Table 8 Comparison of the initial solutions of JVNS and GA1, generated with Biased GRASP and GRASP, respectively

Algorithm	Avg.	Time (s)	Dev. (%)	#Best
Biased GRASP	523.20	2.89	0.07	123
GRASP	539.96	2.95	2.69	55

best solutions found by JVNS. These results highlight that JVNS is a key part of the proposal.

3.2 Final experimentation

The final set of experiments is devoted to performing a thorough comparison of the proposed JVNS with the best algorithms found in the state of the art. In particular, the best method found in the literature for the WTDP is a Genetic Algorithm (GA1) where the initial population is generated with a traditional GRASP algorithm and each individual is improved with a local search method Álvarez-Miranda and Sinnl (2021). In that research, authors also propose an exact approach, but they conclude that GA1 is the best algorithm for the WTDP. Notice that, in this section, the complete set of instances is used in each experiment. The hardware used in that research is equivalent or even better than the one considered in this work, resulting in a fair comparison. We refer the reader to the appendix to see the individual results of each algorithm over every single instance (Tables 12, 13, 14, 15).

The first experiment is designed to compare the quality of the initial solution in both JVNS and GA1 approaches. As it was aforementioned, the previous work considers a GRASP algorithm, while the initial solution for JVNS is generated by the proposed Biased GRASP method. The results of the previous algorithm have been directly transcribed from the original manuscript. Table 8 shows the results obtained in this experiment.

Analyzing the computing time, both algorithms required similar computational time. In terms of quality, Biased GRASP shows its superiority by reaching 123 out of 135 instances, with an average deviation of 0.07%. This deviation indicates that, in those instances in which biased GRASP is not able to reach the best value, it still remains close to it. It is worth mentioning that the GRASP proposed in the state of the art also shows a good performance in terms of deviation, but it is not able to reach more than 55 best solutions. We have performed a non-parametric pairwise Wilcoxon statistical test to evaluate if there are statistically significant differences between both results, and the obtained p value smaller than 0.01 confirms this hypothesis.

The best work in the literature proposes four different exact methods for solving the WTDP, evaluating their performance over the set of MA instances, which are less challenging than the NEW set for an exact approach. We have tested JVNS over this set of instances to evaluate how far from the optimal value JVNS is when it is known. Table 9 shows the results obtained in this experiment.

As it can be derived from the results, the best exact procedure is f2+, which is able to reach all the optimal values in reasonable computing times when compared with the other exact approaches. If we analyze the results of JVNS, we can clearly see that the

Table 9 Comparison of JVNS with the exact methods over the set of MA instances

Algorithm	Avg.	Time (s)	Dev. (%)	#Best
JVNS	95.35	3.48	0.03	44
f1	95.51	449.08	0.11	42
f1+	95.33	157.28	0.00	45
f2	95.33	151.22	0.00	45
f2+	95.33	51.04	0.00	45

Table 10 Comparison of JVNS and GA1 over the NEW testbed proposed in the original work

Number of nodes	Algorithm	Avg.	Time (s)	Dev. (%)	#Best
75	JVNS	421.78	4.43	0.02	43
	GA1	423.87	9.91	0.34	39
100	JVNS	525.53	11.14	0.05	42
	GA1	526.51	23.13	0.23	36
125	JVNS	611.31	22.83	0.12	39
	GA1	611.80	47.76	0.20	37

computing time is an order of magnitude smaller than the best exact method, and two orders of magnitude smaller than the other exact approaches. Indeed, in only 3 s on average, JVNS is able to reach 44 out of 45 instances with a deviation of 0.03%. This value indicates that in the only instance in which it is not able to reach the optimal value, JVNS finds a high-quality local optimum. In this case, the Wilcoxon statistical test results in a p value of 0.317, indicating that there are not statistically significant differences between both results, highlighting the performance of JVNS.

The next experiment is devoted to compare JVNS with the best previous heuristic approach, which is the Genetic Algorithm (GA1) proposed in Álvarez-Miranda and Sinnl (2021). In this case, the set of more challenging NEW instances is considered, which was originally proposed by the previous authors with the aim of highlighting the necessity of a non-exact approach for the WTDP. Table 10 depicts the comparison between these two algorithms.

The results are divided by the number of nodes of the considered instances to better analyze the performance of the algorithms. Regarding the computing times, JVNS is consistently requiring half of the computing time required by GA1. In terms of quality, JVNS reaches almost all the best solutions in each subset, showing the smallest deviation with respect to the best solution found, while GA1 also shows a good performance. The Wilcoxon statistical test results in a p value smaller than 0.01, confirming the superiority of JVNS.

Since both MA and NEW sets seem to be easily solved by both JVNS and GA1, mainly due to the small size of the considered instances for a heuristic approach, we propose a new set of larger and more challenging instances with nodes ranging from 200 to 500, named CSM. With the aim of having a fair comparison, we contacted the authors for their original source code, but, unfortunately, it was not available.

Table 11 Comparison of JVNS and GA1 in the set of more challenging instances CSM

Number of nodes	Algorithm	Avg.	Time (s)	Dev. (%)	#Best
200	JVNS	871.40	100.73	0.00	45
	GA1	937.08	962.77	6.72	1
350	JVNS	1332.04	616.34	0.00	45
	GA1	1794.06	1824.58	29.79	0
500	JVNS	1776.40	1164.68	0.00	45
	GA1	2539.57	1900.48	37.56	0

For that reason, we carefully reimplemented the GA1 algorithm following the detailed description of the original work. Table 11 shows the results obtained in this comparison.

As it can be seen in the results, JVNS performs better when the instances become more complex, showing its scalability when comparing it with GA1. It is worth mentioning the increase in the computing for both approaches, which highlights the difficulty of solving this new set of instances. In spite of this increase, JVNS is still requiring, approximately, half of the computing time than GA1. In terms of quality, it seems that GA1 reaches its limits when solving instances with 200 nodes, since for larger instances, the average deviation drastically increases and it is not able to reach any best solution. On the contrary, JVNS emerges as a competitive algorithm for the WTDP, finding all the best solutions for this new set of challenging instances. Finally, the p value smaller than 0.01 obtained in the Wilcoxon statistical test indicates that there are statistically significant differences between JVNS and GA1.

4 Conclusions and future research

In this research paper, we study the $\mathcal{NP}$-hard Weighted Total Dominating Set Problem (WTDP), which extends the Total Dominating Set Problem (TDP) by incorporating weights to the vertices and edges of the graph. This practical problem has applications in areas, such as facility location, social networks, and communications networks where the number of required open facilities, users, or devices is often large. To test our proposed methodology in realistic scenarios, we generated large-size instances.

The merit of our study goes beyond the mere application of a methodology to a problem, as we delve into deeper analysis and investigation of the methodologies themselves. As such, the main contributions of this research paper are mainly methodological. We proposed a metaheuristic procedure based on the Jump Variable Neighborhood Search (JVNS) methodology to obtain high-quality solutions for large-size instances within a reasonable CPU time frame. We also studied efficient search strategies to compare basic JVNS design with a multi-start one.

Additionally, we analyzed different strategies for the construction of high-quality initial solutions, proposing two heuristic algorithms with different greedy functions and a biased GRASP construction algorithm. In the latter, two stopping criteria, the basic design and the extended, were empirically compared to investigate the impact of

both criteria on the constructed solution. Furthermore, three different variants of the local search procedures based on exchange moves have been studied to obtain a desired trade-off between solution quality and computation time. Finally, we performed statistical tests and numerical experiments to validate our proposals, which showed that our proposed methods and strategies outperformed existing state-of-the-art approaches.

To sum up, our research makes significant contributions to the field of metaheuristic optimization, particularly in the context of the WTDP. Our findings provide a valuable basis for developing further improvements in this area, with potential applications in communication and social networks, among others.

As a future work, we propose the investigation of matheuristic approaches to tackle the WTDP. Matheuristics combine mathematical programming techniques with heuristic methods, such as metaheuristics, to improve the efficiency and effectiveness of the optimization process. We believe that the combination of mathematical models and heuristic search can lead to better solutions for large instances of the WTDP in a reasonable computational time. This would allow us to find high-quality solutions that are guaranteed to be near-optimal.

Moreover, in this research paper, we have focused on finding the dominating set with the smallest weight. As mentioned in the Introduction, this problem arises in communication networks, where reducing the number of transmitters can also lead to decreased radiation emission. For future research, we plan to transform the problem into a multi-criteria minimization problem where the weight assigned to each vertex in the graph represents a cost or nuisance measurement parameter, like the noise level at the transmitter site. Then, the aim is to design an efficient metaheuristic to identify the smallest-size dominating set in a weighted graph that incurs the lowest cost.

Appendix

See Tables 12, 13, 14, and 15.

Table 12 Individual results obtained by the initial solutions of JVNS and GA1 (generated with biased GRASP and GRASP, respectively) for each instance in the NEW testbed

Instance	Biased GRASP		GRASP	
	Avg.	Time(s)	Avg.	Time(s)
NEW-75−0.2-10-50-1	690	1.88	769	1.00
NEW-75−0.2-10-50-2	784	2.02	871	1.00
NEW-75−0.2-10-50-3	662	1.93	765	1.00
NEW-75−0.2-10-50-4	746	1.79	762	1.00
NEW-75−0.2-10-50-5	766	1.90	857	1.00
NEW-75−0.2-25-25-1	502	1.38	556	1.00
NEW-75−0.2-25-25-2	546	1.55	607	1.00
NEW-75−0.2-25-25-3	518	1.44	603	1.00
NEW-75−0.2-25-25-4	503	1.47	521	1.00
NEW-75−0.2-25-25-5	513	1.45	526	1.00
NEW-75−0.2-50-10-1	340	1.30	340	1.00
NEW-75−0.2-50-10-2	382	1.22	414	1.00
NEW-75−0.2-50-10-3	335	1.18	341	1.00
NEW-75−0.2-50-10-4	333	1.22	338	1.00
NEW-75−0.2-50-10-5	348	1.17	353	1.00
NEW-75−0.5-10-50-1	581	1.52	590	1.00
NEW-75−0.5-10-50-2	602	1.43	641	1.00
NEW-75−0.5-10-50-3	554	1.34	545	1.00
NEW-75−0.5-10-50-4	540	1.30	580	1.00
NEW-75−0.5-10-50-5	530	1.38	551	1.00
NEW-75−0.5-25-25-1	387	1.04	402	1.00
NEW-75−0.5-25-25-2	384	1.23	413	1.00
NEW-75−0.5-25-25-3	362	1.09	380	1.00
NEW-75−0.5-25-25-4	366	1.08	371	1.00
NEW-75−0.5-25-25-5	331	1.19	331	1.00
NEW-75−0.5-50-10-1	240	0.69	244	1.00
NEW-75−0.5-50-10-2	238	0.63	245	1.00
NEW-75−0.5-50-10-3	215	0.73	215	1.00
NEW-75−0.5-50-10-4	235	0.68	235	1.00
NEW-75−0.5-50-10-5	206	0.57	206	1.00
NEW-75−0.8-10-50-1	571	0.90	613	2.00
NEW-75−0.8-10-50-2	520	0.82	520	2.00
NEW-75−0.8-10-50-3	543	0.86	543	2.00
NEW-75−0.8-10-50-4	571	0.90	571	2.00
NEW-75−0.8-10-50-5	509	0.86	509	2.00
NEW-75−0.8-25-25-1	357	0.77	360	2.00
NEW-75−0.8-25-25-2	338	0.73	356	2.00
NEW-75−0.8-25-25-3	323	0.75	323	2.00

Table 12 continued

Instance	Biased GRASP		GRASP	
	Avg.	Time(s)	Avg.	Time(s)
NEW-75−0.8-25-25-4	345	0.82	345	2.00
NEW-75−0.8-25-25-5	311	0.77	311	2.00
NEW-75−0.8-50-10-1	182	0.43	182	2.00
NEW-75−0.8-50-10-2	188	0.38	188	2.00
NEW-75−0.8-50-10-3	191	0.38	191	2.00
NEW-75−0.8-50-10-4	196	0.46	196	2.00
NEW-75−0.8-50-10-5	192	0.42	192	2.00
NEW-100−0.2-10-50-1	897	5.10	930	1.00
NEW-100−0.2-10-50-2	961	4.81	983	1.00
NEW-100−0.2-10-50-3	895	5.23	905	1.00
NEW-100−0.2-10-50-4	846	4.85	879	1.00
NEW-100−0.2-10-50-5	840	5.35	907	1.00
NEW-100−0.2-25-25-1	596	3.04	591	1.00
NEW-100−0.2-25-25-2	657	3.47	687	1.00
NEW-100−0.2-25-25-3	615	3.37	648	1.00
NEW-100−0.2-25-25-4	558	3.63	602	1.00
NEW-100−0.2-25-25-5	619	3.15	646	1.00
NEW-100−0.2-50-10-1	418	2.30	422	1.00
NEW-100−0.2-50-10-2	447	2.43	472	1.00
NEW-100−0.2-50-10-3	420	3.34	427	1.00
NEW-100−0.2-50-10-4	403	2.70	418	1.00
NEW-100−0.2-50-10-5	378	2.42	379	1.00
NEW-100−0.5-10-50-1	758	3.23	749	2.00
NEW-100−0.5-10-50-2	700	3.30	705	3.00
NEW-100−0.5-10-50-3	724	3.02	730	3.00
NEW-100−0.5-10-50-4	754	3.14	775	2.00
NEW-100−0.5-10-50-5	726	3.28	743	2.00
NEW-100−0.5-25-25-1	461	2.83	461	3.00
NEW-100−0.5-25-25-2	437	2.92	448	2.00
NEW-100−0.5-25-25-3	434	2.35	443	3.00
NEW-100−0.5-25-25-4	489	2.33	489	2.00
NEW-100−0.5-25-25-5	462	2.71	470	3.00
NEW-100−0.5-50-10-1	260	1.45	260	2.00
NEW-100−0.5-50-10-2	271	1.26	271	2.00
NEW-100−0.5-50-10-3	283	1.33	283	3.00
NEW-100−0.5-50-10-4	291	1.73	296	2.00
NEW-100−0.5-50-10-5	269	1.33	269	2.00
NEW-100−0.8-10-50-1	734	1.88	730	4.00

Table 12 continued

Instance	Biased GRASP		GRASP	
	Avg.	Time(s)	Avg.	Time(s)
NEW-100−0.8-10-50-2	688	1.75	688	4.00
NEW-100−0.8-10-50-3	718	1.94	718	4.00
NEW-100−0.8-10-50-4	712	2.09	709	4.00
NEW-100−0.8-10-50-5	703	1.85	710	4.00
NEW-100−0.8-25-25-1	442	1.64	452	5.00
NEW-100−0.8-25-25-2	430	1.69	430	4.00
NEW-100−0.8-25-25-3	426	1.79	426	4.00
NEW-100−0.8-25-25-4	428	1.63	428	4.00
NEW-100−0.8-25-25-5	432	1.64	432	4.00
NEW-100−0.8-50-10-1	259	0.83	259	4.00
NEW-100−0.8-50-10-2	246	0.95	246	4.00
NEW-100−0.8-50-10-3	238	0.90	238	4.00
NEW-100−0.8-50-10-4	253	0.96	258	4.00
NEW-100−0.8-50-10-5	248	0.85	250	5.00
NEW-125−0.2-10-50-1	1031	10.39	1112	2.00
NEW-125−0.2-10-50-2	1070	10.20	1069	2.00
NEW-125−0.2-10-50-3	958	10.72	1124	2.00
NEW-125−0.2-10-50-4	1072	9.74	1121	2.00
NEW-125−0.2-10-50-5	979	10.93	1112	2.00
NEW-125−0.2-25-25-1	735	7.24	803	2.00
NEW-125−0.2-25-25-2	751	8.19	768	2.00
NEW-125−0.2-25-25-3	733	7.22	752	2.00
NEW-125−0.2-25-25-4	707	7.25	726	2.00
NEW-125−0.2-25-25-5	692	7.57	747	2.00
NEW-125−0.2-50-10-1	455	4.80	457	2.00
NEW-125−0.2-50-10-2	481	5.38	493	2.00
NEW-125−0.2-50-10-3	490	5.12	501	2.00
NEW-125−0.2-50-10-4	468	5.13	504	2.00
NEW-125−0.2-50-10-5	457	4.84	468	2.00
NEW-125−0.5-10-50-1	817	6.58	817	4.00
NEW-125−0.5-10-50-2	821	6.11	827	5.00
NEW-125−0.5-10-50-3	878	5.64	880	4.00
NEW-125−0.5-10-50-4	867	6.31	914	4.00
NEW-125−0.5-10-50-5	871	6.27	906	5.00
NEW-125−0.5-25-25-1	573	5.06	566	5.00
NEW-125−0.5-25-25-2	533	5.36	561	5.00

Table 12 continued

Instance	Biased GRASP		GRASP	
	Avg.	Time(s)	Avg.	Time(s)
NEW-125−0.5-25-25-3	538	4.81	567	5.00
NEW-125−0.5-25-25-4	557	4.65	565	4.00
NEW-125−0.5-25-25-5	548	4.96	548	5.00
NEW-125−0.5-50-10-1	336	2.68	336	4.00
NEW-125−0.5-50-10-2	330	2.59	330	4.00
NEW-125−0.5-50-10-3	315	2.74	315	5.00
NEW-125−0.5-50-10-4	316	2.41	316	5.00
NEW-125−0.5-50-10-5	313	2.59	311	4.00
NEW-125−0.8-10-50-1	793	3.85	793	9.00
NEW-125−0.8-10-50-2	860	3.51	854	8.00
NEW-125−0.8-10-50-3	787	3.81	829	9.00
NEW-125−0.8-10-50-4	777	3.96	829	9.00
NEW-125−0.8-10-50-5	836	3.56	827	8.00
NEW-125−0.8-25-25-1	508	3.44	521	9.00
NEW-125−0.8-25-25-2	498	3.27	499	9.00
NEW-125−0.8-25-25-3	523	3.04	523	9.00
NEW-125−0.8-25-25-4	495	3.45	506	8.00
NEW-125−0.8-25-25-5	512	2.81	519	8.00
NEW-125−0.8-50-10-1	309	1.67	307	8.00
NEW-125−0.8-50-10-2	296	2.05	296	8.00
NEW-125−0.8-50-10-3	297	1.80	294	8.00
NEW-125−0.8-50-10-4	270	2.01	270	9.00
NEW-125−0.8-50-10-5	278	1.87	278	9.00

Table 13 Individual results obtained by the exacts methods f1, f1+, f2, f2+ and the proposed algorithm JVNS for each instance in the MA testbed

Instance	JVNS		f1		f1+		f2		f2+	
	OF	Time(s)	OF	Time(s)	OF	Time(s)	OF	Time(s)	OF	Time(s)
MA-20−0.2-5-5-1	63	0.06	63	1.00	63	1.00	63	1.00	63	1.00
MA-20−0.2-5-5-2	58	0.04	58	1.00	58	1.00	58	1.00	58	1.00
MA-20−0.2-5-5-3	58	0.05	58	3.00	58	1.00	58	1.00	58	1.00
MA-20−0.2-5-5-4	51	0.06	51	1.00	51	1.00	51	1.00	51	1.00
MA-20−0.2-5-5-5	55	0.04	55	1.00	55	1.00	55	1.00	55	1.00
MA-20−0.5-5-5-1	44	0.04	44	1.00	44	1.00	44	1.00	44	1.00
MA-20−0.5-5-5-2	47	0.04	47	1.00	47	1.00	47	1.00	47	1.00
MA-20−0.5-5-5-3	46	0.04	46	1.00	46	1.00	46	1.00	46	1.00
MA-20−0.5-5-5-4	40	0.04	40	1.00	40	1.00	40	1.00	40	1.00
MA-20−0.5-5-5-5	41	0.04	41	1.00	41	1.00	41	1.00	41	1.00
MA-20−0.8-5-5-1	37	0.04	37	1.00	37	1.00	37	1.00	37	1.00
MA-20−0.8-5-5-2	35	0.04	35	3.00	35	1.00	35	1.00	35	1.00
MA-20−0.8-5-5-3	40	0.04	40	1.00	40	1.00	40	1.00	40	1.00
MA-20−0.8-5-5-4	34	0.03	34	1.00	34	1.00	34	1.00	34	1.00
MA-20−0.8-5-5-5	34	0.04	34	1.00	34	1.00	34	1.00	34	1.00
MA-50−0.2-5-5-1	111	1.31	111	2.00	111	1.00	111	4.00	111	1.00
MA-50−0.2-5-5-2	106	1.32	106	3.00	106	1.00	106	3.00	106	1.00
MA-50−0.2-5-5-3	111	1.35	111	8.00	111	1.00	111	4.00	111	1.00
MA-50−0.2-5-5-4	101	1.34	101	4.00	101	1.00	101	4.00	101	1.00
MA-50−0.2-5-5-5	108	1.37	108	13.00	108	1.00	108	6.00	108	1.00
MA-50−0.5-5-5-1	82	0.84	82	4.00	82	3.00	82	3.00	82	2.00

Table 13 continued

Instance	JVNS		f1		f1+		f2		f2+	
	OF	Time(s)	OF	Time(s)	OF	Time(s)	OF	Time(s)	OF	Time(s)
MA-50−0.5-5-5-2	85	0.73	85	5.00	85	2.00	85	3.00	85	2.00
MA-50−0.5-5-5-3	85	0.75	84	22.00	84	3.00	84	4.00	84	2.00
MA-50−0.5-5-5-4	82	1.06	82	16.00	82	3.00	82	4.00	82	2.00
MA-50−0.5-5-5-5	82	0.72	82	15.00	82	4.00	82	4.00	82	2.00
MA-50−0.8-5-5-1	77	0.38	77	6.00	77	7.00	77	5.00	77	4.00
MA-50−0.8-5-5-2	72	0.44	72	3.00	72	3.00	72	3.00	72	2.00
MA-50−0.8-5-5-3	74	0.39	74	3.00	74	3.00	74	4.00	74	2.00
MA-50−0.8-5-5-4	76	0.39	76	6.00	76	5.00	76	4.00	76	3.00
MA-50−0.8-5-5-5	79	0.36	79	13.00	79	15.00	79	7.00	79	7.00
MA-100−0.2-5-5-1	175	18.70	175	898.00	175	92.00	175	566.00	175	50.00
MA-100−0.2-5-5-2	174	19.04	174	251.00	174	14.00	174	276.00	174	12.00
MA-100−0.2-5-5-3	177	19.36	178	1800.00	177	239.00	177	1434.00	177	121.00
MA-100−0.2-5-5-4	169	17.81	169	1800.00	169	81.00	169	562.00	169	37.00
MA-100−0.2-5-5-5	167	17.32	171	1800.00	167	97.00	167	1473.00	167	47.00
MA-100−0.5-5-5-1	147	7.21	147	1800.00	147	304.00	147	292.00	147	108.00
MA-100−0.5-5-5-2	144	6.56	144	707.00	144	158.00	144	152.00	144	51.00
MA-100−0.5-5-5-3	147	6.64	147	995.00	147	401.00	147	186.00	147	128.00
MA-100−0.5-5-5-4	146	7.33	149	1800.00	146	725.00	146	289.00	146	214.00
MA-100−0.5-5-5-5	139	7.42	139	1800.00	139	466.00	139	242.00	139	155.00
MA-100−0.8-5-5-1	136	3.35	136	1655.00	136	346.00	136	172.00	136	97.00
MA-100−0.8-5-5-2	140	2.94	140	759.00	140	894.00	140	283.00	140	249.00
MA-100−0.8-5-5-3	141	3.31	141	1212.00	141	1032.00	141	236.00	141	325.00
MA-100−0.8-5-5-4	141	3.22	141	1800.00	141	1652.00	141	334.00	141	495.00
MA-100−0.8-5-5-5	134	3.27	134	990.00	134	509.00	134	231.00	134	160.00

Table 14 Individual results obtained by JVNS and GA1 for each instance in the NEW testbed

Instance	JVNS		GA1	
	OF	Time(s)	OF	Time(s)
NEW-75−0.2-10-50-1	686	9.08	686	5.00
NEW-75−0.2-10-50-2	770	10.36	794	6.00
NEW-75−0.2-10-50-3	661	9.81	661	6.00
NEW-75−0.2-10-50-4	705	9.35	740	7.00
NEW-75−0.2-10-50-5	758	9.64	779	6.00
NEW-75−0.2-25-25-1	498	6.73	504	6.00
NEW-75−0.2-25-25-2	546	5.48	546	6.00
NEW-75−0.2-25-25-3	518	6.02	518	5.00
NEW-75−0.2-25-25-4	500	8.85	498	6.00
NEW-75−0.2-25-25-5	513	9.12	513	6.00
NEW-75−0.2-50-10-1	339	6.74	339	6.00
NEW-75−0.2-50-10-2	382	7.21	382	5.00
NEW-75−0.2-50-10-3	335	6.23	341	5.00
NEW-75−0.2-50-10-4	333	7.37	333	6.00
NEW-75−0.2-50-10-5	348	6.40	347	6.00
NEW-75−0.5-10-50-1	581	4.25	581	13.00
NEW-75−0.5-10-50-2	602	4.17	602	11.00
NEW-75−0.5-10-50-3	545	4.36	545	10.00
NEW-75−0.5-10-50-4	540	4.53	540	10.00
NEW-75−0.5-10-50-5	519	4.63	519	10.00
NEW-75−0.5-25-25-1	387	4.04	387	10.00
NEW-75−0.5-25-25-2	384	3.81	384	10.00
NEW-75−0.5-25-25-3	362	4.02	362	10.00
NEW-75−0.5-25-25-4	366	3.60	371	9.00
NEW-75−0.5-25-25-5	331	3.84	331	10.00
NEW-75−0.5-50-10-1	240	2.00	240	9.00
NEW-75−0.5-50-10-2	238	2.40	238	9.00
NEW-75−0.5-50-10-3	215	2.68	215	9.00
NEW-75−0.5-50-10-4	235	3.36	235	9.00
NEW-75−0.5-50-10-5	206	3.18	206	8.00
NEW-75−0.8-10-50-1	571	2.19	571	16.00
NEW-75−0.8-10-50-2	520	2.12	520	15.00
NEW-75−0.8-10-50-3	543	2.39	543	15.00
NEW-75−0.8-10-50-4	571	2.28	571	15.00
NEW-75−0.8-10-50-5	509	2.12	509	17.00
NEW-75−0.8-25-25-1	357	1.87	357	15.00
NEW-75−0.8-25-25-2	338	1.80	338	15.00
NEW-75−0.8-25-25-3	323	2.07	323	13.00

Table 14 continued

Instance	JVNS		GA1	
	OF	Time(s)	OF	Time(s)
NEW-75−0.8-25-25-4	345	2.16	345	13.00
NEW-75−0.8-25-25-5	311	2.00	311	15.00
NEW-75−0.8-50-10-1	182	1.10	182	14.00
NEW-75−0.8-50-10-2	188	0.94	188	11.00
NEW-75−0.8-50-10-3	191	0.96	191	11.00
NEW-75−0.8-50-10-4	196	1.04	196	12.00
NEW-75−0.8-50-10-5	192	0.98	192	15.00
NEW-100−0.2-10-50-1	873	20.45	873	12.00
NEW-100−0.2-10-50-2	944	25.17	944	13.00
NEW-100−0.2-10-50-3	878	22.57	878	11.00
NEW-100−0.2-10-50-4	837	22.34	837	11.00
NEW-100−0.2-10-50-5	840	24.68	870	12.00
NEW-100−0.2-25-25-1	591	24.36	591	12.00
NEW-100−0.2-25-25-2	653	22.61	655	11.00
NEW-100−0.2-25-25-3	615	22.84	616	12.00
NEW-100−0.2-25-25-4	552	22.47	552	11.00
NEW-100−0.2-25-25-5	613	22.97	607	12.00
NEW-100−0.2-50-10-1	418	16.64	420	12.00
NEW-100−0.2-50-10-2	447	16.74	456	11.00
NEW-100−0.2-50-10-3	420	17.85	419	11.00
NEW-100−0.2-50-10-4	403	14.97	410	12.00
NEW-100−0.2-50-10-5	375	18.17	379	13.00
NEW-100−0.5-10-50-1	758	8.41	749	26.00
NEW-100−0.5-10-50-2	700	10.29	700	25.00
NEW-100−0.5-10-50-3	718	8.19	718	24.00
NEW-100−0.5-10-50-4	726	8.82	726	26.00
NEW-100−0.5-10-50-5	702	8.65	702	25.00
NEW-100−0.5-25-25-1	461	9.02	461	25.00
NEW-100−0.5-25-25-2	437	9.21	437	19.00
NEW-100−0.5-25-25-3	434	9.52	434	22.00
NEW-100−0.5-25-25-4	482	9.19	482	25.00
NEW-100−0.5-25-25-5	456	9.12	457	23.00
NEW-100−0.5-50-10-1	260	7.57	260	22.00
NEW-100−0.5-50-10-2	271	4.81	271	21.00
NEW-100−0.5-50-10-3	283	7.88	283	21.00
NEW-100−0.5-50-10-4	291	6.00	291	22.00

Table 14 continued

Instance	JVNS		GA1	
	OF	Time(s)	OF	Time(s)
NEW-100−0.5-50-10-5	269	7.48	269	21.00
NEW-100−0.8-10-50-1	730	6.03	730	39.00
NEW-100−0.8-10-50-2	683	5.76	683	37.00
NEW-100−0.8-10-50-3	718	4.62	718	37.00
NEW-100−0.8-10-50-4	709	5.84	709	41.00
NEW-100−0.8-10-50-5	700	5.18	704	39.00
NEW-100−0.8-25-25-1	442	4.16	442	40.00
NEW-100−0.8-25-25-2	430	4.04	430	32.00
NEW-100−0.8-25-25-3	426	4.11	426	36.00
NEW-100−0.8-25-25-4	428	4.05	428	35.00
NEW-100−0.8-25-25-5	432	4.16	432	42.00
NEW-100−0.8-50-10-1	259	2.72	259	32.00
NEW-100−0.8-50-10-2	246	2.87	246	9.00
NEW-100−0.8-50-10-3	238	2.72	238	34.00
NEW-100−0.8-50-10-4	253	3.09	253	34.00
NEW-100−0.8-50-10-5	248	2.76	248	31.00
NEW-125−0.2-10-50-1	1031	43.17	1026	24.00
NEW-125−0.2-10-50-2	1046	58.66	1038	22.00
NEW-125−0.2-10-50-3	935	59.19	947	23.00
NEW-125−0.2-10-50-4	1068	41.43	1051	21.00
NEW-125−0.2-10-50-5	974	62.42	975	25.00
NEW-125−0.2-25-25-1	720	47.17	720	26.00
NEW-125−0.2-25-25-2	751	46.71	748	24.00
NEW-125−0.2-25-25-3	715	38.54	717	21.00
NEW-125−0.2-25-25-4	701	51.23	705	22.00
NEW-125−0.2-25-25-5	685	41.35	697	23.00
NEW-125−0.2-50-10-1	455	28.84	455	21.00
NEW-125−0.2-50-10-2	477	34.22	477	23.00
NEW-125−0.2-50-10-3	490	48.71	490	21.00
NEW-125−0.2-50-10-4	467	30.64	467	23.00
NEW-125−0.2-50-10-5	457	42.30	459	24.00
NEW-125−0.5-10-50-1	817	18.41	817	41.00
NEW-125−0.5-10-50-2	815	18.83	815	45.00
NEW-125−0.5-10-50-3	836	18.40	872	45.00
NEW-125−0.5-10-50-4	867	15.69	867	55.00
NEW-125−0.5-10-50-5	867	17.30	867	55.00
NEW-125−0.5-25-25-1	566	17.01	566	48.00

Table 14 continued

Instance	JVNS		GA1	
	OF	Time(s)	OF	Time(s)
NEW-125−0.5-25-25-2	533	15.43	533	48.00
NEW-125−0.5-25-25-3	538	14.25	538	49.00
NEW-125−0.5-25-25-4	552	15.44	552	53.00
NEW-125−0.5-25-25-5	548	14.35	548	48.00
NEW-125−0.5-50-10-1	334	12.43	334	40.00
NEW-125−0.5-50-10-2	330	12.02	330	38.00
NEW-125−0.5-50-10-3	315	11.15	315	49.00
NEW-125−0.5-50-10-4	316	13.56	316	51.00
NEW-125−0.5-50-10-5	311	13.86	311	40.00
NEW-125−0.8-10-50-1	793	10.99	793	78.00
NEW-125−0.8-10-50-2	845	10.82	845	72.00
NEW-125−0.8-10-50-3	787	11.36	787	74.00
NEW-125−0.8-10-50-4	777	11.80	777	83.00
NEW-125−0.8-10-50-5	827	10.61	813	77.00
NEW-125−0.8-25-25-1	508	7.91	510	69.00
NEW-125−0.8-25-25-2	498	7.75	498	65.00
NEW-125−0.8-25-25-3	513	7.95	513	77.00
NEW-125−0.8-25-25-4	495	8.13	493	75.00
NEW-125−0.8-25-25-5	504	9.07	504	76.00
NEW-125−0.8-50-10-1	307	5.30	307	64.00
NEW-125−0.8-50-10-2	296	6.18	296	57.00
NEW-125−0.8-50-10-3	294	5.72	294	71.00
NEW-125−0.8-50-10-4	270	5.56	270	86.00
NEW-125−0.8-50-10-5	278	5.58	278	77.00

Table 15 Individual results obtained by JVNS and GA1 for each instance in the CSM testbed

Instance	JVNS		GA1	
	OF	Time(s)	OF	Time(s)
CSM_200_0.2_10_50_0	600	144.80	654	770.15
CSM_200_0.2_10_50_1	602	161.99	692	744.12
CSM_200_0.2_10_50_2	600	182.17	675	692.04
CSM_200_0.2_10_50_3	616	131.27	684	766.95
CSM_200_0.2_10_50_4	569	139.64	658	778.11
CSM_200_0.2_25_25_0	981	217.21	1161	970.28
CSM_200_0.2_25_25_1	976	240.23	1110	1005.88
CSM_200_0.2_25_25_2	960	209.94	1107	1070.22
CSM_200_0.2_25_25_3	961	212.59	1044	1088.88
CSM_200_0.2_25_25_4	986	223.07	1154	947.46
CSM_200_0.2_50_10_0	1448	222.21	1692	1369.55
CSM_200_0.2_50_10_1	1469	261.30	1766	1219.78
CSM_200_0.2_50_10_2	1474	251.46	1681	1204.24
CSM_200_0.2_50_10_3	1388	256.15	1610	1292.35
CSM_200_0.2_50_10_4	1395	250.85	1650	1098.83
CSM_200_0.5_10_50_0	452	43.83	468	901.42
CSM_200_0.5_10_50_1	459	42.29	475	925.97
CSM_200_0.5_10_50_2	468	42.78	469	877.50
CSM_200_0.5_10_50_3	485	47.17	495	876.27
CSM_200_0.5_10_50_4	473	39.85	503	855.06
CSM_200_0.5_25_25_0	777	57.11	809	977.23
CSM_200_0.5_25_25_1	789	56.17	813	941.31
CSM_200_0.5_25_25_2	762	57.52	782	1021.13
CSM_200_0.5_25_25_3	795	60.33	808	1008.91
CSM_200_0.5_25_25_4	810	62.45	814	1045.16
CSM_200_0.5_50_10_0	1260	73.20	1278	1046.46
CSM_200_0.5_50_10_1	1219	70.46	1294	1108.63
CSM_200_0.5_50_10_2	1212	95.07	1231	1058.00
CSM_200_0.5_50_10_3	1192	71.99	1267	1005.29
CSM_200_0.5_50_10_4	1199	71.46	1274	1007.62
CSM_200_0.8_10_50_0	415	28.51	419	892.54
CSM_200_0.8_10_50_1	438	26.81	451	867.71
CSM_200_0.8_10_50_2	421	26.61	429	891.12
CSM_200_0.8_10_50_3	437	28.02	450	906.00
CSM_200_0.8_10_50_4	413	29.21	425	883.29
CSM_200_0.8_25_25_0	734	36.57	748	929.82
CSM_200_0.8_25_25_1	714	40.10	744	913.50

Table 15 continued

Instance	JVNS		GA1	
	OF	Time(s)	OF	Time(s)
CSM_200_0.8_25_25_2	717	36.53	729	902.35
CSM_200_0.8_25_25_3	738	37.17	773	916.45
CSM_200_0.8_25_25_4	761	40.67	786	920.69
CSM_200_0.8_50_10_0	1208	40.26	1229	927.94
CSM_200_0.8_50_10_1	1230	39.92	1246	929.90
CSM_200_0.8_50_10_2	1216	40.61	1226	923.72
CSM_200_0.8_50_10_3	1197	43.95	1197	922.46
CSM_200_0.8_50_10_4	1197	41.54	1199	922.77
CSM_350_0.2_10_50_0	927	1014.89	1073	1803.19
CSM_350_0.2_10_50_1	862	858.24	1058	1804.19
CSM_350_0.2_10_50_2	895	1090.68	1070	1818.24
CSM_350_0.2_10_50_3	896	960.16	1051	1804.46
CSM_350_0.2_10_50_4	887	1023.91	1068	1817.50
CSM_350_0.2_25_25_0	1440	1269.64	1934	1805.28
CSM_350_0.2_25_25_1	1459	1262.89	2010	1806.73
CSM_350_0.2_25_25_2	1412	1288.04	1910	1808.76
CSM_350_0.2_25_25_3	1394	1322.82	1956	1804.63
CSM_350_0.2_25_25_4	1438	1424.21	1980	1813.41
CSM_350_0.2_50_10_0	2279	1741.24	3433	1819.43
CSM_350_0.2_50_10_1	2156	1554.72	3494	1812.11
CSM_350_0.2_50_10_2	2215	1483.53	3320	1820.55
CSM_350_0.2_50_10_3	2214	1530.55	3556	1803.45
CSM_350_0.2_50_10_4	2157	1551.85	3471	1805.70
CSM_350_0.5_10_50_0	700	244.40	860	1839.04
CSM_350_0.5_10_50_1	713	228.98	858	1827.00
CSM_350_0.5_10_50_2	676	250.13	869	1848.07
CSM_350_0.5_10_50_3	696	230.46	850	1831.88
CSM_350_0.5_10_50_4	725	201.95	878	1818.23
CSM_350_0.5_25_25_0	1171	457.92	1807	1842.03
CSM_350_0.5_25_25_1	1227	452.56	1837	1827.81
CSM_350_0.5_25_25_2	1179	433.61	1611	1841.12
CSM_350_0.5_25_25_3	1177	416.77	1749	1833.80
CSM_350_0.5_25_25_4	1193	431.12	1697	1838.18
CSM_350_0.5_50_10_0	1900	477.80	3212	1837.24
CSM_350_0.5_50_10_1	1933	549.72	3066	1840.80
CSM_350_0.5_50_10_2	1959	490.02	3328	1837.95
CSM_350_0.5_50_10_3	1963	537.02	2900	1831.29

Table 15 continued

Instance	JVNS		GA1	
	OF	Time(s)	OF	Time(s)
CSM_350_0.5_50_10_4	1959	464.76	2970	1807.96
CSM_350_0.8_10_50_0	650	131.58	682	1843.37
CSM_350_0.8_10_50_1	632	135.53	679	1846.13
CSM_350_0.8_10_50_2	641	141.96	684	1812.22
CSM_350_0.8_10_50_3	631	139.10	688	1826.25
CSM_350_0.8_10_50_4	644	135.21	669	1841.98
CSM_350_0.8_25_25_0	1133	157.19	1214	1833.01
CSM_350_0.8_25_25_1	1111	174.04	1208	1847.53
CSM_350_0.8_25_25_2	1091	186.70	1247	1850.23
CSM_350_0.8_25_25_3	1125	169.01	1230	1807.24
CSM_350_0.8_25_25_4	1092	166.26	1182	1867.19
CSM_350_0.8_50_10_0	1858	196.68	2110	1826.85
CSM_350_0.8_50_10_1	1883	192.70	2011	1820.90
CSM_350_0.8_50_10_2	1876	181.90	2074	1810.52
CSM_350_0.8_50_10_3	1889	177.40	2093	1808.97
CSM_350_0.8_50_10_4	1884	205.55	2086	1814.04
CSM_500_0.2_10_50_0	1135	1830.06	1392	1819.39
CSM_500_0.2_10_50_1	1141	1813.15	1376	1802.85
CSM_500_0.2_10_50_2	1170	1806.12	1452	1832.91
CSM_500_0.2_10_50_3	1136	1819.35	1483	1812.93
CSM_500_0.2_10_50_4	1139	1803.09	1449	1831.61
CSM_500_0.2_25_25_0	1932	1819.89	2843	1893.62
CSM_500_0.2_25_25_1	1854	1835.34	2785	1891.55
CSM_500_0.2_25_25_2	1886	1829.45	2786	1808.43
CSM_500_0.2_25_25_3	1879	1853.40	2516	1803.62
CSM_500_0.2_25_25_4	1946	1831.72	2761	1818.01
CSM_500_0.2_50_10_0	3108	1848.74	4769	1893.00
CSM_500_0.2_50_10_1	2967	1835.35	5013	1800.93
CSM_500_0.2_50_10_2	3011	1811.88	5227	1824.67
CSM_500_0.2_50_10_3	2989	1800.92	4905	1894.16
CSM_500_0.2_50_10_4	2931	1814.70	5037	1826.73
CSM_500_0.5_10_50_0	957	851.29	1322	1866.73
CSM_500_0.5_10_50_1	902	1020.22	1236	1854.97
CSM_500_0.5_10_50_2	927	900.02	1274	1859.99
CSM_500_0.5_10_50_3	938	889.09	1227	1871.50
CSM_500_0.5_10_50_4	949	688.51	1309	1854.16

Table 15 continued

Instance	JVNS		GA1	
	OF	Time(s)	OF	Time(s)
CSM_500_0.5_25_25_0	1586	1357.61	2360	1853.90
CSM_500_0.5_25_25_1	1565	1328.90	2720	1852.22
CSM_500_0.5_25_25_2	1550	1310.22	2465	1859.24
CSM_500_0.5_25_25_3	1590	1234.01	2769	1864.02
CSM_500_0.5_25_25_4	1569	1192.46	2600	1872.73
CSM_500_0.5_50_10_0	2587	1404.38	4557	1829.10
CSM_500_0.5_50_10_1	2659	1338.04	4580	1831.23
CSM_500_0.5_50_10_2	2634	1410.40	4357	1823.80
CSM_500_0.5_50_10_3	2553	1446.41	4235	1843.98
CSM_500_0.5_50_10_4	2631	1427.80	4609	1834.04
CSM_500_0.8_10_50_0	864	394.51	949	2002.55
CSM_500_0.8_10_50_1	876	373.37	917	2023.97
CSM_500_0.8_10_50_2	891	357.42	963	2013.56
CSM_500_0.8_10_50_3	855	363.48	914	1998.57
CSM_500_0.8_10_50_4	860	358.12	919	2039.70
CSM_500_0.8_25_25_0	1475	562.28	1665	1978.82
CSM_500_0.8_25_25_1	1487	503.48	1563	2042.10
CSM_500_0.8_25_25_2	1464	514.17	1681	2035.73
CSM_500_0.8_25_25_3	1478	510.39	1606	2033.05
CSM_500_0.8_25_25_4	1486	522.88	1604	2007.98
CSM_500_0.8_50_10_0	2493	557.17	2868	2025.01
CSM_500_0.8_50_10_1	2458	582.73	2840	2003.86
CSM_500_0.8_50_10_2	2442	543.08	2830	1994.57
CSM_500_0.8_50_10_3	2514	555.86	2842	1998.85
CSM_500_0.8_50_10_4	2474	559.22	2706	1997.36

Acknowledgements This research has been partially supported by the Ministerio de Ciencia e Innovación of Spain (Grant Ref. PID2021–125709OB-C21 and PID2021–125709OA-C22) funded by MCIN/AEI /10.13039/501100011033 / FEDER, UE. It has been also supported by the Generalitat Valenciana (CIAICO/2021/224).

Author Contributions Alejandra Casado: algorithm design, algorithm implementation, and writing. Jesús Sánchez-Oro: methodology, algorithm design, and writing—original draft preparation. Anna Martínez-Gavara: methodology, algorithm design, and writing.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature.

Data availability All the data, including instances, results, and source code, are publicly available at https://grafo.etsii.urjc.es/wtdp.

References

Álvarez-Miranda E, Sinnl M (2021) Exact and heuristic algorithms for the weighted total domination problem. Comput Oper Res 127:105157

Bai X, Zhao D, Bai S, Wang Q, Li W, Mu D (2020) Minimum connected dominating sets in heterogeneous 3d wireless ad hoc networks. Ad Hoc Netw 97:102023

Balakrishnan R, Ranganathan K (2012) A textbook of graph theory. Springer, Berlin

Beasley JE (1987) An algorithm for set covering problem. Eur J Oper Res 31:85–93

Beasley JE, Chu PC (1996) A genetic algorithm for the set covering problem. Eur J Oper Res 94:392–404

Berge C (1962) The theory of graphs and its applications. Methuen & co. Ltd, London

Bresina JL (1996) Heuristic-biased stochastic sampling. AAAI/IAAI 1:271–278

Brimberg J, Salhi S, Todosijević R, Urošević D (2023) Variable neighborhood search: the power of change and simplicity. Comput Oper Res 155:106221

Buxey G (1979) The vehicle scheduling problem and monte carlo simulation. J Oper Res Soc 30:563–573

Campos V, Martí R, Sánchez-Oro J, Duarte A (2014) Grasp with path relinking for the orienteering problem. J Oper Res Soc 65:1800–1813

Campan A, Truta TM, Beckerich M (2015) Fast dominating set algorithms for social networks. In: Midwest Artificial Intelligence and Cognitive Science Conference, pp. 55–62

Caprara A, Toth P, Fischetti M (2000) Algorithms for the set covering problem. Ann Oper Res 98:353–371

Casado A, Pérez-Peló S, Sánchez-Oro J, Duarte A (2022) A grasp algorithm with tabu search improvement for solving the maximum intersection of k-subsets problem. J Heuris 28:121–146

Casado A, Bermudo S, López-Sánchez A, Sánchez-Oro J (2023) An iterated greedy algorithm for finding the minimum dominating set in graphs. Math Comput Simul 207:41–58

Casado A, Mladenović N, Sánchez-Oro J, Duarte A (2023) Variable neighborhood search approach with intensified shake for monitor placement. Networks 81:319–333

Cockayne EJ, Hedetniemi ST (1977) Towards a theory of domination in graphs. Networks 7:247–261

Cockayne EJ, Dawes R, Hedetniemi ST (1980) Total domination in graphs. Networks 10:211–219

Corcoran P, Gagarin A (2021) Heuristics for k-domination models of facility location problems in street networks. Comput Oper Res 133:105368

Faulin J, Juan AA (2008) The algacea-1 method for the capacitated vehicle routing problem. Int Trans Oper Res 15:599–621

Feo TA, Resende MG (1989) A probabilistic heuristic for a computationally difficult set covering problem. Oper Res Lett 8:67–71

Ferone D, Gruler A, Festa P, Juan AA (2019) Enhancing and extending the classical grasp framework with biased randomisation and simulation. J Oper Res Soc 70:1362–1375

Fleszar K, Hindi KS (2004) Solving the resource-constrained project scheduling problem by a variable neighbourhood search. Eur J Oper Res 155: 402–413. (Financial Risk in Open Economies)

Fleurent C, Glover F (1999) Improved constructive multistart strategies for the quadratic assignment problem using adaptive memory. INFORMS J Comput 11:198–204

Goddard W, Henning MA (2013) Independent domination in graphs: a survey and recent results. Disc Math 313:839–854

Grasas A, Juan AA, Faulin J, De Armas J, Ramalhinho H (2017) Biased randomization of heuristics using skewed probability distributions: a survey and some applications. Comput Ind Eng 110:216–228

Guha S, Khuller S (1998) Approximation algorithms for connected dominating sets. Algorithmica 20:374–387

Hansen P, Mladenović N, Brimberg J, Pérez JAM (2019) Variable neighborhood search. Springer, Berlin

Hansen P, Mladenović N (2006) First vs. best improvement: An empirical study. Discrete Appl Math 154: 802–817. (IV ALIO/EURO Workshop on Applied Combinatorial Optimization)

Haynes TW, Hedetniemi S, Slater P (1998) Fundamentals of domination in graphs. CRC Press, Boca Raton

Haynes TW, Hedetniemi ST, Henning MA (2020) Topics in domination in graphs, vol 64. Springer, Berlin
Haynes TW, Hedetniemi ST, Henning MA et al (2021) Structures of domination in graphs, vol 66. Springer, Berlin
Haynes TW, Hedetniemi SM, Hedetniemi ST, Henning MA (2002) Domination in graphs applied to electric power networks. SIAM J Disc Math 15:519–529
Haynes TW, Hedetniemi ST, Henning MA (2022) Domination in graphs: core concepts. Manuscript. Springer, New York
Henning MA (2009) A survey of selected recent results on total domination in graphs. Disc Math 309:32–63
Henning MA, Jafari Rad N (2012) Locating-total domination in graphs. Disc Appl Math 160:1986–1993
Hwang SF, Chang GJ (1991) The k-neighbor domination problem. Eur J Oper Res 52:373–377
Juan AA, Faulin J, Ferrer A, Lourenço HR, Barrios B (2013) Mirha: multi-start biased randomization of heuristics with adaptive local search for solving non-smooth routing problems. Top 21:109–132
Laskar R, Pfaff J, Hedetniemi S, Hedetniemi S (1984) On the algorithmic complexity of total domination. SIAM J Algeb Disc Methods 5:420–425
Levin MS (2020) On combinatorial optimization for dominating sets (literature survey, new models). arXiv preprint arXiv:2009.09288
Li R, Hu S, Zhao P, Zhou Y, Yin M (2018) A novel local search algorithm for the minimum capacitated dominating set. J Oper Res Soc 69:849–863
Ma Y, Cai Q, Yao S (2019) Integer linear programming models for the weighted total domination problem. Appl Math Comput 358:146–150
Mladenović N, Hansen P (1997) Variable neighborhood search. Comput Oper Res 24:1097–1100
Napoletano A, Martínez-Gavara A, Festa P, Pastore T, Martí R (2019) Heuristics for the constrained incremental graph drawing problem. Eur J Oper Res 274:710–729
Ore O (1962) Theory of graphs. In: Colloquium Publications. American Mathematical Society
Resende MG, Ribeiro CC (2016) Optimization by GRASP—greedy randomized adaptive search procedures. Springer, Berlin
Sarubbi JF, Mesquita CM, Wanner EF, Santos VF, Silva CM (2016) A strategy for clustering students minimizing the number of bus stops for solving the school bus routing problem. In: NOMS 2016-2016 IEEE/IFIP network operations and management symposium, IEEE. pp. 1175–1180
Tutte WT, Tutte WT (2001) Graph theory, vol 21. Cambridge university press, Cambridge
Wang F, Du H, Camacho E, Xu K, Lee W, Shi Y, Shan S (2011) On positive influence dominating sets in social networks. Theor Comput Sci 412:265–269
Zverovich V (2021) Modern applications of graph theory. Oxford University Press, Oxford