Mathematics and Computers in Simulation 228 (2025) 485–497

Contents lists available at ScienceDirect

Mathematics and Computers in Simulation

journal homepage: www.elsevier.com/locate/matcom

Original articles

Finding the minimum k-weighted dominating sets using heuristic algorithms

E. Barrena [a,*], S. Bermudo [a], A.G. Hernández-Díaz [a], A.D. López-Sánchez [a], J.A. Zamudio [b]

[a] Department of Economics, Quantitative Methods, and Economic History. Universidad Pablo de Olavide, ES-41013, Seville, Spain
[b] CISPA Helmholtz Center for Information Security, Saarbrücken, Germany

ARTICLE INFO

Keywords:
Edge-weight
Dominating set
Metaheuristic algorithm
Iterated greedy algorithm

ABSTRACT

In this work, we propose, analyze, and solve a generalization of the k-dominating set problem in a graph, when we consider a weighted graph. Given a graph with weights in its edges, a set of vertices is a k-weighted dominating set if for every vertex outside the set, the sum of the weights from it to its adjacent vertices in the set is bigger than or equal to k. The k-weighted domination number is the minimum cardinality among all k-weighted dominating sets. Since the problem of finding the k-weighted domination number is $\mathcal{NP}$-hard, we have proposed several problem-adapted construction and reconstruction techniques and embedded them in an Iterated Greedy algorithm. The resulting sixteen variants of the Iterated Greedy algorithm have been compared with an exact algorithm. Computational results show that the proposal is able to find optimal or near-optimal solutions within a short computational time. To the best of our knowledge, the k-weighted dominating set problem has never been studied before in the literature and, therefore, there is no other state-of-the-art algorithm to solve it. We have also included a comparison with a particular case of our problem, the minimum dominating set problem and, on average, we achieve same quality results within around 50% of computation time.

1. Introduction

The problem of finding the minimum dominating set in a graph, or some of its variants, can model many real situations such as the study of social networks (see [1]), the design of wireless sensor networks (see [2]), biological networks (see [3]) or even the location of facilities networks (see [4]). Since this problem is $\mathcal{NP}$-hard, the calculus of the cardinality of such a set has been extensively studied in the literature (see, for instance, Cockayne and Hedetniemi [5], Haynes et al. [6,7,8], Henning and Yeo [9], Cockayne et al. [10], Henning [11], Favaron [12], Li [13], Hansberg and Volkmann [14] and Bermudo et al. [15,16]), and authors have given many lower and upper bounds for that parameter. To formally define the problem, let $G = (V, E)$ be a simple graph of order $n = |V|$ and size $m = |E|$, being $|V|$ and $|E|$ the cardinality of the vertex and edge sets, respectively. For two adjacent vertices $u, v \in V$, we denote by uv the edge joining them, and we denote by $N(v)$ the set of adjacent vertices of v, that is, $N(v) = \{u : uv \in E\}$. A set $D \subseteq V$ is a *dominating set* if every vertex in $V \setminus D$ is adjacent to some vertex in D. The *domination number*, denoted by $\gamma(G)$, is the cardinality of the minimum dominating set. Although many authors have given the exact value of this parameter in particular graphs or bounds for it in general graphs (see, for instance, McCuaig and Shepherd [17], Reed [18] and Haynes et al. [6,7]), just

* Corresponding author.
E-mail address: ebarrena@upo.es (E. Barrena).

https://doi.org/10.1016/j.matcom.2024.09.010
Received 22 April 2024; Received in revised form 3 September 2024; Accepted 15 September 2024
Available online 18 September 2024
0378-4754/© 2024 The Authors. Published by Elsevier B.V. on behalf of International Association for Mathematics and Computers in Simulation (IMACS). This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

a few papers have addressed the problem of solving it in general graphs using metaheuristics, we refer the reader to Potluri and Singh [19,20], Chalupa [21] and Casado et al. [22].

A variant of a dominating set is the k-dominating set. A set $D \subseteq V$ is a *k-dominating set* if every vertex in $V \setminus D$ is adjacent to at least k vertices in D. The *k-domination number*, denoted by $\gamma_k(G)$, is the cardinality of the minimum k-dominating set. From a theoretical point of view, the problem of finding bounds for the k-domination number has been widely studied, see for example [14,23–26]. A recent work that addresses the minimum k-dominating set problem using heuristic algorithms was published by Nguyen et al. [27]. The authors solved real social networks with up to 36,000 vertices and 200,000 edges.

In this work we propose a generalization of the k-domination number in a graph, when we consider an edge-weighted graph. This variant is coined as k-weighted domination number and it is computed as the minimum cardinality among all k-weighted dominating sets. A k-weighted dominating set must satisfy that, for every vertex v outside the set, the sum of the edge weights from v to the vertices in the set is greater than or equal to k. This new variant of the k-dominating set problem models some real-world problems previously mentioned but, in this case, the network edges have associated weights. For example, if we study social networks we could consider that the influence of one user on another user may be different. Related to the design or location of wireless sensor networks it may happen that an user receives signal from several sensors with different intensity. Regarding logistics, an application of this problem could be the case of freight transportation when there are constraints on the edge capacities, e.g., on the vehicles performing the corresponding route. The problem of optimizing distribution locations in these cases can be modeled by means of the problem we propose, but not by any of the existing domination-related problems in the literature. Moreover, from a theoretical point of view, generalizations of well-known problems enables to extend the knowledge and to provide better insights.

The main contributions of this paper are highlighted next:

- We generalize the k-dominating set problem when, instead of unweighted graphs, we consider weighted graphs resulting in a new problem, the k-weighted dominating set problem (k-WDSP).
- The k-WDSP considers weights in edges representing the flow exchange between vertices and, to the best of our knowledge, the k-WDSP has never been considered before in the literature.
- We solve the problem using several variants of an Iterated Greedy (IG) algorithm, which is able to solve graphs with different sizes and structures.
- Different destruction and reconstruction procedures have been proposed and included in the considered algorithm taking advantage of the knowledge of the problem.
- We compare the resulting 16 variants of an Iterated Greedy algorithm to address the k-WDSP, arising when combining the proposed destruction and reconstruction procedures.
- We have compared the best variant of the IG algorithm with an exact procedure to test the potential of our proposal and we are able to solve large instances in which the exact algorithm gets stuck.
- Since there is no other state-of-the-art algorithm to solve the k-WDSP, we have also included a comparison with a particular case of our problem, the minimum dominating set problem recently solved by Casado et al. [22], with the aim of evaluating the contribution of our algorithmic proposal.

This work is structured in the following way. Section 2 defines the problem including a mathematical formulation and some observations. Section 3 describes general and specific details of the algorithmic proposal to solve the problem of interest in this paper, the k-WDSP. Section 4 presents the computational results to test the quality of the proposal, including the fine-tuning phase of the most influential parameters (Section 4.2.1) and a comparison against an exact algorithm and against a state-of-art algorithm that solves a version of the proposed problem (Section 4.2.2). Finally, Section 5 includes the main conclusions of this research and some guides for further research.

2. Problem definition

This section is focused on formally defining the proposed problem, including some analytical observations and a mathematical formulation.

Let $G = (V, E, w)$ be an edge-weighted graph, where $w : E \to R^+$ is a function that assigns a positive value to any edge $uv \in E$. An unweighted graph could be considered as a weighted graph if this positive value is always equal to one. A set $D \subseteq V$ is a *k-weighted dominating set* (k-WDS) if $\sum_{v \in D} w(uv) \geq k$ for every vertex $u \in V \setminus D$, where we consider that $w(uv) = 0$ if $uv \notin E$. Furthermore, we say that a vertex $u \in V \setminus D$ is k-weighted dominated by D if $\sum_{v \in D} w(uv) \geq k$, we denote by $B(D)$ the set of all k-weighted dominated vertices by D and by $C(D) = V \setminus (D \cup B(D))$ the vertices that are neither in the k-weighted dominating set nor in the k-weighted dominated set. The *k-weighted domination number*, denoted by $\gamma_{kw}(G)$, is the minimum cardinality among all k-weighted dominating sets. The minimum k-weighted dominating set problem (k-WDSP) is the problem of finding a minimum k-weighted dominating set in the weighted graph G.

2.1. Observations

Based on this new definition, we can make the following observations.

Observation 2.1. *If $w(uv) = 1$ for any edge $uv \in E$ and $k = 1$, then the k-weighted dominating set becomes the usual dominating set.*

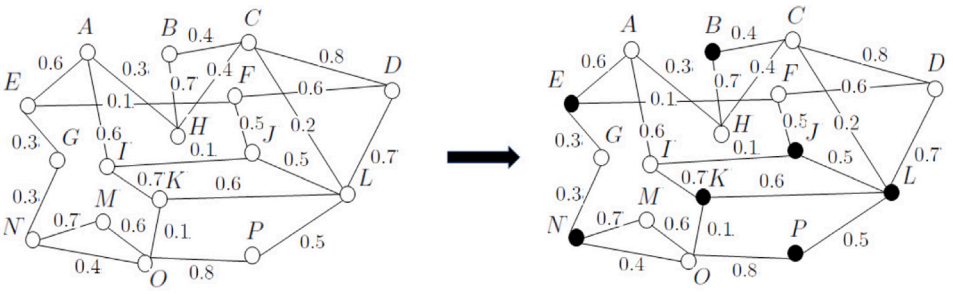


Fig. 1. An example of a weighted graph and a 0.6-weighted dominating set (black vertices).

Observation 2.2. *If $w(uv) = 1$ for any edge $uv \in E$, then the k-weighted dominating set becomes the usual k-dominating set.*

Observation 2.3. *If $w(uv) = 1/k$ for any edge $uv \in E$, then the 1-weighted dominating set becomes the usual k-dominating set in the same graph without the weights in its edges.*

Observation 2.4. *If $w(uv) \geq k$ for any edge $uv \in E$, then the k-weighted dominating set becomes the usual dominating set in the same graph without the weights in its edges.*

Observation 2.5. *If D is a k-weighted dominating set and there exists a vertex v such that $\sum_{u \in N(v)} w(uv) < k$, then $v \in D$. That is, if the sum of weights that a vertex receives from its adjacent vertices is smaller than k, then this vertex belongs to the k-weighted dominating set.*

Fig. 1 includes a weighted graph on the left and a 0.6-weighted dominating set of seven vertices (B,E,J,K,L,N,P) on the right. Note that the set depicted in such figure is a 0.6-weighted dominating set, but it is not the minimum one, given that it would be a set performed by six vertices (A, F, G, H, L, O).

2.2. Mathematical model

In order to mathematically model the k-WDSP, an Integer Linear Programming (ILP) formulation is included. Given a weighted graph G with vertices $V = \{v_1, v_2, \ldots, v_n\}$, we denote by $W = (w_{ij})$ the weight matrix, a square $n \times n$ matrix where w_{ij} represents the weight of the edge $v_i v_j$, this means, $w_{ij} = w(v_i v_j)$ when $v_i v_j \in E$ and $w_{ij} = 0$ when $v_i v_j \notin E$. Without loss of generality along this manuscript we consider that $w_{ij} \in [0, 1]$ to easily rescale the instances that are solved in the Computational Results Section. The ILP formulation uses binary variables x_i that take value 1 if vertex v_i is selected to be part of the set and 0 otherwise. The problem can be formulated as follows:

$$\min \sum_{i=1}^{n} x_i \tag{1}$$

$$\text{s.t.} \sum_{i=1}^{n} w_{ij} x_i \geq k(1 - x_j) \quad \forall j = 1, \ldots, n, \tag{2}$$

$$x_i \in \{0, 1\} \quad \forall i = 1, \ldots, n. \tag{3}$$

The objective function (1) represents the total number of selected vertices, that is, the cardinality of a k-WDS, while constraints (2) ensure that each non-selected vertex receives a total weight of at least k units from the selected vertices in the k-WDS. Finally, constraints (3) define the binary nature of the decision variables.

Note that Haynes et al. [6] proved that the problem of finding a minimum dominating set is $\mathcal{NP}$-hard, which implies that the k-WDSP is also $\mathcal{NP}$-hard, since the former is a particular case of the later when $k = 1$ and all edges have a weight equal to one. Given that the k-WDSP is $\mathcal{NP}$-hard for arbitrary weighted graphs, we propose to use a metaheuristic to solve it since, as it is proved in Section 4, when graphs have no specific structures and the size is large, the ILP formulation is not able to solve the k-WDSP.

3. Algorithmic proposal

This section presents a description of the algorithmic proposal to find the minimum k-WDS for any graph, no matter the size or the structure. Specifically, we have opted by an Iterated Greedy (IG) metaheuristic to address the considered problem. IG was first proposed by Ruiz and Stützle [28] to solve a combinatorial optimization problem, the flowshop scheduling problem. Since then, the IG has been successfully applied to handle many different combinatorial optimization problems. We refer the reader to some research works about real problems that can be modeled using graphs, such as labeling problems [29], the minimum dominating set problem, Casado et al. [22], community detection problems [30], the location of the minimum number of regenerators in networks [31], or even a multi-objective facility location problem [32], among others.

In general terms, the IG is based on the idea of starting from an initial solution that is partially destroyed and subsequently repaired. Those two phases, known as the destruction and reconstruction phases, are iteratively applied to each incumbent solution finding a sequence of solutions along the procedure until an acceptance or stopping criterion is met. The rationale behind the IG algorithm is that the construction of the initial solution and also the reconstruction of the partially destroyed solution iterate over greedy methods. It is important to mention that the destruction of the incumbent solution could also be an iterative greedy method. However, and as expected, if the construction of the initial solution, the partial destruction of the incumbent solution and its reconstruction are all based on greedy algorithms, we are intensifying the search in the same region of the search space and that implies that the solution may get stuck in local optima. To introduce diversification and explore different regions of the search space, randomization must be introduced in the IG framework. In this sense, the most common phase to introduce diversification is the destruction one.

3.1. Iterated Greedy algorithm

In this section, the general pseudocode of the IG to solve the k-WDSP is detailed, see Algorithm 1.

Algorithm 1 IG($G = (V, E, w), \beta, \Delta$)

```
1:  D ← InitialSolution(G)
2:  δ ← 0
3:  while δ < Δ do
4:      D_d ← Destruction(D, β)
5:      D_r ← Reconstruction(D_d)
6:      if |D_r| < |D| then
7:          D ← D_r
8:          δ ← 0
9:      else if |D_r| = |D| then
10:         R ← generateRandomNumber(0, 1)
11:         if R ≤ 0.5 then
12:             D ← D_r
13:             δ ← 0
14:         end if
15:     else
16:         δ = δ + 1
17:     end if
18: end while
19: return D
```

The IG algorithm requires three input parameters: the weighted graph, $G = (V, E, w)$; the proportion of vertices that will be removed in the destruction phase, β; and the stopping criterion, which is the maximum number of consecutive iterations without improvements, Δ.

The IG algorithm always needs an initial solution, D, (step 1). In our case, and as it is described in Section 3.2, we propose two different ways to obtain an initial solution, randomly or problem-based. Furthermore, as a part of the procedure to built an initial solution, the *redundancy check* of vertices is applied in order to disregard useless vertices. Up to this point, the initial solution is a minimal k-weighted dominating set, but not necessarily the one with minimum cardinality.

Next, the algorithm performs a number of iterative improvements to enhance the quality of the solution (steps 3–18). Specifically, a maximum number of iterations is performed while no improvements are found, which is controlled by the input parameter Δ. Each iteration involves a partial destruction of the solution based on the parameter β (step 4), see Section 3.3 where details are included. More precisely, a predefined proportion β of the vertices belonging to the solution is removed and, subsequently, the algorithm reconstructs the solution by adding vertices (step 5) following any of the procedures that will be described in Section 3.4, until the dominance condition is met. At this juncture, we have a different solution compared to the incumbent one, D, and if its cardinality is smaller (step 6), it qualifies as a better solution (step 7). In cases where the cardinalities are equal (step 9), one of the two solutions is randomly chosen (step 10). If the reconstructed solution has a larger cardinality, it is discarded. As previously mentioned, this iterative loop continues until Δ consecutive iterations are executed without any improvement, thus signifying convergence. However, if we encounter an improvement during an iteration, the counter tracking the number of iterations without improvement is reset to zero (step 8 and 13). Furthermore, in such cases, we update the solution with the best solution found. This mechanism ensures that the algorithm continues refining the incumbent solution as long as enhancements are being made, allowing the possibility of continually improving the solution until the convergence criterion is met. Otherwise, if the algorithm is not able to improve the solution, the number of iterations without improvement increases in one unit (step 16) and the procedures finishes when the number of iterations without improvement is the maximum allowed, i.e., when $\delta = \Delta$. The IG returns the best set, D, found during the search (step 19).

Having explained the general framework of the IG algorithm, we now detail each of its parts to solve the considered problem.

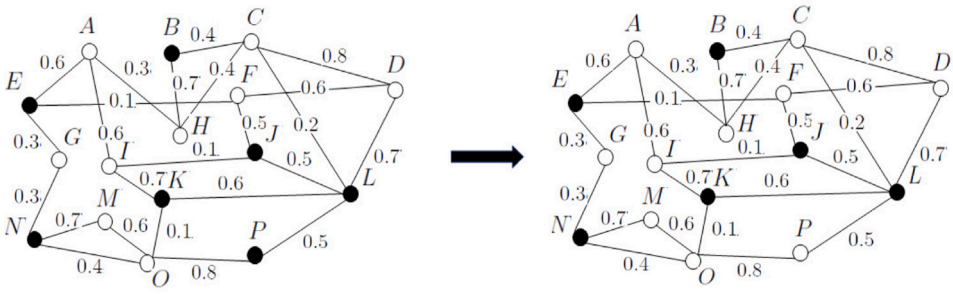


Fig. 2. The set obtained after removing a vertex from D following the DD criterion.

3.2. Initial solution

The IG algorithm always starts from an initial solution that must be given or constructed. The initial solution could be randomly constructed or could be a solution in which the knowledge of the k-WDSP is exploited. As expected, a random initial solution would be found quickly although we do not know how good it would be in terms of quality. However, implementing a greedy procedure would yield a good initial solution which might be advisable with respect to the quality, but the computing time to get it would be high.

In this paper, we propose to construct the initial solution by greedily inserting the most promising vertex at each iteration. Initially, the procedure starts from an empty solution, $D \leftarrow \emptyset$, and at each step a new vertex is included into the solution until D is a k-weighted dominated set, probably not being the minimum. To decide the vertex to be inserted at each step, we measure the contribution of each one to the set and we select the one that provides the maximum contribution. The contribution of a vertex $v \in V \setminus D$ to the set D is measured as the number of new k-weighted dominated vertices by the set $D \cup \{v\}$. Formally, the contribution of a vertex $v \in V \setminus D$ is measured as $|B(D \cup \{v\})| - |B(D)|$. If two vertices yield the same contribution, one of them would be randomly selected to be included into the set D.

Finally, since the set D might have unnecessary vertices, we implement a procedure named *redundancy check*. This verification consists in checking if some vertices in the set are useless, that is, if the set remains a k-WDS when these vertices are eliminated from the set D. Formally, the *redundancy check* applied to a set D removes from D the vertices $v \in D$ such that $B(D) \cup \{v\} = B(D \setminus \{v\})$. From a heuristic point of view, this can be considered as a local search algorithm since it performs some iterative changes in order to improve the solution until a stopping criterion is met, in this case until no further improvements are achieved.

3.3. Destruction method

Once we have a feasible solution D, the IG algorithm partially destructs such solution, in particular $\lfloor \beta \cdot |D| \rfloor$ vertices are removed from D, where $\beta \in [0, 1]$. The value of β is tuned in Section 4 to measure its impact on the quality of the solutions found by the algorithm. As expected, small values of β barely modify the incumbent solution, while large values of β construct a practically new solution.

The destruction phase can be performed following either a random or a greedy criterion depending on if our interest is to diversify or intensify, respectively. In this work, both alternatives are considered to evaluate the effect of each of them in the performance of the final configuration of the IG algorithm. Therefore, we have proposed one random destruction and three greedy destruction procedures depending on the criterion followed to remove vertices. The three different greedy destruction procedures are based on our attempt to introduce knowledge of the problem and help the algorithm to obtain better solutions in less computational time.

As we could deduce, the *random destruction*, denoted as RD, removes $\lfloor \beta \cdot |D| \rfloor$ vertices from the incumbent solution D. Those vertices are selected at random, yielding an unfeasible solution. The random destruction allows the algorithm to explore solutions in new search spaces that, otherwise, would be never considered and this introduces diversification.

The other three criteria, named *greedy destruction procedures*, remove $\lfloor \beta \cdot |D| \rfloor$ vertices from the incumbent solution D, but following a greedy criterion, that is, the best option based on a specific criterion. Opposite to the random destruction, any greedy destruction follows an intensification strategy since it removes the least promising vertices according to the greedy function considered to destruct the solution. We have considered the following three greedy destruction criteria.

(1) The *greedy destruction based on vertex degree*, denoted as DD, iteratively removes the vertex $v \in D$ with fewer neighbors in $V \setminus D$, no matter the weight of the adjacent edges.
The left hand side of Fig. 2 represents an example of an initial 0.6-weighted dominating set and the right hand side represents which vertex must be removed from the set D following the DD criterion. As we can see, vertex P has one neighbor in $V \setminus D$, and the remaining vertices in D have two or three neighbors in $V \setminus D$. Therefore, the candidate to be removed from D is vertex P.

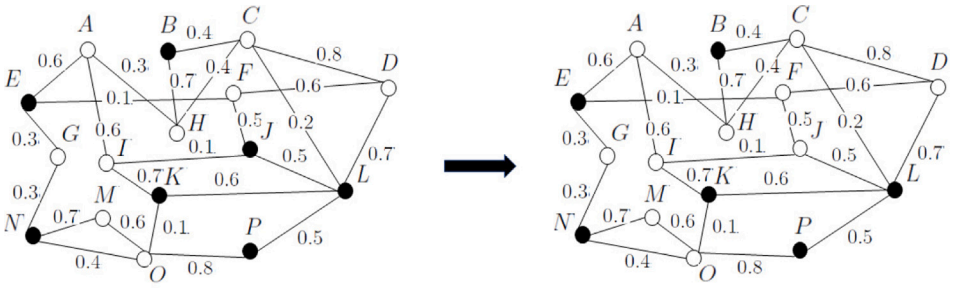


Fig. 3. The set obtained after removing a vertex from D following the WD criterion.

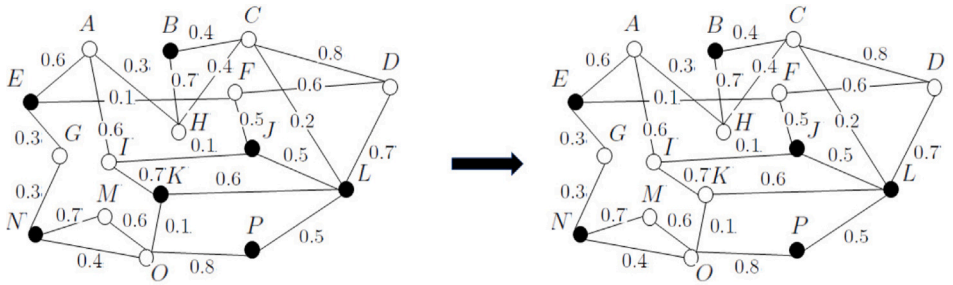


Fig. 4. The set obtained after removing a vertex from D following the CD criterion.

(2) The *greedy destruction based on weights*, denoted as WD, takes into account the sum of the weights to eliminate vertices. The method iteratively removes the vertex $v \in D$ such that the value $wd(v) := \sum_{u \in N(v) \cap B(D)} w(vu)$ is minimum.
Again, the left hand side of Fig. 3 represents an example of an initial 0.6-weighted dominating set and the right hand side represents which vertex must be removed from the set D following the WD criterion. As we can see, according to this criterion, wd of vertex J is 0.6; wd of vertices K and P are equal to 0.8; wd of vertex L is 0.9; wd of vertex E equals 1.0; wd of vertex B equals to 1.1; and wd of vertex N is 1.4. Therefore, the only candidate to leave the solution is vertex J.

(3) The *greedy destruction based on the completeness*, denoted as CD, iteratively removes the vertex $v \in D$ that yields the largest $B(D \setminus \{v\})$ set, which obviously corresponds to the vertex that yields the smallest $|B(D)| - |B(D \setminus \{v\})|$. Therefore, this destruction criterion removes the vertex such that its absence worsens as little as possible the set of k-weighted dominated vertices.
Let us continuous with the same graph and set D representing a 0.6-weighted dominating set on the left side of Fig. 4. The right hand side represents which vertex must be removed from the set D following the CD criterion. As can be observed in Fig. 4, if we remove vertex E from the set D, then $|B(D)| - |B(D \setminus \{\mathrm{E}\})| = 3$, since A, G and F are not dominated by the set $D \setminus \{\mathrm{E}\}$, that is, if we remove vertex E from set D, the set of k-weighted dominated vertices decreases in three units. In case of eliminating the vertex B from the set D, then $|B(D)| - |B(D \setminus \{\mathrm{B}\})| = 2$, since H and C are not dominated by the set $D \setminus \{\mathrm{B}\}$. Analogously happens with vertex N. When removing vertex J from the set D, $|B(D)| - |B(D \setminus \{\mathrm{J}\})| = 1$, because F is not dominated by $D \setminus \{\mathrm{J}\}$. The same applies to P, because $\mathrm{O} \notin B(D \setminus \{\mathrm{P}\})$. A special situation occurs with vertex L, where $|B(D)| - |B(D \setminus \{\mathrm{L}\})| = 1$ since D and C are not dominated by the set $D \setminus \{\mathrm{L}\}$ but $\mathrm{L} \in B(D \setminus \{\mathrm{L}\})$. Finally, if we eliminate vertex K from the set D, then $|B(D \setminus \{\mathrm{K}\})| = |B(D)|$ because $\mathrm{I} \notin B(D \setminus \{\mathrm{K}\})$ but $\mathrm{K} \in B(D \setminus \{\mathrm{K}\})$. Therefore, vertex K is the candidate to be removed following the CD criterion, since $|B(D)| - |B(D \setminus \{\mathrm{K}\})|$ is the smallest value among all the candidates.

3.4. Reconstruction method

In the reconstruction phase we use four procedures similar to those used in the destruction part. The first one is a *random reconstruction*, denoted as RR, that is, we randomly add vertices until the set D is again a k-WDS, that is, until $D \cup B(D) = V$. The three *greedy construction* procedures consider the following three greedy construction criteria.

(1) The *greedy reconstruction based on vertex degree*, denoted as DR, iteratively adds the vertex $v \in V \setminus D$ with more neighbors in $C(D)$, no matter the weight of such edges.
If we take $k = 0.6$, the left hand side of Fig. 5 represents a graph with an infeasible solution. In this Figure, $D = \{\mathrm{B}, \mathrm{L}, \mathrm{N}\}$ (black vertices), $B(D) = \{\mathrm{C}, \mathrm{D}, \mathrm{H}, \mathrm{K}, \mathrm{M}\}$ (gray vertices) and $C(D) = \{\mathrm{A}, \mathrm{E}, \mathrm{F}, \mathrm{G}, \mathrm{I}, \mathrm{J}, \mathrm{O}, \mathrm{P}\}$ (white vertices). One vertex must be added to the partial solution D following the DR criterion. As we can see, vertex C has no neighbors in $C(D)$; vertices D, G, H, M, O and P have one neighbor in $C(D)$; vertices A, F, I, J and K have two neighbors in $C(D)$ and finally, vertex E has three neighbors in $C(D)$. Therefore, the candidate to be part of the solution is vertex E since it has more neighbors.

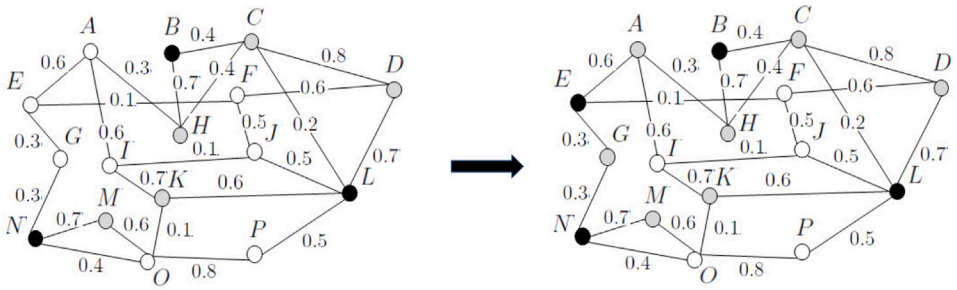


Fig. 5. The set obtained after adding a vertex to D following the DR criterion.

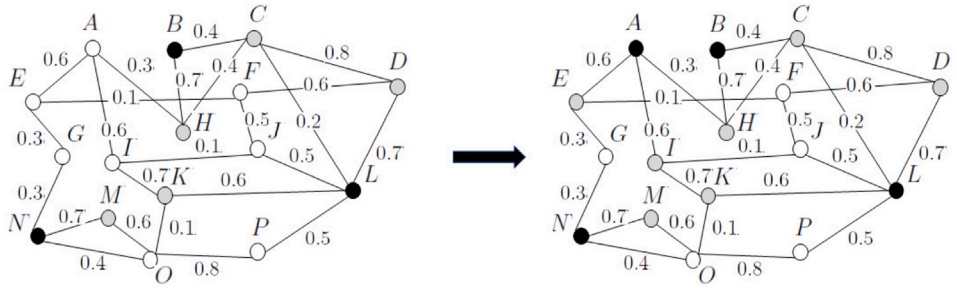


Fig. 6. The set obtained after adding a vertex to D following the WR criterion.

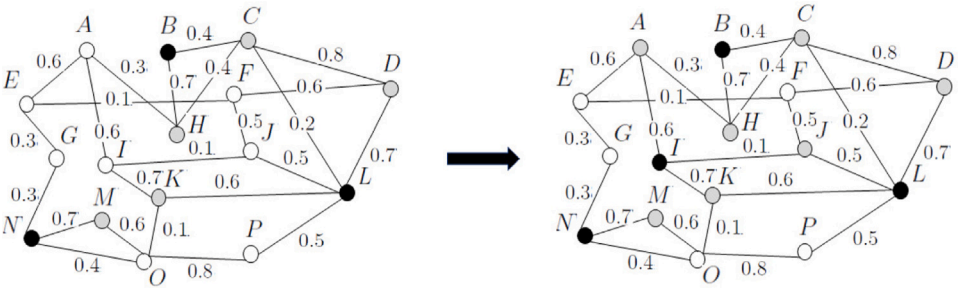


Fig. 7. The set obtained after adding a vertex to D following the CR criterion.

(2) The *greedy reconstruction based on weights*, denoted as WR, is the opposite of its corresponding destruction criterion. The method iteratively adds the vertex $v \in V \setminus D$ such that the value $wr(v) := \sum_{u \in N(v) \cap C(D)} w(vu)$ is maximum.
Let us continuous with the same graph, the left hand side of Fig. 6 represents a graph with an infeasible solution, $D = \{B, L, N\}$ and one vertex must be added to the partial solution D following the WR criterion. The values of the vertices in $V \setminus D$ are the following: wr of vertex A is equal to 1.2; wr of vertex E equals 1.0; wr of vertices K, O and P are equal to 0.8; wr of vertex I equals 0.7; wr of vertices D, F, J and M are equal to 0.6; wr of vertices G and H are equal to 0.3; and finally, wr of vertex C equals zero. Therefore, vertex A is the candidate to be added to the set D.

(3) The *greedy reconstruction based on the completeness*, denoted as CR, iteratively adds the vertex $v \in V \setminus D$ that yields the biggest value for $|B(D \cup \{v\})| - |B(D)|$.
Again, the left hand side of Fig. 7 represents a graph with an infeasible solution, $D = \{B, L, N\}$ and one vertex must be added to the partial solution D following the CR criterion. The above value for the vertices A, E and I are equal to 2; for the vertices F, O and P are equal to 1; for the vertices D, G, J, K and M are equal to 0; and finally, for the vertices C and H are equal to −1. Therefore, the candidates vertices are A, E and I and one of them is randomly selected to be part of the set D, for example vertex I.

It is important to emphasize that once that a solution is reconstructed following any of the explained criteria, we apply the *redundancy check* again, acting as a local search algorithm to remove unnecessary vertices in the obtained set.

4. Computational results

This section is devoted to solve the k-WDSP and discuss the performance of our algorithmic approach. To that end, a total of 156 instances are solved on an AMD Ryzen 9 5950x 16-core processor × 32 with 128 GB RAM. The algorithmic proposal has been implemented using Java 17, while the exact algorithm has been solved using Gurobi 10.0.1. We have structured the section as follows. First, a description of the instances has been done and, secondly, the computational experiments are included to test our proposal. Specifically, the experiments are divided into preliminary and final ones. The first ones to tune the algorithm parameters, and the second ones to evaluate the performance of the final configuration of our proposal.

4.1. Instances

The set of instances that have been solved is divided into two different subsets. Both subsets are based on instances from the literature, to which we randomly add edge weights from the set $\{0.1, 0.2, \ldots, 0.9, 1\}$ maintaining the instance structure (V, E). These sets are the following:

- 64 small-size instances based on the ones presented in [33]. The size of these instances varies from 100 to 400 vertices, and from 1029 to 40 120 edges.
- 76 randomly generated large-size instances that are solved in the paper of Casado et al. [22] (available in the repository https://grafo.etsii.urjc.es/en/publication/casado-202341/). In these instances, the number of vertices varies between 500 and 4000, and the number of edges ranges between 25 200 and 4 010 000 with a density[1] between 20% and 50%.

These 140 instances are available at https://zenodo.org/record/8318881. Note that, even though 140 instances have been considered, a total of 156 instances appear in this repository since, in the preliminary experiments, we have tested the impact of changing the edge weights to 16 instances.

When it comes to resolving instances, for each of them we have considered different values of k, specifically the considered k-values have been 0.10, 0.25, 0.50, 0.75, and 1.

4.2. Experiments

The experiments are divided into two different sections: preliminary and final results. The preliminary experiments randomly consider 20% of the large-size instances and focus on tuning the input parameters of the IG algorithm and on selecting the best strategy and variant to solve the considered problem. Furthermore, some of the analytical observations are empirically tested. The first experiment, included in the Final Result Section, compares the exact algorithm against the best variant of the IG. As far as we are aware, there is no other state-of-the-art algorithm to solve the k-WDSP. Therefore, with the aim of evaluating the contribution of our proposal, we have also included a comparison with a particular case of our problem, the minimum dominating set problem recently solved by Casado et al. [22].

Tables reported in this manuscript include some of the following metrics:

- **OF**, the average objective function value obtained by each algorithm over the set of considered instances.
- **T(s)**, the average computing time required by the algorithm (in seconds).
- **Dev(%)**, the average percentage deviation with respect to the best solution found by the methodologies under comparison, that is, $\frac{OF - best\ OF}{best\ OF} \cdot 100$.
- **Gap(%)**, the average optimality gap calculated as $\frac{UB - LB}{UB} \cdot 100$, where LB and UB are the best known lower and upper objective bounds.

Note that average values are shown since, as previously mentioned, all the instances have been solved considering five different values of k.

4.2.1. Preliminary results

As previously mentioned, this section is devoted to tune the input parameters of the IG algorithm and to select the best strategy and variant to solve the k-WDSP. Furthermore, the analytical observations are tested by means of some empirical experiments. To that end, and in order to avoid overfitting, the preliminary experimentation is performed over a subset of 16 representative instances out of the 76 large-size instances randomly generated.

Experiment 1. The first experiment is devoted to compare the 16 IG variants arising when combining in pairs the destruction and reconstruction procedures presented in Sections 3.3 and 3.4, respectively. Each IG variant, named as AxB in the first column of Table 1, indicates that the destruction method A and the reconstruction method B are combined. Specifically, Table 1 shows these results on average, for the considered instances and k-values, to assess the impact of such combinations. The method starts from an initial solution that has been obtained using the greedy construction explained in Section 3.2. To be fair, all the IG combinations start from the same initiating solution. Furthermore, the remaining parameters have been preset to the same values. More precisely,

Table 1
Comparison of the average objective function value, OF, and execution times, T (s), of the 16 IG variants arising when combining destruction and reconstruction procedures in pairs. Best (lower) results are in bold font.

IG variants	OF	T (s)	Dev (%)
RDxRR	14.74	14.42	1.03%
RDxDR	14.70	**13.91**	0.77%
RDxWR	14.73	14.24	0.94%
RDxCR	**14.59**	17.13	**0.00%**
DDxRR	14.66	15.07	0.51%
DDxDR	14.83	14.60	1.63%
DDxWR	14.68	14.98	0.60%
DDxCR	14.66	73.03	0.51%
WDxRR	14.66	16.80	0.51%
WDxDR	14.70	15.21	0.77%
WDxWR	14.68	15.35	0.60%
WDxCR	14.75	74.51	1.11%
CDxRR	14.69	18.97	0.69%
CDxDR	14.65	18.74	0.43%
CDxWR	14.71	17.59	0.86%
CDxCR	14.66	48.49	0.51%

Table 2
Impact of the initial solution, considering the 16 IG variants, on the average objective function value, OF, and execution times, T (s). In both cases lower is better. Best results of the OF are in bold font.

IG variants	Random		Greedy		Dev (%)
	OF	T (s)	OF	T (s)	
RDxRR	18.21	0.20	**14.74**	14.42	23.54%
RDxDR	18.31	0.05	**14.70**	13.91	24.56%
RDxWR	18.34	0.08	**14.73**	14.24	24.51%
RDxCR	18.20	9.41	**14.59**	17.13	24.74%
DDxRR	17.14	9.26	**14.66**	15.07	16.92%
DDxDR	18.79	0.33	**14.83**	14.60	26.70%
DDxWR	18.80	0.46	**14.68**	14.98	28.07%
DDxCR	16.49	92.08	**14.66**	73.03	12.48%
WDxRR	17.06	8.86	**14.66**	16.80	16.37%
WDxDR	18.88	0.46	**14.70**	15.21	28.44%
WDxWR	18.94	0.54	**14.68**	15.35	29.02%
WDxCR	16.60	86.26	**14.75**	74.51	12.54%
CDxRR	17.04	11.81	**14.69**	18.97	16.00%
CDxDR	18.14	3.66	**14.65**	18.74	23.82%
CDxWR	18.16	3.56	**14.71**	17.59	23.45%
CDxCR	17.43	43.44	**14.66**	48.49	18.89%
Average	17.91	16.90	**14.69**	25.19	22.00%

the destruction parameter is $\beta = 0.2$ and the maximum number of iterations without improvements has been fixed to $\Delta = 160$, as done in [22].

As expected, see Table 1, the fastest variant of the IG is the one that combines the random destruction and the greedy reconstruction based on the degree (RDxDR), although this is not the one yielding the best performance. The IG variant that yields the best quality is the one that randomly destroys solutions and that uses a greedy reconstruction based on the completeness (RDxCR). The random destruction introduces diversification in the search, exploring different search spaces, while the greedy reconstruction intensifies in such new search spaces. Bold values show the best value for the objective function (OF), execution time (T(s)) and deviation (Dev(%)).

Experiment 2. The second experimentation measures the impact of the initial solution in the IG algorithm. We would like to check if the performance of the algorithm is similar independently of the initial solution. To carry out this test, the algorithm starts from an initial solution randomly built and from an initial solution obtained by means of the greedy construction. Again, the 16 IG variants have been compared under this assumption. As considered in the previous experiment, we have destructed a 20% of the solution, i.e., $\beta = 0.2$ and $\Delta = 160$ iterations without improvements has been fixed in order to stop the algorithm.

Regarding the quality of the final solution, see Table 2, it can be observed that, when the algorithm starts from a greedy initial solution, it yields better results in terms of the objective function, even though the computational time is larger in almost all the IG variants. Bold values show the average results and the best values for the objective function (OF). In conclusion, and according

[1] The density of a graph $G = (V, E, w)$ is computed as $\frac{2|E|}{|V|(|V|-1)}$.

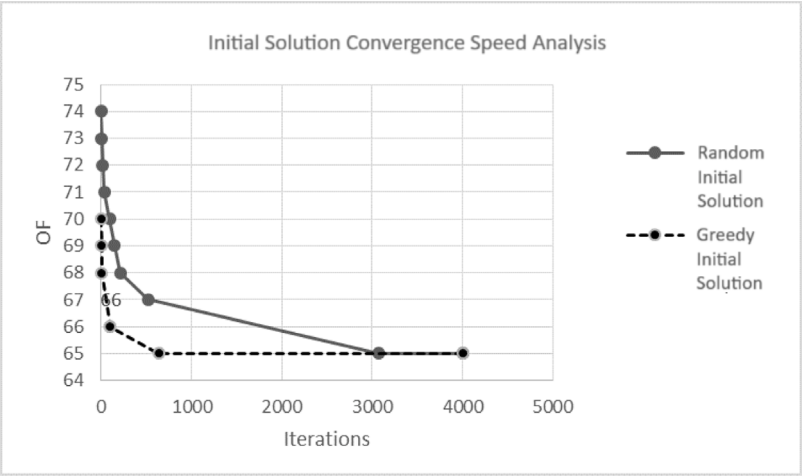


Fig. 8. Convergence speed depending on the initial solution.

Table 3
Impact of the β parameter on the average objective function value, OF, and execution times, T (s). In both cases lower is better. Selected results are in bold font.

β	OF	T (s)
0.1	14.62	11.00
0.2	**14.61**	**12.00**
0.3	14.60	12.86
0.4	14.59	13.73

to the average results, starting from an initial random solution the algorithm obtains an average objective function of 17.91 in 16.90 s versus an average objective function of 14.69 in 25.19 s if the algorithm starts from a greedy initial solution constructed as explained in Section 3.2.

In order to graphically show the convergence speed of the IG algorithm depending on the initial solution (random versus greedy), we have solved one instance without limiting the number of iterations. To depict this experiment, we have selected one challenge instance with 1000 vertices and 200 300 edges and we have fixed the value of $k = 10$. Fig. 8 represents how the objective function value converges when the number of iterations increases. This figure starts form an initial solution randomly obtained and from an initial solution built with the greedy procedure.

As we can observe, when we start from a random initial solution, the algorithm does not converge quickly. Table 2 considers 160 iterations without improvement in both cases and the quality of the solutions obtained starting from a random initial solution is worse than the ones obtained starting from a greedy initial solution. However, as it can be observed in Fig. 8, more iterations than 160 iterations would be necessary to obtain solutions of similar quality. Summarizing, the IG algorithm converges much faster (less than 1000 iterations) when starting with a more sophisticated initial solution than when starting with a random initial solution (3000 iterations).

Experiment 3. Next experiment measures the impact of the β parameter in the results. To perform this analysis, we have kept the IG variant that attains the best solutions, i.e., the one that randomly destroys solutions and greedily reconstructs the solutions using the complete criteria (RDxCR). For this experiment, we have considered the following β values: $\beta = \{0.1, 0.2, 0.3, 0.4\}$ and β-values larger than 0.5 are not tested since this would imply destroying more than half of the entire solution. Again, $\Delta = 160$ iterations without improvements has been fixed in order to stop the algorithm.

As it can be observed in Table 3, results are quite similar among them in terms of objective function value when varying the β value. As expected, the larger the β value, the most time consuming is the algorithm. On the contrary, the larger the β value, the better the average objective function value since we are escaping from the local optimum. We have selected $\beta = 0.2$ since it seems a good trade-off between the solution quality and the computing time. Bold values show the best average results obtained with the selected β value for the objective function (OF) and the executing time (T(s)).

Experiment 4. We also perform a parametric analysis on the number of iterations without improvement (Δ). Again, to carry out this analysis, we consider the IG variant that achieves the best solutions and $\beta = 0.2$. The tested values for Δ have been 64, 96, 128, and 160.

In view of the results obtained in Table 4, the improvement of the objective function is negligible when the value of Δ increases. As expected, the computing time increases as Δ increases. Therefore, we have set the value of Δ to 64 iterations without improvements. Bold values show the best average values obtained with the selected Δ value for the objective function (OF) and the executing time (T(s)).

Table 4
Impact of Δ parameter on the average objective function value, OF, and execution times, T (s). In both cases lower is better. Selected results are in bold font.

Δ	OF	T (s)
64	**14.60**	**11.46**
96	14.60	12.17
128	14.61	12.75
160	14.59	13.20

Table 5
Comparison of the average objective function value, OF, and execution times, T (s), of the IG versus the exact algorithm for different graphs with different number of vertices, |**V**|, and edges, |**E**|. Best (lower) results of the OF are in bold font.

|**V**|	|**E**|	Greedy		Gurobi			Dev (%)
		OF	T (s)	OF	T (s)	Gap	
100	1 787.81	7.41	0.00	**7.30**	4.89	0.00	1.51%
200	7 073.44	9.15	0.00	**8.82**	872.83	0.04	3.74%
300	15 860.81	**10.00**	0.10	10.08	1484.88	0.14	0.79%
400	28 103.44	**11.00**	0.17	11.41	1738.31	0.26	3.59%
500	37 675.00	**11.73**	1.13	12.33	1800.00	0.51	4.87%
1000	175 325.00	**13.30**	1.23	13.60	1800.00	0.62	2.21%
2000	700 650.00	**14.96**	7.93	16.15	1800.00	0.69	7.37%
3000	1 575 975.00	**16.00**	19.40	21.08	1800.00	0.73	24.10%
4000	2 801 300.00	**16.71**	38.50	22.11	1800.00	0.74	24.42%

Experiment 5. To close the Preliminary Results section, we analyze the impact of the edge weights in the objective function value and in the convergence speed. To conduct this experiment, instead of randomly assigning weights to all the edges (see Section 4.1), half of the edges are randomly weighted between 0.1 and 0.3 and the other half between 0.8 and 1. In such a way, the edge weights are heterogeneous since more extreme values are assigned.

On average, we have obtained that when the edge weights are assigned following this logic of disparity, the average value of the objective function increases up to 15.49 versus 14.58 vertices. Similarly, the computational time also increases up to 13.09 versus 10.76 s. Therefore, these results suggest that when the weights are very different among them, the difficulty of the problem increases.

4.2.2. Final results

Once that all parameters have been tuned, we solve the k-WDSP in all the instances previously described with the best IG configuration. Recall that, according to the preliminary experiments, the final configuration combines RDxCR and fixes the parameters β equals to 0.2 and Δ equals to 64.

In this section, two experiments are conducted. The first one compares our algorithm against an exact algorithm, see Table 5. The second one compares our algorithm against the state-of-the art algorithm (SOTA) by fixing $k = 1$ and all the edge weights equal to 1, thus becoming the minimum dominating set problem, see Table 6.

As mentioned, the first experiment compares our algorithm and an exact algorithm (Branch and Bound algorithm truncated with a time limit equal to 1800 s). So, Table 5 presents the average results obtained by solving the problem formulation presented in Section 2.2 using Gurobi against those obtained with the proposed IG methodology. As commented in Section 4.1, each row shows the average results over the 80 instances obtained by varying the density and k-values. Bold values highlight the best values for the average objective function (OF) obtained at each row. As expected, for the small instances (100 and 200 vertices), the exact algorithm yields near-optimal solutions, but for instances with 300 vertices or more, the IG outperforms the exact algorithm, both in terms of the objective function values and computational time. Notice that, even though results obtained by the exact algorithm are better for the small instances, the computational time is significantly higher. Furthermore, the majority of the realistic situations are modeled with large graphs and in such situations the IG is able to attain the best values of the objective function in a short computation time.

As there is no other contribution that deals with the k-WDSP, to test our algorithmic proposal, we include a second final experiment that compares the results of our IG algorithm with the one applied to solve a particular case of the k-WDSP, the minimum dominating set problem [22]. As mentioned in the Observation 2.4, when the weights of all edges are greater than or equal to the k-value, the k-WDSP becomes the dominating set problem in non-weighted graphs. Since we consider random weights between 0.1 and 1, the k-WDSP becomes the dominating set problem in non-weighted graphs if $k = 0.1$.

Table 6 shows the results obtained with both algorithms. Bold values show the best values for the average objective function (OF) obtained at each row. Note that our objective is not to outperform the results obtained by Casado et al. [22], but to empirically see Observation 2.4. Observe that our IG proposal requires shorter computational time because the stopping condition of the IG algorithm is set to a maximum of 64 iterations without improvements whereas the SOTA algorithm fixes this parameter to 160 iterations. However, it is worth mentioning that both algorithms achieve similar quality solutions in terms of the objective function value even though our IG proposal requires shorter computational times than the SOTA algorithm.

Table 6
Comparison of the average objective function value, OF, and execution times, T (s), of the IG versus SOTA for different graphs with different number of vertices, |**V**|, and edges, |**E**|. Best (lower) results of the OF are in bold font.

|**V**|	|**E**|	IG		SOTA		Dev (%)
		OF	T (s)	OF	T (s)	
500	37 675	8.30	0.00	**8.20**	0.00	1.22%
1000	175 325	**9.31**	0.43	9.44	2.00	1.40%
2000	700 650	**10.67**	4.25	10.68	8.00	0.09%
3000	1 575 975	11.43	10.50	**11.38**	25.00	0.44%
4000	2 801 300	12.35	22.62	**12.33**	41.00	0.16%

In order to validate the results, non-parametric statistical analysis is performed. To that end, we perform the pairwise nonparametric Wilcoxon statistical test to compare the quality of results (in terms of the objective function value) and the computing times.

On one hand, we compare the results of our proposed Iterated Greedy algorithm and the results obtained by a truncated exact algorithm. As expected, when analyzing the average objective function values (12.28 versus 13.68, respectively), the null hypothesis is rejected, $p - value < 0.001$, which concludes that our proposal outperforms the truncated exact algorithm. A similar conclusion is reached regarding the computational time since, on average, the IG spends 7.69 s and the truncated exact algorithm takes 1445.82 s. Therefore, there are significant evidences to reject the hypothesis that the computational times spent by both algorithms are equal.

On the other hand, we consider a particular case of our problem and compare the results obtained by our proposal and the results obtained by the state-of-the-art algorithm [22]. As the average objective function values indicate (10.5 by both algorithms), the null hypothesis is not rejected, $p - value = 0.496$, which indicates that the quality of the results are equal. However, if we focus on the computational time, the state-of-the-art algorithm takes almost twice than our proposal. Therefore, there are significant evidences to reject the null hypothesis, that states that the computational times spent by both algorithms are equal. We can thus conclude that our algorithm is much faster. Nevertheless, this is not the main objective of the comparison. Our only aim is to check that the Observation 2.4 is correct and that our algorithm is competitive with the state-of-the-art algorithm.

5. Conclusions

We have proposed, analyzed and solved the problem of finding the minimum k-weighted dominating set in edge-weighted graphs. An analysis of some properties of the problem describes this as a generalization of other well-known variants of the dominating set problem.

We propose an ILP formulation of the problem and solve instances up to optimality up to nearly 200 vertices with Gurobi. In order to deal with larger size instances, we have proposed and compared sixteen variants of an Iterated Greedy algorithm against the exact algorithm. To evaluate the algorithms, we have solved a set of 156 instances based on instances from the literature to which we add random weights.

We have performed several experiments. As expected, preliminary results show that the IG variants that take more advantage of the problem characteristics yield better results. Random and problem-based initial solutions have been tested and results evidence that starting with a random solution takes around 32% less time but it achieves worse quality solutions (22% of deviation). Parametric analysis have been performed to tune the main input parameters of our IG algorithms. Different edge-weight patterns have been tested to check their impact on the resulting objective function, and we observe that the difficulty of the problem increases when the edge weight differences increase. To conclude, we have performed two final experiments with all the instances, the first comparing the best IG with the exact algorithm to test the limits of the exact procedure and the IG performance. For the small-size instances up to nearly 200 vertices, the ILP solver finds near-optimal solutions, with an average optimality gap of 0.41%, but it is time consuming, reaching the provided time limit of 30 min for instances with more than 400 vertices. Since, to the best of our knowledge, there is no other state-of-the-art algorithms to solve the k-WDSP, the second final experiment compares the results of our methodology with the one applied to solve a particular case of the proposed problem, the minimum dominating set problem [22]. Although computing times are not comparable since different computing resources are used, the objective function values are similar in both cases and our proposal requires shorter computation times. In particular, in average, we achieve same quality results within around 50% of computation time.

As future directions of research, it would be interesting to propose different methodologies to solve the k-WDSP such as matheuristics, which may be able to include mathematical features to the heuristic procedures. Furthermore, we realize that there exist many different variants of this class of combinatorial optimization problems that can solve real-life situations when including weights, not only in edges or arcs, but also in vertices. Moreover, it may also be interesting to explore other objective functions.

CRediT authorship contribution statement

E. Barrena: Writing – review & editing, Writing – original draft, Supervision, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization, Validation. **S. Bermudo:** Writing – review & editing, Supervision, Investigation, Formal analysis, Conceptualization, Validation. **A.G. Hernández-Díaz:** Writing – review & editing, Resources, Methodology,

Funding acquisition, Investigation, Formal analysis, Conceptualization, Validation. **A.D. López-Sánchez:** Writing – review & editing, Writing – original draft, Supervision, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization, Validation. **J.A. Zamudio:** Writing – original draft, Visualization, Software, Resources, Data curation, Investigation, Conceptualization, Validation.

Acknowledgments

This research has been partially supported by the "Ministerio de Ciencia e Innovación" through Project PID2022-139543OB-C41. S. Bermudo also acknowledges support from "Plan Nacional I+D+I" Grant PID2021-127842NB-I00.

J.A. Zamudio also acknowledges support the Grant "INVESTIGO" of the "Universidad Pablo de Olavide para la generación de empleo en el marco del Plan de Recuperación, Transformación *y* Resiliencia para Andalucía".

Funding for open access publishing: Universidad Pablo de Olavide

References

[1] P. Sun, X. Ma, Dominating communities for hierarchical control of complex networks, Inform. Sci. 414 (2017) 247–259.
[2] M.T. Ding-Zhu Du, Y.L.D.L. Thai, S. Zhu, Strongly connected dominating sets in wireless sensor networks with unidirectional links, in: Proceedings of AP-Web, in: LNCS, vol. 3841, 2016, pp. 13–24.
[3] J.C. Nacher, T. Akutsu, Minimum dominating set-based methods for analyzing biological networks, Methods 102 (2016) 57–63.
[4] J.N. Hooker, R.S. Garfinkel, C.K. Chen, Finite dominating sets for network location problems, Oper. Res. 39 (1991) 100–118.
[5] E.J. Cockayne, S.T. Hedetniemi, Towards a theory of domination in graphs, Networks 7 (3) (1977) 247–261.
[6] T.W. Haynes, S. Hedetniemi, P.J. Slater, Domination in Graphs: Advanced Topics, Marcel Dekker, New York, 1998.
[7] T.W. Haynes, S. Hedetniemi, P.J. Slater, Fundamentals of Domination in Graphs, Marcel Dekker, New York, 1998.
[8] T.W. Haynes, S.T. Hedetniemi, M.A. Henning, Structures of Domination in Graphs, Springer Nature, 2021.
[9] M.A. Henning, A. Yeo, Total Domination in Graphs, Springer Monographs in Mathematics, 2013.
[10] E.J. Cockayne, R.M. Dawes, S.T. Hedetniemi, Total domination in graphs, Networks 10 (3) (1980) 211–219.
[11] M.A. Henning, Total domination in graphs, J. Graph Theory 35 (1) (2000) 21–45.
[12] O. Favaron, K -domination and k -independence in graphs, Ars Combin. 25 (1988) 159–167.
[13] S. Li, On connected k -domination numbers of graphs, Discrete Math. 274 (1–3) (2004) 303–310.
[14] A. Hansberg, L. Volkmann, Upper bounds on the k -domination number and the k -Roman domination number, Discrete Appl. Math. 157 (7) (2009) 1634–1639.
[15] S. Bermudo, J.C. Hernández-Gómez, J.M. Sigarreta, On the total *k*-domination in graphs, Discuss. Math. Graph Theory 38 (1) (2018) 301–317.
[16] S. Bermudo, A. Cabrera Martínez, F.A. Hernández Mira, J.M. Sigarreta, On the global total *k*-domination number of graphs, Discrete Appl. Math. 263 (2019) 42–50.
[17] W. McCuaig, B. Shepherd, Domination in graphs of minimum degree two, J. Graph Theory 13 (1989) 749–762.
[18] B. Reed, Paths, stars, and the number three, Combin. Probab. Comput. 5 (3) (1996) 277–295.
[19] A. Potluri, A. Singh, Two hybrid meta-heuristic approaches for minimum dominating set problem, 2011, pp. 97–104.
[20] A. Potluri, A. Singh, Hybrid metaheuristic algorithms for minimum weight dominating set, Appl. Soft Comput. 13 (2013) 76–88.
[21] D. Chalupa, An order-based algorithm for minimum dominating set with application in graph mining, Inform. Sci. 426 (2018) 101–116.
[22] A. Casado, S. Bermudo, A. López-Sánchez, J. Sánchez-Oro, An iterated greedy algorithm for finding the minimum dominating set in graphs, Math. Comput. Simulation 207 (2023) 41–58.
[23] E.J. Cockayne, B. Gamble, B. Shepherd, An upper bound for the k -domination number of a graph, J. Graph Theory 9 (4) (1985) 533–534.
[24] L. Volkmann, A bound on the k -domination number of a graph, Czechoslovak Math. J. 60 (1) (2010) 77–83.
[25] E. DeLaViña, W. Goddard, M.A. Henning, R. Pepper, E.R. Vaughan, Bounds on the k -domination number of a graph, Appl. Math. Lett. 24 (6) (2011) 996–998.
[26] S. Bermudo, R.A. Higuita, J. Rada, *k*-Domination and total *k*-domination numbers in catacondensed hexagonal systems, Math. Biosci. Eng. 19 (7) (2022) 7138–7155.
[27] M. Nguyen, M. Hà, D. Nguyen, T. Tran, Solving the k-dominating set problem on very large-scale networks, Comput. Soc. Netw. 7 (2020) 4.
[28] R. Ruiz, T. Stützle, A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem, European J. Oper. Res. 177 (3) (2007) 2033–2049.
[29] M. Lozano, E. Rodriguez-Tello, Population-based iterated greedy algorithm for the S-labeling problem, Comput. Oper. Res. 155 (2023) 106224.
[30] J. Sánchez-Oro, A. Duarte, Iterated Greedy algorithm for performing community detection in social networks, Future Gener. Comput. Syst. 88 (2018) 785–791.
[31] J.D. Quintana, R. Martin-Santamaria, J. Sanchez-Oro, A. Duarte, Solving the regenerator location problem with an iterated greedy approach, Appl. Soft Comput. 111 (2021) 107659.
[32] S. Pérez-Peló, J. Sánchez-Oro, A.D. López-Sánchez, A. Duarte, A multi-objective parallel iterated greedy for solving the p-center and p-dispersion problem, Electronics 8 (12) (2019).
[33] E. Álvarez-Miranda, M. Sinnl, Exact and heuristic algorithms for the weighted total domination problem, Comput. Oper. Res. 127 (2021) 105157.