Solving the Minimum Positive Influence Dominating Set Problem in Social Networks Using Metaheuristics

Iván Penedo[1], Isaac Lozano-Osorio[1(✉)], Jesús Sánchez-Oro[1], and Óscar Cordón[2]

[1] Department of Computer Science and Statistics, Universidad Rey Juan Carlos, Móstoles, Spain
{ivan.penedo,isaac.lozano,jesus.sanchezoro}@urjc.es
[2] Department of Computer Science and Artificial Intelligence, University of Granada, Granada, Spain
ocordon@decsai.ugr.es

Abstract. The rise of the internet and social networks has posed new challenges in studying people's behavior on these platforms. People tend to trust or align with a small group of users, leading to the development of viral marketing techniques to effectively propagate information about products or services. This has led to the definition of problems related to social influence maximization/minimization and dominance sets. The Minimum Positive Influence Dominating Sets (MPIDS) problem involves finding a minimum cardinality dominance set to influence an entire social network. For a vertex to be influenced, at least half of its neighbors must be in the dominance set. Considering that MPIDS is an $\mathcal{NP}$-hard problem, where exact approximations are impractical due to the size of social networks, this work proposes using Basic Variable Neighborhood Search (BVNS). Given an initial solution generated by a constructive method, this metaheuristic consists of two phases: a shaking and an improvement phase. In the shaking phase, the solution is modified by removing and reconstructing it using a randomized greedy approach. The improvement phase involves an innovative local search strategy based on generating holes, which removes the δ-neighborhood of a vertex to facilitate a greedy solution reconstruction.

Keywords: Social network influence · Metaheuristics · Basic Variable Neighborhood Search · Minimum Positive Influence Dominating Sets

1 Introduction

Social Influence Maximization (SIM) problems have been extensively studied in the context of viral marketing, where consumers sequentially influence their social relationships to purchase a product [4]. Some related work shows that interactions between users in a social network can be used to prevent the spread

S. Cavero et al. (Eds.): ICVNS 2025, LNCS 16256, pp. 1–14, 2026.
https://doi.org/10.1007/978-3-032-19582-1_1

of diseases [8], conduct viral marketing campaigns [11], show recommendations in e-learning software [17], study social relationships [7], and prevent tobacco or other substance abuse [25]. For these problems, we seek to identify those users who accelerate or reduce the diffusion of influence in the network [12,13,21].

The Minimum Positive Influence Dominating Set (MPIDS) seeks the smallest set of users to influence an entire social network. This problem was formally defined in [26] where it was shown to be an $\mathcal{NP}$-hard problem for general graphs. Also, in the scope of these problems, a social network is formally modeled as an undirected graph $G = (V, E)$. Each edge $(u, v) \in E$ has two ends u and v, indicating that these users are connected, hence one user has a relationship with another and can influence him/her in the represented social network. Furthermore, we define $N(v)$ as the set of vertices adjacent to v, i.e., $\{u \in V : (u, v) \in E\}$.

Given an undirected graph $G = (V, E)$, a Positive Influence Dominating Set (PIDS) tries to find a set of vertices $D \subseteq V$ such that at least half of the neighbors of any vertex lie in D, i.e., $|D \cap N(v)| \geq |N(v)|/2 \; \forall v \in V$. Specifically, the goal of the MPIDS problem is to obtain a dominant set of positive influence with minimum cardinality. Lin et al. [10] defined the following linear integer programming model 1.

$$\begin{aligned} \min & \sum_{i=1}^{|V|} x_i \\ \text{s.t.} & \sum_{v_j \in N(v_i)} x_j \geq \left\lceil \frac{|N(v_i)|}{2} \right\rceil \quad \forall i \in V \\ & x_i \in \{0, 1\} \qquad\qquad\qquad\quad \forall i \in V \end{aligned} \tag{1}$$

Binary variables $x_i \in \{0, 1\}$ are assigned to each vertex $i \in V$, where $x_i = 1$ indicates that vertex i is selected, while $x_i = 0$ implies the opposite. On the other hand, the main constraint forces any feasible solution to have the necessary vertices selected so that all vertices $v_i \in V$ have at least half of their neighbors selected in D.

The main drawback is the size of today's social networks, which makes mathematical models unable in some cases to even provide a feasible solution. For this reason, there are different greedy or heuristic approaches in the literature.

First, Wang et al. [26] proposed a greedy algorithm that, at each iteration, selected the vertex that could influence a larger number of vertices, with an algorithmic complexity of $O(|V|^3)$. Later, Raei et al. [20] defined the parameter *cover-degree* to prioritize certain vertices for the greedy criteria reducing the time complexity to $O(|V|^2)$. Subsequently, the Fast Greedy Algorithm (FGA) presented by Pan et al. [19] followed the same greedy strategy but propagating through the neighborhood of the last dominated vertex and prioritizing dominance over vertices that do not satisfy the constraint, reducing the complexity to $O(|V| \log |V| + |E|)$. The Improved Greedy Algorithm (IGA-PIDS) [3] is based on FGA and uses the parameter *need-degree* in addition to *cover-degree* to prioritize vertices. It also performs a pruning of the network at the beginning to obtain those vertices that must always be dominated, and a final sieve to not

dominate redundant vertices in the dominance set at the end of the algorithm execution. The latter proposal obtains higher quality solutions than FGA, while maintaining a time complexity of $O(|V| + |E|)$.

On the other hand, different metaheuristic algorithms for the problem can be found in the literature. One of the first proposals was the ILPMA memetic algorithm by Lin et al. [10], which presented the mathematical formulation of the problem, in addition to proposing a Tabu Search to perform local optimizations on the dominance set. The first variant of a Construct, Merge, Solve and Adapt (CMSA) algorithm, proposed by Akbay and Blum [1], generated several solutions, then mixed them and performed several iterations using auxiliary adaptive structures to vary the generation of dominance sets. These structures were replaced by a variable n_a denoting the number of solutions generated in each iteration, which was incremented upon finding a solution with the same value of the objective function and reset upon finding a better solution in a later self-adaptive variant of that algorithm [2]. Then, the Iterated Carousel Greedy (ICG) [23] algorithm replaced the Tabu Search of [10] with an iterative process of destruction and reconstruction of the dominance set. Finally, the FastPIDS algorithm of Sun et al. [24] proposes several reduction rules for dominance set generation, as well as reusing the *need-degree* function from the literature. Also, the algorithm combines a greedy construction using a hybrid criteria with a local search based on vertex swapping, combined with a *two-level satisfaction judgment* mechanism. This mechanism uses two propositions to learn which partial dominance sets are promising and which are not to add new vertices to this set using the hybrid heuristic. The FastPIDS algorithm is currently the best performing algorithm according to the literature, so we will use it to compare against our proposal.

This paper presents a heuristic approach to provide quality solutions to the MPIDS problem with reduced computational time. A Basic Variable Neighborhood Search (BVNS) algorithm [18] has been designed, proposing a novel improvement based on creating holes in the solution and then applying an intelligent reconstruction of the dominance set.

The article is organized as follows. The design of the proposal is carried out in Sect. 2, which also describes the constructive stage and the improvement phase using the local search process. Section 3 presents the results obtained in the different experiments performed. These will be compared with the results of the best existing algorithm for the MPIDS problem to make a fair comparison. Finally, Sect. 4 mentions the conclusions reached with the development of this work and future work.

2 Basic Variable Neighborhood Search

The Variable Neighborhood Search (VNS) metaheuristic was proposed by Mladenović and Hansen in 1997 [18] as a systematic approach to explore multiple neighborhoods in optimization problems. Its main contribution lies in two key ideas: considering diverse neighborhood structures during the search and systematically switching between them to escape local optima. The algorithm alternates

phases of exploration (shaking) and exploitation (local search), progressively expanding the search radius. Thanks to this strategy, VNS effectively balances intensification and diversification, allowing it to explore promising regions of the solution space efficiently.

The pseudocode corresponding to the BVNS metaheuristic is presented in the Algorithm 1 where increasing neighborhoods are reached through the Shake operator, ensuring systematic diversification [6]. It requires as input the network G, together with the parameters α, δ (refer to the descriptions below), the maximum execution time t_{max}, and the maximum number of neighborhoods k_{max}. The procedure starts with a trivial initial solution that contains all vertices V (line 1). Then, a main loop is executed as long as the time limit is not reached (line 2).

In each iteration, an initial solution is generated through the `Construct` procedure, which uses α as a randomness parameter (line 3). Subsequently, this solution is improved by means of a local search with depth vertex removal parameter δ (line 4) and the neighborhood index k is initialized (line 5). From this solution, the characteristic BVNS process is repeated until variable k reaches the value of parameter k_{max} (line 6): increasing neighborhoods are reached through the `Shake` operator, which perturbs the current solution according to the neighborhood index k (line 7). Then, local search is applied again to intensify the generated solution (line 8) and it is decided whether to advance to a more distant neighborhood or to restart from the first one (line 9).

This process allows us to escape from local optima in a systematic way by exploring different neighborhood structures. When the algorithm reaches $k > k_{\max}$, if the solution D obtained in the BVNS process is better than the best known solution D_b (line 11), D_b is updated with the new dominance set (line 12). Finally, after running out of execution time, the best solution found D_b is returned (line 15).

Algorithm 1. *BVNS* $(G, \alpha, \delta, t_{\max}, k_{\max})$

```
1: D_b ← V
2: while time ≤ t_max do
3:     D ← Construct(G, α)
4:     D ← LocalSearch(D, δ)
5:     k ← 1
6:     while k ≤ k_max do
7:         D' ← Shake(D, k, α)
8:         D' ← LocalSearch(D', δ)
9:         k ← NeighborhoodChange(D, D', k)
10:    end while
11:    if |D| < |D_b| then
12:        D_b ← D
13:    end if
14: end while
15: return D_b
```

Since the constructive process may cause the proposed solution to contain vertices that are not necessary to satisfy the constraints of the problem, a refinement process has been added. This process runs through the vertices added to the dominance set to check which ones can be removed while maintaining a feasible solution. For a vertex d to be considered redundant and therefore removed from the solution, the following constraint must be satisfied:

$$|N(v) \cap D| > \left\lceil \frac{N(v)}{2} \right\rceil \forall v \in N(d)$$

Therefore, the method will go through all the vertices belonging to the solution and eliminate those that meet the above restriction.

2.1 Constructive Procedure

The constructive phase seeks to obtain an initial solution starting from an empty solution and using a greedy criteria to select the candidate vertices to add to the dominance set.

The criteria followed in this work is based on the degree $|N(v)|$ of each of the vertices $v \in V$. In the constructive phase, it is necessary to evaluate the contribution of each of its vertices with a greedy criteria. Influence problems in social networks use datasets with millions of users, for that reason, a simple greedy construction is used for generating an initial solution before the BVNS procedure. Therefore, the constructive procedure iteratively adds the vertex with the largest degree to the incumbent solution until it becomes feasible.

Before this, it is important to note that some of the reductions proposed by Sun et al. [24] have been applied, which allow us to determine whether certain vertices should belong or not to the solution for different characteristics of the vertices shown below. The first reduction shows that, if a leaf vertex has only one parent vertex, in order to obtain a feasible solution it is necessary that the parent is always in the set D. Meanwhile, in case of having a vertex structure forming a triangle between the relations, it will be necessary that the solution D contains two of the three vertices, leaving the third out of it.

2.2 Improvement Process

The novel proposal in this work consists of a local search based on the generation of holes (see Algorithm 2) called Piercing Local Search (PLS). This method is proposed with the objective of eliminating vertices from a certain region of the solution. After performing the elimination, the solution will be reconstructed in an intelligent way to obtain better solutions.

The algorithm iterates until it finds no improvement in the entire solution set or reaches the runtime limit (lines 1 to 3). The iteration starts looking for some improvement for a vertex of the dominance set (line 4). To this new solution, the piercing process (line 5) is applied, which will be detailed later in Algorithm 3. Subsequently, the solution is reconstructed using the same greedy

criteria described in the constructive procedure (Sect. 2.1) (line 6). Finally, those redundant vertices are removed (line 7) as in Algorithm 1 and, if the obtained solution is better than the previous one (line 8), the best solution is updated (line 9), starting the search again (line 11). Otherwise, it continues searching for an improvement over the next vertex in the set D.

Algorithm 2. *Improve* (D, δ)

```
1: improve ← TRUE
2: while improve and time ≤ t_max do
3:    improve ← FALSE
4:    for all v ∈ D do
5:       D' ← Pierce(D, v, δ)
6:       D'' ← Reconstruct(D')
7:       D''' ← RemoveRedundant(D'')
8:       if |D'''| < |D| then
9:          D ← D'''
10:         improve ← TRUE
11:         go to 3
12:      else
13:         D' ← D
14:      end if
15:   end for
16: end while
17: return D
```

In Algorithm 3 the pseudocode of the piercing process is presented, which aims to pierce a dominance set D' by eliminating several vertices starting from an initial vertex v, avoiding to eliminate those previously fixed by the reduction rules. During this process, all vertices that are δ levels deep from vertex v in the network will be eliminated, the neighborhood with $\delta = 1$ being the set $\{v\}$. In this procedure, if the last expansion level has not been reached (line 1), the vertex v is removed (line 2) and for each of the adjacent vertices (line 3) the algorithm is called with a lower expansion level (line 4), since the level that would correspond to vertex v has been calculated in line 2. Finally, the pierced dominance set D is returned (line 7).

Algorithm 3. *Pierce* (D, v, δ)

```
1: if δ > 0 then
2:    D ← D \ {v}
3:    for all u ∈ N(v) do
4:       Pierce(D, u, δ − 1)
5:    end for
6: end if
7: return D
```

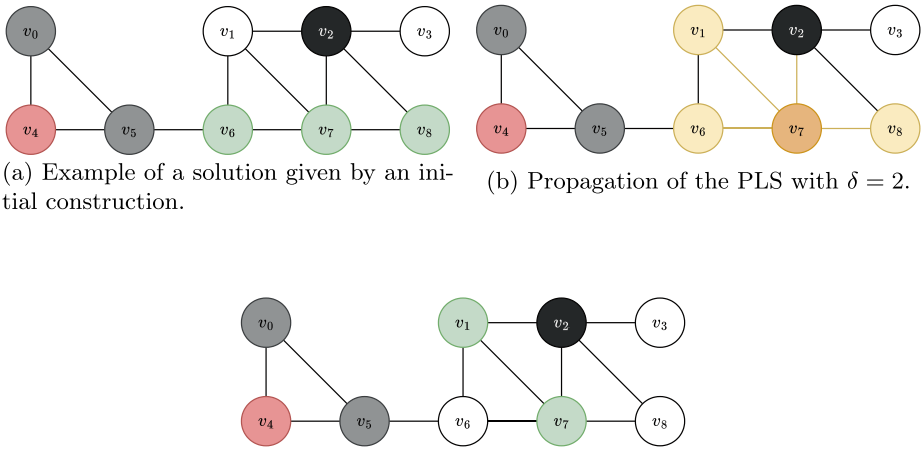
Starting from the incomplete solutions, the dominant set will be reconstructed by adding the necessary vertices to make it feasible using the greedy degree-based heuristic described in Sect. 2.1.

To graphically illustrate the method, Fig. 1 shows examples of the improvement phase of the implemented BVNS process. The different components of this figure show a social network represented as a graph with 9 vertices and 12 edges at different times of the piercing phase. These vertices are identified with different colors where each color represents a different state. The green ones are those included in the solution set by the constructive method, the black ones are those fixed by the reduction rule of the leaf vertices, and the gray ones by the reduction rule related to the triangles. Meanwhile, the white vertices are not in the solution and the red ones would be excluded by the triangle reduction rule [24]. Finally, orange indicates the initial vertex of the PLS propagation and yellow those affected by the piercing process.

The initial solution proposed by the constructive phase of Fig. 1a has an objective function $|D| = |\{v_0, v_2, v_5, v_6, v_7, v_8\}| = 6$. Applying PLS on vertex v_7, vertices $\{v_1, v_6, v_7, v_8\}$ are affected, which are removed from the solution set (Fig. 1b). It should be noted that vertex v_2 should also be affected, but it is not as it is marked as fixed by the reduction of the leaf vertices. After subsequent greedy reconstruction of the solution set (Fig. 1c), a new solution is obtained such that $|D| = |\{v_0, v_1, v_2, v_5, v_7\}| = 5$, which would be a solution of minimum cardinality for the MPIDS problem.

2.3 Shake Procedure

To enhance the efficiency and exploration of the method of the method, the shaking phase is employed, drawing inspiration from the principles of VNS [18].



(a) Example of a solution given by an initial construction.

(b) Propagation of the PLS with $\delta = 2$.

(c) MPIDS solution after PLS.

Fig. 1. Instance composed of 9 vertices and 12 relations, before (a), during (b) and after (c) of the PLS process.

In this context, the shaking phase involves generating a solution D' from a predefined neighborhood $N_k(D)$ of the current solution D. The purpose of this generation is to effectively perturb the solution, thereby enabling the search to escape local optima and explore different regions of the solution space, preventing cycles that could arise from deterministic rules [18]. In this specific implementation, the parameter k defines the number of nodes removed from the current dominating set solution. This process moves the solution to a perturbed state D' from which a new local search can begin. Following this removal, the solution is then intelligently reconstructed.

This reconstructive method will be referred to as the Randomized Greedy Algorithm (RGA). Specifically, the RGA will use the α parameter to determine a subset of vertices, selecting among them the candidate with the highest degree. Here, the α parameter controls the degree of randomness of the method. Specifically, a value of $\alpha = 1$ would denote a fully greedy criterion (since it would evaluate all candidates in the *RCL*), while a value of $\alpha = 0$ would define a fully random one (since it would only evaluate one candidate at random).

First, we generate a list of vertices *NF* formed by those vertices that do not yet fulfill the feasibility function such that

$$NF = \left\{ v \in V : |N(v) \cap D| < \left\lceil \frac{|N(v)|}{2} \right\rceil \right\}$$

It is important to note that, for a solution to be feasible, it is necessary to achieve that $NF = \emptyset$, which defines the stopping criteria of the constructive method.

Next, a random element v is chosen from *NF*, and the candidate list *CL* that can become part of the solution, formed by $N(v) \setminus D$, is generated, with D being the solution under construction.

Once the *CL* is available, the restricted candidate list (*RCL*) is created. While in the traditional scheme this list includes the most promising candidates, in the proposal of this work it uses a fraction α of candidates randomly obtained from the *CL*, which are the only ones evaluated afterwards. The reason for this modification is that, given the size of the social networks, the evaluation of all candidates from the *CL* would require a high computational time. The next vertex to be added to the solution will be the one with the highest degree, i.e., $max_{c \in RCL}|N(c)|$. Candidates will continue to be added until the original vertex v meets the feasibility constraints, at which point it will be removed from *NF* and a new candidate will be chosen.

3 Experimental Results

This section discusses the results obtained in the various experiments carried out. The designed algorithm has been executed on 196 instances collected from the literature, obtained from public repositories such as *Stanford Network Analysis Project* (SNAP) [9] and *Network Repository* [22].

Table 1. Instance characteristics

Metric	Total	Average	Min.	Max.
Size (B)	33 304 329 818	170 791 434	492	4 311 190 746
# Vertices	415 204 269	2 129 253	34	59 216 211
# Edges	2 100 272 636	10 770 628	78	261 321 071

Table 1 lists the main characteristics of these instances. A statistical summary of three key metrics is included: size in bytes, number of vertices and number of edges, along with their total, average, minimum and maximum values. Overall, more than 415 million vertices are considered, where some networks are extremely small (34 vertices), while the largest exceeds 59 million vertices. Meanwhile, more than 2.1 billion edges are recorded, where the smallest network contains 78 connections, while the largest one exceeds 261 million edges.

The designed experimentation consists of two phases and has been performed on a server virtual machine with an AMD EPYC 7282 (2.8 GHz) processor using a single core, 128 GB of RAM, Ubuntu Server 20.04, and Java 21. The first phase includes a preliminary experiment performed on a diverse subset of instances (Sect. 3.1), whose purpose is to define the best parameter values for the algorithm. To avoid overfitting the parameter settings [5], this subset is obtained from the selection of 25% of the available instances. The second phase consists of a final experimentation to validate the best configuration against the state of the art (Sect. 3.2).

All experiments have been evaluated using the same metrics, including: *Average*, which represents the mean of the objective function value of all instances for an algorithm; *Time (s)*, which indicates the average execution time in seconds; *Dev. (%)*, which shows the mean of the percentage deviation of the objective function versus the best known value of the experiment for each instance; and # *Best*, which indicates the number of best solutions found in the experiment. Also, the best of the values in each of these metrics in each experiment has been indicated in bold in each of the tables in this section.

It should be noted that, in the literature, authors establish a stopping criteria of one hour to obtain a solution for some of the instances and 1000 s for others. For this reason, in the experimentation carried out, the execution time of the proposed algorithms has been limited to one hour.

3.1 Preliminary Experiments

Table 2 allows us to analyze the contribution of the different approaches studied: i) a single random construction of the solution (*Random*); ii) multiple random runs limiting the execution time to one hour (*Random (3600 s)*); iii) a single greedy construction (*Greedy*); and iv) a greedy construction with redundant vertex elimination (*Greedy**).

Table 2. Comparison of random and greedy constructions.

Algorithm	Average	Time (s)	Dev. (%)	# Bests
Random	776363.86	3.13	23.23	0
Random (3600 s)	776157.63	3600.00	19.02	3
Greedy	650048.98	**2.74**	20.44	1
*Greedy**	**621659.84**	2.87	**0.42**	**46**

It can be seen how the greedy criteria used is suitable for this problem, since it is able to obtain one of the lowest average values of the objective function and a low deviation in a very short time, compared to the randomized algorithm that uses more than 3000 s. The deviation of this run is particularly noteworthy, because even obtaining a reduction of more than 100000 vertices in the Average metric, it obtains a slightly higher value than the algorithm *Random(3600 s)*. This is due to the large difference in the sizes of the instances, since a difference of 3 or 5 vertices shows between 20% and 30% deviation from the best, while to reach that deviation in large instances there must be a difference of 0.5 million vertices. Since *Greedy* performs better on large instances and worse on small ones, it is penalized with a higher deviation even though it has reduced the Average.

On the other hand, the *Greedy** algorithm with redundant vertices elimination shows even better results than the *Greedy* algorithm. With a slight increase in execution time, it obtains the best value in 46 out of the total 49 instances, leaving the rest of the proposals behind in terms of objective function value and deviation. This shows that the process of eliminating redundant vertices contributes positively in the proposal. Consequently, the *Greedy** method is chosen to validate the δ size of holes within the PLS.

In order to obtain the best configuration of the PLS, two runs have been performed with different values of the δ parameter, collected in Table 3. These values have been 2 and 3, since $\delta = 1$ would only eliminate itself and values of $\delta > 3$ could eliminate the solution almost completely as a consequence of the small world property of networks.

Table 3. Comparison of different values for the δ parameter in the piercing phase.

δ	Average	Time (s)	Dev. (%)	# Bests
1	621659.84	**2.87**	4.25	3
2	565363.71	252.13	**0.13**	**40**
3	**565325.10**	675.54	0.54	13

With this experimentation, we can observe that, although with a value $\delta = 3$ we obtain a slight improvement in terms of the average value of the objective

function, the results with $\delta = 2$ have a smaller deviation and a higher number of better solutions with a notably lower time. This is important since, for large instances, the shorter the time required for local search, the more iterations of BVNS can be performed, allowing better solutions to be found.

In order to find the best configuration of the parameter α in the RGA construct, five runs have been performed with $\delta = 2$ and $\alpha = \{0.00, 0.25, 0.50, 0.75, 1.00\}$ plus a sixth run $\alpha = RND$ where a uniform random value in $\alpha = [0.00, 1.00]$ is generated for each new construct. The execution time considered remains one hour for all configurations except for $\alpha = 0.00$ which, being a greedy construct, always starts from the same initial solution and one execution per instance is required. The results obtained are shown in Table 4.

As can be seen, the value $\alpha = 0.00$ is the one that has obtained the best average, with a shorter time, since its generative criteria is totally greedy. Alternatively, the execution with $\alpha = 1.00$ has obtained a higher quality in terms of better results per number of instances, but obtaining a high average value compared to the rest of the results, which suggests a good performance in small instances, but a weakness with large instances.

Table 4. Comparison for the best configuration of the α parameter in the shake phase for reconstruction.

α	Average	Time (s)	Dev. (%)	# Bests
RND	593147.63	3600.23	0.44	5
0.00	**565363.71**	**252.13**	1.30	6
0.25	615990.47	3600.16	0.51	12
0.50	593060.12	3600.17	0.35	15
0.75	579615.39	3600.22	**0.25**	14
1.00	643084.12	3600.26	0.63	**24**

It has been decided to use the $\alpha = 0.75$ configuration since, in addition to obtaining a competent quality with respect to the two previous α values mentioned above, the best deviation with 0.25% and the second best value in terms of average objective function, it provides a large variability in the construction phase.

As for the $k_{\max}$ value, we used $k_{\max} = 1$, where 1 is the number of nodes, since higher $k_{\max}$ was time-consuming and results in worse results.

3.2 Comparison with the State of the Art

Once the best configuration of the implementation was obtained, a final experimentation has been performed with a greedy initial solution construction, a randomized greedy reconstruction with $\alpha = 0.75$, a redundant vertex elimination, and the PLS with two levels of expansion $\delta = 2$. In order to make a comparison with the best state-of-the-art results obtained by the FastPIDS algorithm

designed by Sun et al. [24], the execution time has been limited again to one hour. Note that, compared to the previous study, we have just performed one run instead of using the average of 10 runs. Table 5 collects the results of this comparison.

Table 5. Comparison of the state-of-the-art algorithm and our proposal.

Algorithm	Average	Time (s)	Dev. (%)	# Bests
FastPIDS	**1023870.82**	**3600.00**	**0.01**	**190**
BVNS	1038446.79	3600.03	1.26	18

The evaluation performed reflects that the previous algorithm obtains better results in all the metrics considered, with a lower average value and achieving the best results over 190 instances of the total of 196 instances evaluated. However, the proposed metaheuristic solution obtains reasonable results, with a deviation close to 1.26% and achieving better average in 6 of the instances. The obtained results show how a metaheuristic design based on the combination of several simple methods can show a competitive performance. The next section shows some conclusions and future work described that raise promising ideas for further improvement of these approaches.

4 Conclusion and Future Work

In this work, a BVNS approach has been proposed for the MPIDS problem in social networks. An efficient construction phase has been designed that provides a great variability in the process of obtaining initial solutions on which, in a later improvement stage, it is possible to reduce considerably the size of the dominance set. On the other hand, the proposed PLS generates local optima from the given solutions in a reduced computation time. This constructive and improvement phase, combined with the solution permutation in the BVNS metaheuristic, offers competitive results with respect to FastPIDS.

Some future work derived from this research is to create new intelligent constructive movements [14], as well as studying some alternative greedy criteria to obtain higher quality solutions in a reduced computation time. Finally, we plan to perform a factorial experimentation using tools such as iRACE [15] to obtain the best configuration of the algorithm, selecting the preliminary instances more carefully [16].

Acknowledgments. The authors appreciate the support of the Comunidad Autónoma de Madrid (grant ref. TEC-2024/COM-404), Ministerio de Economía y Competitividad (grant ref. PID2021-125709OA-C22), and Ministerio para la Transformación Digital y de la Función Pública (Cátedra ENIA AI4DDS, grant ref. TSI-100930-2023-3).

References

1. Akbay, M.A., Blum, C.: Application of CMSA to the minimum positive influence dominating set problem. In: Artificial Intelligence Research and Development, pp. 17–26. IOS Press, October 2021. https://doi.org/10.3233/faia210112
2. Akbay, M.A., López Serrano, A., Blum, C.: A self-adaptive variant of CMSA: application to the minimum positive influence dominating set problem. Int. J. Comput. Intell. Syst. **15**(1) (2022). https://doi.org/10.1007/s44196-022-00098-1
3. Bouamama, S., Blum, C.: An improved greedy heuristic for the minimum positive influence dominating set problem in social networks. Algorithms **14**(3), 79 (2021). https://doi.org/10.3390/a14030079
4. Domingos, P., Richardson, M.: Mining the network value of customers. In: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '01, pp. 57–66. Association for Computing Machinery, New York (2001). https://doi.org/10.1145/502512.502525
5. Heitsch, H., Römisch, W.: Scenario reduction algorithms in stochastic programming. Comput. Optim. Appl. **24**(2), 187–206 (2003). https://doi.org/10.1023/A:1021805924152
6. Herrán, A., Colmenar, J.M., Duarte, A.: A variable neighborhood search approach for the vertex bisection problem. Inf. Sci. **476**, 1–18 (2019). https://doi.org/10.1016/j.ins.2018.09.063
7. King, S.F., Burgess, T.F.: Understanding success and failure in customer relationship management. Ind. Mark. Manage. **37**(4), 421–431 (2008). https://doi.org/10.1016/j.indmarman.2007.02.005
8. Klovdahl, A.S.: Social networks and the spread of infectious diseases: the aids example. Soc. Sci. Med. **21**(11), 1203–1216 (1985). https://doi.org/10.1016/0277-9536(85)90269-2
9. Leskovec, J., Krevl, A.: SNAP Datasets: Stanford large network dataset collection, June 2014. http://snap.stanford.edu/data
10. Lin, G., Guan, J., Feng, H.: An ILP based memetic algorithm for finding minimum positive influence dominating sets in social networks. Physica A Stat. Mech. Appl. **500**, 199–209 (2018). https://doi.org/10.1016/j.physa.2018.02.119
11. Long, C., Wong, R.C.W.: Viral marketing for dedicated customers. Inf. Syst. **46**, 1–23 (2014). https://doi.org/10.1016/j.is.2014.05.003
12. Lozano-Osorio, I., Sánchez-Oro, J., Duarte, A.: An efficient and effective grasp algorithm for the budget influence maximization problem. J. Ambient. Intell. Humaniz. Comput. **15**(3), 2023–2034 (2023). https://doi.org/10.1007/s12652-023-04680-z
13. Lozano-Osorio, I., Sánchez-Oro, J., Duarte, A., Cordón, O.: A quick GRASP-based method for influence maximization in social networks. J. Ambient. Intell. Humaniz. Comput. **14**(4), 3767–3779 (2021). https://doi.org/10.1007/s12652-021-03510-4
14. Lozano-Osorio, I., Sánchez-Oro, J., Sörensen, K.: Determining good solutions and validating them with a metaheuristic approach in social network influence minimization problems. Eur. J. Oper. Res. (2025). https://doi.org/10.1016/j.ejor.2025.11.024
15. López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Birattari, M., Stützle, T.: The irace package: iterated racing for automatic algorithm configuration. Oper. Res. Perspect. **3**, 43–58 (2016). https://doi.org/10.1016/j.orp.2016.09.002
16. Martín-Santamaría, R., Cavero, S., Herrán, A., Duarte, A., Colmenar, J.M.: A practical methodology for reproducible experimentation: an application to the double-row facility layout problem. Evol. Comput. **32**(1), 69–104 (2024). https://doi.org/10.1162/evco_a_00317

17. Miller, R., Lammas, N.: Social media and its implications for viral marketing. Asia Pacific Pub. Relat. J. **11** (2010)
18. Mladenović, N., Hansen, P.: Variable neighborhood search. Comput. Oper. Res. **24**(11), 1097–1100 (1997). https://doi.org/10.1016/s0305-0548(97)00031-2
19. Pan, J., Bu, T.M.: A fast greedy algorithm for finding minimum positive influence dominating sets in social networks. In: IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), IEEE INFOCOM 2019, April 2019, pp. 360–364. IEEE (2019). https://doi.org/10.1109/infcomw.2019.8845129
20. Raei, H., Yazdani, N., Asadpour, M.: A new algorithm for positive influence dominating set in social networks. In: 2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, August 2012, pp. 253–257. IEEE (2012). https://doi.org/10.1109/asonam.2012.51
21. Robles, J.F., Chica, M., Cordon, O.: Evolutionary multiobjective optimization to target social network influentials in viral marketing. Exp. Syst. Appl. **147**, 113183 (2020). https://doi.org/10.1016/j.eswa.2020.113183
22. Rossi, R.A., Ahmed, N.K.: The network data repository with interactive graph analytics and visualization. In: AAAI (2015). https://networkrepository.com
23. Shan, Y., Kang, Q., Xiao, R., Chen, Y., Kang, Y.: An iterated carousel greedy algorithm for finding minimum positive influence dominating sets in social networks. IEEE Trans. Comput. Soc. Syst. **9**(3), 830–838 (2022). https://doi.org/10.1109/tcss.2021.3096247
24. Sun, R., Wu, J., Jin, C., Wang, Y., Zhou, W., Yin, M.: An efficient local search algorithm for minimum positive influence dominating set problem. Comput. Oper. Res. **154**, 106197 (2023). https://doi.org/10.1016/j.cor.2023.106197
25. Wang, F., Camacho, E., Xu, K.: Positive influence dominating set in online social networks. In: Du, D.-Z., Hu, X., Pardalos, P.M. (eds.) COCOA 2009. LNCS, vol. 5573, pp. 313–321. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02026-1_29
26. Wang, F., et al.: On positive influence dominating sets in social networks. Theoret. Comput. Sci. **412**(3), 265–269 (2011). https://doi.org/10.1016/j.tcs.2009.10.001