Knowledge-Based Systems 278 (2023) 110827

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

Dynamic Path Relinking for the Target Set Selection problem

Isaac Lozano-Osorio, Andrea Oliva-García, Jesús Sánchez-Oro *

Department of Computer Science and Statistics, University Rey Juan Carlos, C/Tulipán, S/N, Móstoles, 28933, Spain

ARTICLE INFO

Article history:
Received 12 April 2023
Received in revised form 18 July 2023
Accepted 20 July 2023
Available online 1 August 2023

Dataset link: https://grafo.etsii.urjc.es/TSS/

Keywords:
Target Set Selection problem
Influence maximization
Dynamic Path Relinking
GRASP
Social networks
Metaheuristics

ABSTRACT

This research proposes the use of metaheuristics for solving the Target Set Selection (TSS) problem. This problem emerges in the context of influence maximization problems, in which the objective is to maximize the number of active users when spreading information throughout a social network. Among all the influence maximization variants, TSS introduces the concept of reward of each user, which is the benefit associated to its activation. Therefore, the problem tries to maximize the reward obtained among all active users by selecting an initial set of users. Each user has also associated an activation cost, and the total sum of activation costs of the initial set of selected users cannot exceed a certain budget. In particular, two Path Relinking approaches are proposed, comparing them with the best method found in the state of the art. Additionally, a more challenging set of instances are derived from real-life social networks, where the best previous method is not able to find a feasible solution. The experimental results show the efficiency and efficacy of the proposal, supported by non-parametric statistical tests.

1. Introduction

The evolution of Social Networks has been in continuous growth in the last years. Nowadays, almost everyone frequently uses one or more Social Networks for both posting and gathering information. Due to the relevance of Social Networks in several contexts, such as politics, marketing, or disease control, among others, scientists and practitioners have put all their efforts in designing and developing algorithms to automatically analyze and collect the most relevant information from them.

Among all the different problems that emerge from Social Network Analysis, this research is focused on Influence Maximization Problems. This is a large family of hard combinatorial optimization problems where it is necessary to select a set of users from a Social Network with the aim of maximizing the spread of information throughout the network. In particular, this study is focused on the Target Set Selection Problem in which, given a certain budget, it is necessary to select a subset of users whose total cost is smaller than the given budget, with the aim of maximizing the reach of information dissemination through the network.

In the context of Target Set Selection Problem (TSS), there are two main variants: guaranteeing reaching the complete network (or even a certain part of it) with the minimum number of initial users or maximizing the number of users reached while not exceeding an initial budget. This proposal is focused on solving the latter, which is usually named Max-TSS.

Although the main application of this problem is to increase the impact of advertising a product for a company [1,2], there are several applications in different fields. For instance, in politics, the Max-TSS can be used for reducing the impact of fake news or misinformation from two opposite approaches: identifying the individuals which are mostly spreading fake news through the network, or to boost those individuals which are transmitting reliable information [3]. Even more, this application is closely related to disease control, since it has been proven that the diseases spreads following the same model as information through Social Networks [4].

In this work, a metaheuristic algorithm based on a combination of Greedy Randomized Adaptive Search Procedure and Path Relinking is presented with the aim of providing high-quality solutions for the Max-TSS in reasonable computing times when considering large real-life Social Networks. Metaheuristics were originally defined by [5] as "a high-level problem-independent algorithmic framework that provides a set of guidelines or strategies to develop heuristic optimization algorithms". In the last decades, metaheuristics have become one of the most extended approximate type of algorithms for solving hard and complex combinatorial optimization problems. These are robust algorithms which are able to provide high-quality

* Corresponding author.
E-mail addresses: isaac.lozano@urjc.es (I. Lozano-Osorio), andrea.oliva@urjc.es (A. Oliva-García), jesus.sanchezoro@urjc.es (J. Sánchez-Oro).
URLs: https://grafo.etsii.urjc.es/en/author/isaac-lozano-osorio/ (I. Lozano-Osorio), https://grafo.etsii.urjc.es/en/author/andrea-oliva-garcia/ (A. Oliva-García), https://jesussanchezoro.github.io/ (J. Sánchez-Oro).

https://doi.org/10.1016/j.knosys.2023.110827

solutions in reasonable computing times. However, they cannot guarantee the optimality of the solution. Therefore, they are recommended when it is not possible to compute the optimal solution with an exact algorithm. This algorithm is tested with large-scale graphs derived from real social networks, analyzing the contribution of each part of the algorithm. This analysis will allow researchers and practitioners to select and adapt some parts of the proposed algorithm to other related problems. One of the main issues when dealing with Target Set Selection consists of getting stuck in local optima. The algorithm based on Dynamic Path Relinking proposed in this research is able to escape from local optima by creating a path between two high-quality solutions, which will lead the algorithm to explore a wider portion of the search space. In particular, Dynamic Path Relinking will be able to identify the most promising subsets of influential users and strategically combine them in a path to generate new diverse and high-quality solutions.

The main contributions of this research can be summed up as follows:

- A constructive procedure based on a novel heuristic is proposed, being able to generate feasible solutions in small computing times.
- The proposed local search is optimized by reducing the evaluation of the objective function, which is the most computationally demanding part of the algorithm.
- Dynamic Path Relinking is proposed with the aim of creating paths between high-quality solutions and compared with the classical Static Path Relinking, Simulated Annealing and Cost-Effective Forward selection.
- The dataset of instances has been extended with real-life networks where the previous methods are not able to provide a solution in reasonable computing times.

Rest of the manuscript is structured as follows. Section 2 shows the related work with Target Set Selection Problem; Section 3 formally defines the Target Set Selection Problem; Section 4 thoroughly describes the algorithmic proposal and all the components designed; Section 5 presents a detailed computational experimentation to evaluate the contribution of each component and test the proposal with the best previous algorithm; and, finally, Section 6 draws some conclusions derived from the research, as well as highlights some future research lines.

2. Related work

In this section, we introduce some related work about TSS as well as a brief survey of existing methods for solving this problem, either based on exact methods, approximation methods, heuristics or computational intelligence algorithms.

Richardson et al. [6] initially formulated the problem of selecting target nodes in SNs. The TSS was originally proposed in [7], where it was proven to be $\mathcal{NP}$-hard, and the authors proposed a polynomial-time approximation algorithm for a probabilistic variant of the problem.

Kempe et al. [7] proposed three influence diffusion models that play an important role in understanding the diffusion phenomenon: the independent cascade model (ICM), the weighted cascade model (WCM), and the linear threshold model (LTM). In subsequent researches, several proofs of $\mathcal{NP}$-hardness were proposed [4,8–11], all of them supported by approximation results for specific types of network topologies [12,13].

The research on Social Network Influence problems has been focused on finding exact methods under restricted conditions. In particular, the TSS has been tackled from both exact [14–17], and heuristic perspectives [18,19].

The fact that the TSS problem when considering LTM as a diffusion model can be described as a hard combinatorial optimization problem has attracted the attention of both academic and practitioners. In fact, the problem can be stated as an Integer Linear Programming (ILP) problem, which is able to solve small problem instances. Two models have been proposed: a time-dependent ILP [20], which derives instantly from the definition, and a time-independent one [14].

While the previous results are certainly of interest, allowing researchers to extract information about graph properties which are a key part of the complexity of the problem, all of them deal with exactly solving the problem. This also holds for variations of the problem, such as the latency-bounded TSS (which aims to activate all the nodes of a graph in a bounded number of rounds), for which recent research is focused on exact methods for specific cases or making use of certain properties [21–23].

Additionally, some variants with specific constraints derived from real applications have also been considered, tackled with evolutionary metaheuristics [24,25]. Swarm intelligence algorithms have been applied to a multi-objective problem dealing with maximizing the spread of influence of a set while minimizing its size [26]. An evolutionary algorithm (EA) was applied in [19] to a variant of the TSS problem that was tackled in this research.

Ravelo et al. [24] proposed a new TSS variant denoted as the maximum effortreward GAP Target Set Selection problem (Max-TSS), a new $\mathcal{NP}$-hard version. To the best of our knowledge [27,28], the best approach for solving the Max-TSS is based on binary linear models and Lagrangian relaxations, which are solved by dynamic programming algorithms [19]. With the aim of improving the bounds, authors embed the dynamic programming algorithms in subgradient methods, which are used to generate feasible solutions for the problem. In the research, the authors highlight that their heuristic is able to reach near-optimal solutions in almost all the considered instances. Although the approach is able to find optimal solutions, its main drawback relies on the size of real-life networks. In the original work, authors optimally solved instances up to 58 nodes, while current social networks usually have more than 1000 nodes. With the aim of evaluating the limits of the exact proposal, this work tests the previous exact approach with graphs derived from real social networks such as Twitch or LastFM, among others.

Several works stated that metaheuristics scarce in the Social Network Influence Problems [25,27]. The use of Path Relinking in Graph Theory and Network Science has led to several successful research in the last years [29–31].

3. Problem definition

As it was aforementioned, the objective of the Target Set Selection Problem (TSS) is to find the most influential nodes in a social network. Let us define a social network as a graph $G = (V, E)$ where the set of vertices V, with $|V| = n$, represents the users of the social network and the set of edges E, with $|E| = m$, is conformed with pairs (u, v), with $u, v \in V$, indicating that there is any kind of relation between users u and v. Then, given a social network modeled as a graph $G = (V, E)$ and a maximum budget K, the TSS problem tries to find a subset of users $S \subseteq V$ whose effort does not exceed the maximum value K, with the aim of maximizing the number of influenced users in the social network. Since the concept of influencing can be ambiguous, it is necessary to perform some initial definitions to clarify it.

First of all, it is necessary to define how a node can potentially influence another one. In the context of TSS, the influence of a user follows a rational influence function $\psi : V \times V \rightarrow [0, 1]$ over every pair of users. This function measures the influence of

a user over another one. Notice that if two users u, v are not connected, then $\psi(v, u) = 0$. Then, a set of activated users S^t influences a non-activated user u if and only if the sum of the potential influence of all users in S^t is larger than or equal to 1, i.e., $\sum_{v \in S^t} \psi(v, u) \geq 1$. This rational influence function was given by several authors [8–10,19] and it tries to model a behavior in which if several users simultaneously influence a certain user, then it will be activated. The specific value of the function is determined by [19], and it is based on the number of interactions among users, i.e., the larger the number of interactions, the larger the function value. We denote S^t as the set of activated users in a certain iteration t. This model suggest a different approach to the well-known Influence Diffusion Models which requires from smaller computing times than probabilistic methods which are rather time consuming to obtain robust solutions.

Having defined the process of influencing a node given a set of activated nodes, it is necessary to define the influence propagation process. Starting from a set of initially activated nodes S, this process consists of activating all those non-activated nodes which are influenced by the activated ones. Without loss of generality, a solution for the TSS is given by the set of initially activated nodes S. Notice that this process is iteratively applied until no new nodes are activated. Given a step t and a set of activated nodes S^t, the set of nodes which are activated after applying a single iteration of the influence propagation process is denoted as S^{t+1}. The influence propagation process then stops when no new nodes are activated, i.e., $S^t = S^{t-1}$.

In TSS, each node has an associated effort to activate it, which is defined by the function $\alpha : V \rightarrow \mathbb{Z}^+$, as well as a reward obtained when it is influenced or initially activated which is defined by $\beta : V \rightarrow \mathbb{Z}^+$. Thus, the effort α is only considered for those nodes which are part of the set of the initially activated users S, and the reward β is earned whether a user u is initially activated or subsequently activated by the set of activated users. Another fact in TSS is that, when a node is activated at step t, it remains activated through the entire influence propagation process. Let us denote x_v^t as a binary variable which takes the value 1 if the node v is activated at step t and 0 otherwise (we refer the reader to [19] to a more detailed description of the model). Then,

$$x_v^{t-1} \leq x_v^t, \quad \forall v \in V, 1 \leq t \leq T$$

where T indicates the maximum number of steps in the influence propagation process, i.e., number of iterations in which new nodes are activated.

A solution S for the TSS is feasible if and only if the sum of the efforts of the initially activated nodes is smaller than or equal to K, which is a constraint of the problem, i.e., $\sum_{v \in V} \alpha(v) \cdot x_v^0 \leq K$. The objective function for the TSS is then evaluated as the sum of rewards obtained by the active nodes in the last iteration of the influence propagation process. In mathematical terms,

$$TSS(S) = \sum_{v \in V} \beta(v) \cdot x_v^T$$

Notice that evaluating the TSS is a computationally demanding procedure. In particular, the computational complexity of this evaluation is $O\left(n^2\right)$ since it is necessary to traverse all the nodes and, for each new activated node, it is required to traverse again the set of nodes searching for new potential nodes to be activated. The TSS then seeks for a solution $S^\star$ with the maximum objective function value. More formally,

$$S^\star \leftarrow \arg\max_{S \in \mathbb{SS}} TSS(S)$$

where $\mathbb{SS}$ represents the set of all feasible solutions, i.e., all possible combination of nodes whose sum of effort is smaller than or equal to K.

Fig. 1 depicts an example of the complete influence propagation process over a network with 5 nodes and 10 edges. The solution under evaluation is conformed by nodes C and E, i.e., $S = \{\mathtt{C}, \mathtt{E}\}$. Without loss of generality, let us suppose that the sum of costs $\alpha(\mathtt{C}) + \alpha(\mathtt{E})$ is smaller than or equal to K.

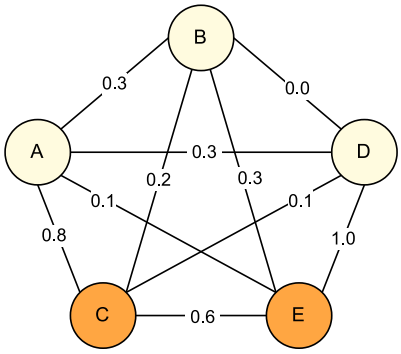
Fig. 1(a) shows the initial step $t = 0$, where the activated nodes are the ones initially selected C and E, i.e., $S = \{\mathtt{C}, \mathtt{E}\}$. Then, in the first iteration of the influence propagation process, depicted in Fig. 1(b), it is evaluated whether non-influenced nodes A, B, and D are influenced or not. Starting with node A, it is necessary to evaluate $\psi(\mathtt{C}, \mathtt{A}) + \psi(\mathtt{E}, \mathtt{A}) = 0.8 + 0.1 = 0.9 < 1.0$. Therefore, node A is not influenced in this step. A similar evaluation is performed with node B, resulting in $\psi(\mathtt{C}, \mathtt{B}) + \psi(\mathtt{E}, \mathtt{B}) = 0.2 + 0.3 = 0.5 < 1.0$, indicating that node B is not activated. Finally, when performing the evaluation of node D, we obtain $\psi(\mathtt{C}, \mathtt{B}) + \psi(\mathtt{E}, \mathtt{B}) = 0.1 + 1.0 = 1.1 \geq 1.0$. Then, after the first iteration, node D is included in the set of activated nodes, resulting in $S^1 = \{\mathtt{C}, \mathtt{D}, \mathtt{E}\}$.

Since $S^0 \neq S^1$, it is necessary to continue with the influence propagation process. If we now evaluate the second iteration, Fig. 1(c), starting with node A, $\sum_{v \in S^1} \psi(v, \mathtt{A}) = 0.8 + 0.1 + 0.3 = 1.2 \geq 1.0$. Then, node A will be included in the set of activated nodes in the next iteration, S^2. In the case of node B, the evaluation is $\sum_{v \in S^1} \psi(v, \mathtt{B}) = 0.2 + 0.3 + 0.0 = 0.5 < 1.0$, so the node B is not activated. After this iteration, $S^2 = \{\mathtt{A}, \mathtt{C}, \mathtt{D}, \mathtt{E}\} \neq S^1$, so an additional iteration is required. Fig. 1(d) illustrates the last iteration of the influence propagation process. In this case, it is only required to evaluate node B, resulting in $\sum_{v \in S^2} \psi(v, \mathtt{B}) = 0.3 + 0.2 + 0.3 + 0.0 = 0.8 < 1.0$. Therefore, node B is not activated. Since no new nodes are included in the set of activated nodes, i.e., $S^2 = S^3$, the influence propagation process stops in this iteration, returning the reward associated to the activated nodes.

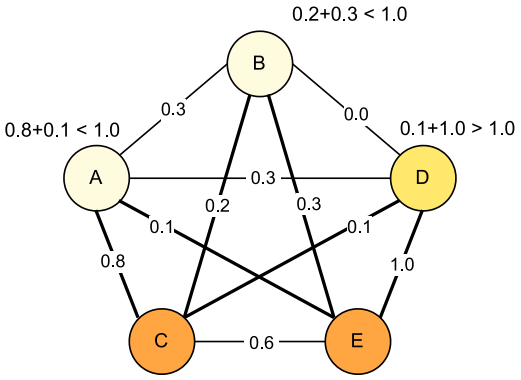
4. Algorithmic approach

This work presents an algorithm based on Path Relinking (PR) [32] for solving the TSS problem. Path Relinking was originally presented as a framework for combining intensification and diversification strategies in the context of Tabu Search [33]. PR relies on the idea of connecting two high-quality solutions creating a path between them, with the expectation of finding promising solutions during the exploration of the path. The algorithm tries to include in the first solution, usually named as initial solution, attributes of the second solution, named as guiding solution. Both the initial and the guiding solutions present a high quality and, therefore, it is expected that the path created between them explores new promising regions of the search space. It has been traditionally combined with Greedy Randomized Adaptive Search Procedure (GRASP) since Laguna and Marti [34] adapted PR to increase the intensification phase of GRASP. There are two main PR strategies extended in the literature: Static PR and Dynamic PR. In this work, both strategies are tested in the context of TSS. First of all, it is necessary to design a specific path-creation method between two solutions in the context of TSS. Given two solutions S_i and S_g, which are initial and guiding solutions, the objective is to create a path from S_i to S_g by removing attributes from the initial solution which are not present in the guiding one, and replacing them with attributes which are in the guiding solution but not in the initial one.

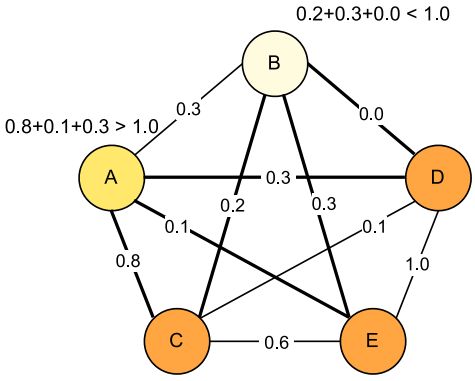
The path-creation method designed for the TSS problem iteratively removes nodes belonging to S_i but not to S_g, i.e., $S_i \setminus S_g$, and includes nodes which are in S_g but not in S_i, i.e., $S_g \setminus S_i$. The process of selecting the node to be removed and included in each iteration can be performed randomly (Random Path Relinking), greedily (Greedy Path Relinking), or in a more elaborated manner



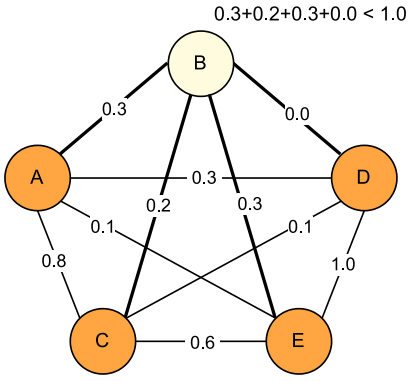
(a) Step I: original solution with $S = \{\mathtt{C}, \mathtt{E}\}$.



(b) Step II: $S^1 = \{\mathtt{C}, \mathtt{E}, \mathtt{D}\}$



(c) Step III: $S^2 = \{\mathtt{C}, \mathtt{E}, \mathtt{D}, \mathtt{A}\}$



(d) Step IV: $S^3 = \{\mathtt{C}, \mathtt{E}, \mathtt{D}, \mathtt{A}\}$

Fig. 1. Influence propagation process over a network with 5 nodes and 10 edges, considering the solution $S = \{\mathtt{C}, \mathtt{E}\}$.

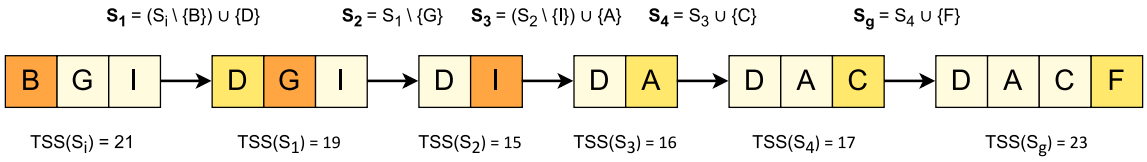


Fig. 2. Example of a path between solutions $S_i = \{\mathtt{B}, \mathtt{G}, \mathtt{I}\}$ and $S_g = \{\mathtt{D}, \mathtt{A}, \mathtt{C}, \mathtt{F}\}$.

(Greedy Randomized Path Relinking). Notice that both Greedy Path Relinking and Greedy Randomized Path Relinking are more computationally demanding than Random Path Relinking. This is mainly because they require to generate the complete set of feasible solutions in each step of the path, and also evaluate each one of them. Since the computational effort is a critical part of TSS, we have selected Random Path Relinking (RPR) which, additionally, increases the diversity of the search. Fig. 2 shows a possible path between the initial solution $S_i = \{\mathtt{B}, \mathtt{G}, \mathtt{I}\}$, with an objective function value of $TSS(S_i) = 21$, and the guiding solution $S_g = \{\mathtt{D}, \mathtt{A}, \mathtt{C}, \mathtt{F}\}$, with an objective function value of $TSS(S_g) = 23$.

The path starts by selecting the elements which need to be included during the path as $S_g \setminus S_i = \{\mathtt{D}, \mathtt{A}, \mathtt{C}, \mathtt{F}\}$, since S_i and S_g do not have any element in common. Then, it is required to select those elements which will be removed during the path creation, which are $S_i \setminus S_g = \{\mathtt{B}, \mathtt{G}, \mathtt{I}\}$. Then, in each iteration, an element is removed from the incumbent solution, and a new element is included in the solution if the maximum allowed budget K is not exceeded. Since Random Path Relinking is considered, the element to be removed and the one to be included is selected at random.

In the path created in the figure, the first step generates the solution $S_1 = (S_i \setminus \{\mathtt{B}\}) \cup \{\mathtt{D}\} = \{\mathtt{D}, \mathtt{G}, \mathtt{I}\}$, resulting in an objective function value of 19. In the next step, solution S_2 is created by removing node $\mathtt{G}$, $S_2 = S_1 \setminus \{\mathtt{G}\} = \{\mathtt{D}, \mathtt{I}\}$, with an objective function value of 15. Notice that, in this case, no new element is included in the incumbent solution, assuming that, in this point, the available budget would be exceeded. Then, S_3 is generated as $S_3 = (S_2 \setminus \{\mathtt{I}\}) \cup \{\mathtt{A}\} = \{\mathtt{D}, \mathtt{A}\}$, with an objective function value of 16. At this point, there are no nodes to be removed from solution S_3, i.e., $S_3 \setminus S_g = \emptyset$. However, there are still nodes to be included, selecting in this stage node $\mathtt{C}$, resulting in $S_4 = S_3 \cup \{\mathtt{C}\} = \{\mathtt{D}, \mathtt{A}, \mathtt{C}\}$ with an objective function value of 17. In the last step, the guiding solution is reached by including the last node $\mathtt{F}$, finishing the path. It is worth mentioning that none of the solutions created during the path are necessarily a local optimum with respect to any neighborhood. Therefore, the local search method described in Section 4.3 is applied to the best solution found in the path (ties

are broken randomly), excluding the initial and guiding solutions, i.e., S_1 in the figure. Once the path creation has been described, it is necessary to present the Path Relinking variants considered in this work. In particular, two of the most extended variants have been studied: Static Path Relinking (see Section 4.1) and Dynamic Path Relinking (see Section 4.2). The proposed algorithm will leverage the path creating of Path Relinking to select the most promising subsets of influential users of two different solutions and combine them to generate an eventually better set of influential users.

4.1. Static path relinking

Static Path Relinking (SPR) [35] is the most basic version of Path Relinking. SPR requires from an Elite Set (ES) of high quality and diverse solutions that can be generated either at random or by using a more elaborated procedure. In this problem, the ES is generated by using the GRASP algorithm presented in Section 4.3 which balances intensification and diversification. The number of solutions generated will be discussed later in the experimental results section (Section 5). Then, the ES is conformed with the best solutions found with GRASP. Again, the size of the ES is a parameter of the algorithm which will be adjusted in Section 5. Finally, all the solutions in the ES are combined with the Random Path Relinking method. Algorithm 1 shows the pseudocode of the method.

Algorithm 1 SPR$(G = (V, E), \Delta, \delta)$

```
1:  P ← ∅
2:  for i = 1 ... Δ do
3:      S ← Construct(G)
4:      S' ← Improve(S)
5:      if S' ∉ P then
6:          P ← P ∪ {S'}
7:      end if
8:  end for
9:  ES ← SelectBest(P, δ)
10: S_b ← argmax_{S∈ES} TSS(S)
11: for i = 1 ... δ − 1 do
12:     for j = i + 1 ... δ do
13:         S ← RPR(ES_i, ES_j)
14:         S' ← Improve(S)
15:         if TSS(S') > TSS(S_b) then
16:             S_b ← S'
17:         end if
18:     end for
19: end for
20: return S_b
```

The method starts by creating the initial population of solutions P (step 1). Then, SPR iterates until generating a set of Δ solutions (steps 2–8). In each iteration, a solution is generated (step 3) and then improved (step 4) using the constructive and local search methods presented in Section 4.3. The generated solution is then added to the initial population if and only if it has not been explored yet (steps 5–7).

The Elite Set ES is generated with the δ most promising solutions of P (step 9), and the best solution found is initialized (step 10). Then, all the solutions in the ES are combined using the Random Path Relinking method (step 13). The combined solution is later improved (step 14) and compared with the best solution found so far, updating it if necessary (steps 15–17). The method ends returning the best solution found during the search (step 20).

The analysis of computational complexity of SPR can be divided into two different phases: the one corresponding to GRASP phase (steps 2–8) and the one corresponding to the path creation itself (steps 11–19). The complexity of the GRASP phase is detailed in Section 4.3. The complexity of the second phase is evaluated as $O\left(\delta \cdot \delta \cdot \left(n + n^3 \log n + n^2 + n^2\right)\right)$. In particular, the first two δ factors indicate the loops in steps 11 and 12. Then, the complexity of RPR is $O(n)$ since, in the worst case (two completely different solutions), it will perform n iterations. Then, the complexity of the local search is $O\left(n^3 \log n\right)$ as stated in Section 4.3 and, finally, the complexity of comparing two solutions is $O\left(n^2\right)$ for each solution evaluation. Therefore, the final complexity of this method is $O\left(\delta^2 \cdot n^3 \log n\right)$.

4.2. Dynamic path relinking

Dynamic Path Relinking (DPR) [36] avoids the generation of a complete population of solutions and then combine them by dynamically creating new solutions and paths between them. In particular, the method starts by creating the ES with a fixed number of solutions created with GRASP, which is in continuous evolution during the process. Algorithm 2 details the proposed Dynamic Path Relinking method.

Algorithm 2 DPR$(G = (V, E), \Gamma, \gamma)$

```
1:  ES ← ∅
2:  for i = 1 ... Γ do
3:      S ← Construct(G)
4:      S' ← Improve(S)
5:      if S' ∉ ES then
6:          ES ← ES ∪ {S'}
7:      end if
8:  end for
9:  for i = 1 ... γ do
10:     S ← Construct(G)
11:     S' ← Improve(S)
12:     PS ← {S'}
13:     for S_ES ∈ ES do
14:         S_C ← RPR(S_ES, S')
15:         S'_C ← Improve(S_C)
16:         PS ← PS ∪ {S'_C}
17:     end for
18:     UpdateES(ES, PS)
19: end for
20: S_b ← argmax_{S∈ES} TSS(S)
21: return S_b
```

Similarly to SPR, DPR starts by creating the Elite Set ES (steps 1–8). However, contrary to SPR, the ES is initialized with the first Γ solutions generated by applying the constructive and local improvement methods. Then, for a fixed number of iterations γ (which is an input parameter of DPR), a new solution S' is constructed and improved (steps 10–11). The set PS will contain all the new solutions explored during the current iteration, and it is initialized with solution S' (step 12). The method then iterates over every solution $S_{ES} \in ES$ (steps 13–17), creating a path from S_{ES} to the newly generated solution S' using RPR (step 14). The best solution found in the path is then improved and added to PS (steps 15–16). Once all the paths have been created, the method $UpdateES(ES, PS)$ tries to insert every generated solution into the Elite Set (step 18). In particular, if the solution under evaluation S_P is better than the worst solution found in the ES, then S_P is included in the ES, replacing the most similar solution to S_P of the ES among those with worse objective function value than S_p (i.e., the one with the minimum distance to S_P).

In this point, it is important to define a distance metric between two solutions. Since the solution representation for the TSS

is a set of nodes, the distance between two solutions S_1 and S_2 is measured as the number of different nodes. More formally,

$$d(S_1, S_2) = |S_1 \setminus S_2| + |S_2 \setminus S_1|$$

The algorithm ends with the best solution included in the Elite Set, which is also the best solution found in the search (step 21).

Having described the proposal, it is interesting to highlight the dynamic feature of this variant of Path Relinking. In the case of SPR, the initial set of solutions is generated and then combined, but the new solutions found in the path are never considered neither as initial nor guiding solutions. However, DPR dynamically generates new solutions which are considered in future paths, thus leading to a wider exploration of the search space. Section 5 will detailedly analyze the effect of this dynamic feature on the quality of the solutions found.

The computational complexity of this method is evaluated similarly to SPR. The first phase, which generates the ES, has the same complexity as GRASP, $O\left(\Gamma \cdot n^3 \log n\right)$. In the second phase, the dynamic feature of DPR increases the complexity since the local search is performed in each iteration, resulting in a final complexity of $O\left(\gamma \cdot \left(n + n^3 \log n + \Gamma \cdot \left(n + n^3 \log n\right)\right)\right)$, which can be reduced to $O\left(\gamma \cdot \Gamma \cdot \left(n^3 \log n\right)\right)$.

4.3. GRASP

Path Relinking requires from a method to generate high-quality and diverse solutions in order to create promising paths during the search in both static and dynamic variants. Although these solutions can be generated at random, it has been experimentally shown in several works that designing a specific constructive and local improvement method for the problem under consideration usually leads to better results [29–31,37]. In the context of influence maximization problems, the Greedy Randomized Adaptive Search Procedure (GRASP) has been shown to be an effective and efficient method to generate them [38].

GRASP is a trajectory-based metaheuristic originally proposed in [39], and formally defined in [40]. The metaheuristic is divided into two well-differenced phases: construction and local improvement. The key part of GRASP is the construction method, which is able to generate not only high-quality solutions but also diverse ones. Then, the local improvement method is responsible for finding a local optimum with respect to the initial solution. These two phases are iteratively applied until a certain termination criterion is reached, which is usually a maximum number of iterations. Notice that the computational complexity of GRASP directly depends on the complexity of the constructive procedure and the local search method, resulting in the maximum complexity between both of them.

Fig. 3 shows a general view of the main advantages of this metaheuristic. In particular, Fig. 3(a) shows the performance of a completely greedy algorithm, including the local search method. In this case, solution S_1 is generated and, then, the local search is able to reach a local optimum with respect to the considered neighborhood. However, the method stagnates in the local optimum, and it is not able to escape from it.

The advantage of GRASP is shown in Fig. 3(b). In this case, the construction phase of GRASP generates seven diverse and high-quality solutions instead of a single one. The diversification of GRASP increases the probability of reaching different regions of the search space. In the graphical example, this diversification finds solutions S_6 and S_7 which are not the best initial solutions (indeed, S_6 is the worst initial solution in terms of quality), but the application of the local search method ends in a better solution than the one found with the greedy approach.

Constructive method

The constructive method proposed for TSS problem follows the GRASP philosophy of diversification by avoiding totally greedy decisions. Algorithm 3 shows the pseudocode of the proposed method.

Algorithm 3 `Construct`$(G = (V, E), K, \omega)$

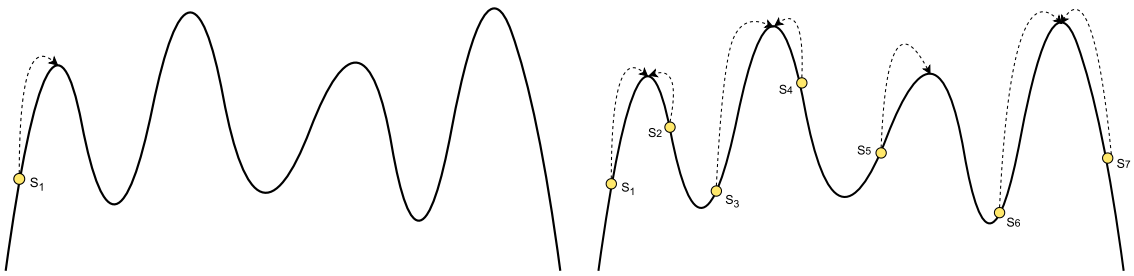
1: $v \leftarrow Random(V)$
2: $S \leftarrow \{v\}$
3: $CL \leftarrow V \setminus \{v\}$
4: **while** $\sum_{v \in S} \alpha(v) \leq K$ **and** $CL \neq \emptyset$ **do**
5: $\quad g_{\min} \leftarrow \min_{c \in CL} g(c, S)$
6: $\quad g_{\max} \leftarrow \max_{c \in CL} g(c, S)$
7: $\quad \mu \leftarrow g_{\max} - \omega \cdot (g_{\max} - g_{\min})$
8: $\quad RCL \leftarrow \{c \in CL : g(c) \geq \mu \wedge \sum_{v \in S} \alpha(v) + \alpha(c) \leq K\}$
9: $\quad v \leftarrow Random(RCL)$
10: $\quad S \leftarrow S \cup \{v\}$
11: $\quad CL \leftarrow CL \setminus \{v\}$
12: **end while**
13: **return** S

The algorithm requires from three input parameters: the input SN, $G = (V, E)$; the maximum allowed budget, K, and the parameter that controls the greediness/randomness of the search, ω. Notice that in the GRASP literature this parameter is usually referred as α. However, we have changed the notation to avoid confusion with the effort of a node, which is named as α.

With the aim of increasing diversity, the method selects the first node to be included at random from the set of users V (step 1), initializing the solution under construction S (step 2). Then, the candidate list CL is created with all the nodes but v (step 3). The constructive method iteratively adds a node to the solution while the budget is not exceeded and the candidate list is not empty (steps 4–12). In each iteration, the minimum and maximum value of a certain greedy function are computed (steps 5–6). The aim of the greedy function is to evaluate how promising a candidate is, and it is a key part of the constructive procedure. The proposed greedy functions will be described below. Then, a threshold μ is evaluated in order to establish a limit to consider whether a node is promising or not, which depends on the value of $g_{\min}$ and $g_{\max}$. Notice that the threshold completely depends on the value of the input parameter $\omega \in [0, 1]$. In particular, if $\omega = 0$, then $\mu = g_{\max}$ and the method becomes completely greedy, while if $\omega = 1$, then $\mu = g_{\min}$ and the method is totally random. Having this in mind, it is important to find a balance between randomness and greediness, so the value of this input parameter will be adjusted in the experiments (see Section 5.1). With this threshold, the restricted candidate list RCL is created (step 8), containing all the nodes whose greedy function value is larger than or equal to the threshold μ, considering that they do not exceed the maximum budget. Once the RCL is constructed, the next element is selected at random from it (since all the nodes in RCL are promising) to favor diversity (step 9). The selected node is then added to the incumbent solution (step 10), updating the CL by removing it (step 11). Finally, the method returns the constructed solution S (step 13). The computational complexity of this method is $O\left(n \cdot O(g)\right)$, where $O(g)$ indicates the complexity of the considered greedy function.

Having defined the constructive procedure, it is necessary to present the considered greedy functions. Specifically, two different greedy functions are evaluated in this research. The first greedy function is traditionally considered in the GRASP literature, and it consists of directly evaluating the objective function value if the node under evaluation were added to the incumbent solution. More formally,

$$g_{of}(c, S) \leftarrow TSS(S \cup \{c\})$$



(a) Example of a greedy algorithm. (b) Example of GRASP performance.

Fig. 3. Comparison between a greedy construction with local search and GRASP algorithm.

which has a computational complexity of $O\left(n^2\right)$ since it is directly the complexity of the objective function evaluation.

However, as it was stated in Section 3, the evaluation of the objective function for the TSS is a rather computationally demanding process, so a new greedy function is proposed with the aim of reducing the computational effort of the evaluation, since it will be performed in each iteration of the construction process.

The second greedy function proposed, named $g_{dg}(c, S)$, considers that the relevance of a node is directly proportional to its degree. In other words, if a node is connected to several nodes, then it will probably influence a large amount of its adjacent nodes. Then, this greedy function is evaluated as the degree of the evaluated node:

$$g_{dg}(c, S) \leftarrow |u \in V : (c, u) \in E|$$

with a computational complexity of $O(1)$ since it only requires to evaluate the degree of a node.

Therefore, the constructive method considering g_{of} has a computational complexity of $O\left(n^3\right)$, while considering g_{dg} reduces its complexity to $O\left(n\right)$. Both greedy functions will be tested in Section 5.1.

Local search

The solution generated with the constructive procedure is not necessarily a local optimum with respect to any neighborhood. For that reason, the second phase of GRASP consists of a local improvement method that finds a local optimum starting from the initial solution. In order to define a local search method, it is necessary to establish three main components: the move operator, the neighborhood explored, and the order in which it is explored.

Starting from the initial solution S, it is not possible to add new nodes, since the constructive procedure stops when the maximum budget is exceeded with any of the remaining nodes. Therefore, the proposed move operator is defined in two steps: remove and add. In particular, the move operator removes a node from the solution and then iteratively adds nodes until the maximum budget is reached:

$$move(S, u, V, K) = (S \setminus \{u\}) \cup \left\{ v \in V \setminus (S \cup \{u\}) : \sum_{s \in S \cup \{u\}} \alpha(s) + \alpha(v) \leq K \right\}$$

Having defined the move operator, the next step to propose a local search method is to define the neighborhood that will be explored during the search. In the case of TSS, the neighborhood is defined as the set of solutions that can be reached by performing a single move operator. More formally,

$$N(S) \leftarrow \{S' \leftarrow move(S, u, V, K) \quad \forall u \in S\}$$

Finally, it is necessary to indicate the order in which the neighborhood is explored. There are two main search strategies in local search methods: the first and best improvement. The former performs the first move that leads to an improvement in the current neighborhood, while the latter explores the complete neighborhood, performing the move that results in the best solution of the neighborhood. Best improvement is usually more computationally demanding than the first improvement, since it requires to explore the complete neighborhood in each iteration, although they have been shown to provide similar results for several combinatorial optimization problems [41,42].

In the context of TSS, the computational effort is a critical part of the algorithm, so we have decided to use the first improvement method with the aim of reducing the computing time to perform a local search method. With the aim of avoiding biasing the search, the neighborhood is explored at random, performing the first movement that results in a better solution.

To sum up, the local search method, denoted as Standard Local Search (SLS), follows a first improvement strategy based on a move operator which removes a node from the solution and replaces it with all the nodes that can be added without exceeding the allowed budget. The computational complexity of a single iteration of SLS is $O\left(|S| \cdot n \cdot n^2\right) = O\left(n^4\right)$ since for each element in S, it tries to perform the move operator with a non-selected node and, finally, it requires to evaluate the resulting solution to check if an improvement is found, which has a complexity of $O\left(n^2\right)$ as stated in Section 3.

With the aim of further reducing the computational effort of the local search method, three improvements are proposed, resulting in an Advanced Local Search (ALS). The first improvement tries to escape from cycling the search by avoiding the exploration of already visited solutions. In order to do so, each visited solution is associated with a unique number, i.e., hash code, evaluated following a hash function. Then, every time a solution is visited, it is evaluated if its corresponding hash code has not already been included in the set of visited solutions. If so, the method undoes the move and continues with the next iteration, avoiding repeating the exploration of the same region of the search space.

The second improvement is devoted to limit the nodes explored during the search, discarding those nodes which will result in an unfeasible solution. In order to do so, the candidate nodes to be added are sorted with respect to their effort value in ascending order. Then, only those nodes whose effort value is smaller than or equal to the available budget are explored. Additionally, to favor diversity, the exploration is performed at random among all nodes that satisfy this constraint.

The objective of the last improvement is to reduce the computing time required to evaluate the influence of a node by caching it. Specifically, the influence of a node (i.e., those nodes that are affected by its activation), is calculated at the beginning of the local search method. Then, every time a node is selected to

be removed or added to the solution, the influence of that node over the other nodes of the graph is updated. As a result, it is not necessary to completely evaluate the objective function in each iteration but to check the corresponding pre-calculated influence.

In order to evaluate the optimization of ALS with respect to SLS, the computational complexity of ALS is calculated. In this case, the traversed nodes are sorted with a complexity of $O(n \log n)$ in each iteration. The last improvement reduces the complexity of evaluating a solution, from $O\left(n^2\right)$ to $O(n)$, since it only requires to traverse the nodes once for updating the value of ψ. Therefore, the final complexity of this method is $O(|S| \cdot n \log n \cdot n) = O\left(n^3 \log n\right)$. It is worth mentioning that this complexity refers to the worst case which, in the case of ALS, is hard to reach, since the second improvement limits the number of nodes explored during the search.

5. Results

This section is devoted to providing a detailed analysis of the performance of the proposed algorithm. In particular, the section is divided into two different subsets of experiments: preliminary and final. The former is designed to select the best configuration for the proposed algorithms in terms of components and parameter values, while the latter performs a competitive testing with the best algorithm found in the literature to analyze the efficiency and efficacy of the proposal. All experiments have been performed on an AMD EPYC 7282 16-core virtual CPU with 32 GB of RAM, using Java 17. All instances and source codes have been made publicly available at https://grafo.etsii.urjc.es/TSS.

The dataset used to perform the experiments has been derived from the best algorithm found in the literature to provide a fair comparison. This set of instances is conformed with 82 instances derived from real-life social networks which have been extensively used in social network analysis. The main drawback of this dataset is that the largest network is conformed with 58 nodes, which might not be challenging enough considering the current size of social networks. To mitigate this drawback, we have added 8 additional instances, with sizes from 67 to 10,312 nodes. The size of each instance is included in Table 8 in Appendix.

Given a node v, its effort $\alpha(v)$ and reward $\beta(v)$ is also given by the instance. The value of the effort and the reward for the original instances have been directly derived from the original dataset. For the new instances, we would like to thank the authors for kindly sending us the code [19] to calculate all the required parameters to generate the instance. Following this definition, the users with larger influence has a larger associated reward. Regarding the effort, those users which are easily influenced has a smaller effort.

5.1. Preliminary results

The preliminary experiments are designed to adjust the parameters of the proposal and to select the best configuration of elements to be included in the algorithm. The experiments have been designed to incrementally configure the algorithm following a sequential design. Although a full-factorial experimentation may better adjust the parameters, it has been shown that the results are usually equivalent [43].

With the aim of avoiding overfitting, the preliminary experiments are performed over a subset of 18 representative instances, which is a 20% of the complete set of 90 instances. All the experiments report the following metrics: Avg., the average objective function value; Dev. (%), the average deviation with respect to the best solution found in the experiment; Time (s), the average computing time required to execute the algorithm measured in seconds; and, # Best, the number of times that the algorithm reaches the best solution found in the experiment. Notice that in the tables in which the optimal value is known, # Best is replaced by # Optima.

Table 1
Comparison of the two proposed greedy functions when considering different values for ω.

Constructive	ω	Avg.	Dev. (%)	Time (s)	#Best
g_{of}	0.25	47.28	4604.56	400.07	15
	0.50	46.94	3985.67	400.04	15
	0.75	44.61	7259.96	400.03	14
	RND	45.00	5435.67	400.04	15
g_{dg}	0.25	3530.56	1.08	160.89	15
	0.50	3551.00	2.06	137.62	12
	0.75	3571.39	1.37	**121.09**	14
	RND	**3637.94**	**0.40**	122.42	**17**

Table 2
Comparison between the two proposed local search strategies.

Algorithm	Avg.	Dev. (%)	Time (s)	#Best
SLS	3546.22	0.95	1202.26	16
ALS	**3828.61**	**0.00**	**285.79**	**18**

The first experiment is designed to select the greedy function to be considered in the constructive phase of GRASP, as well as to select the best value for the ω parameter, which controls the greediness of the construction. In particular, greedy functions g_{of} and g_{dg} are tested, each one of them considering the values $\omega = \{0.25, 0.50, 0.75, RND\}$, with a fixed number of 100 iterations. Table 1 shows the results obtained by each combination of parameters. Notice that the maximum time allowed to each constructive is set to 3600 s.

The most remarkable thing about the results is that the greedy function based on the objective function value, g_{of}, consistently produces drastically worse results than the ones provided by g_{dg}. The rationale behind this is the inclusion of more challenging instances, in which g_{of} only generates a small number of solutions in the maximum allowed time, thus resulting in low-quality solutions.

If we now analyze the results obtained by g_{dg}, the computing time is clearly smaller than g_{of}, being able to solve all the instances without reaching the maximum allowed time. It is worth mentioning that the differences in computing time among the ω values are negligible. The best results in terms of quality are obtained with $\omega = RND$, with the smallest deviation (0.40%) and the largest number of best solutions found (17 out of 18). Therefore, we select g_{dg} and $\omega = RND$ for the constructive procedure.

The second experiment is designed for evaluating the performance of ALS with respect to SLS. Since the SLS is rather computationally demanding, the maximum allowed computing time is set to three hours per instance (10,800 s). Table 2 shows the results obtained when comparing both local search strategies.

Analyzing the quality of the local search methods, both are equivalent when considering the smallest instances. However, the differences appear when including the most challenging ones. In this case, SLS is not able to finish, so it is not able to reach the best solution in 2 out of 18 instances. Regarding the computing time, ALS is more than four times faster than SLS. Therefore, we select ALS for the remaining experiments as the local search method.

Having configured the constructive and local search methods, the next experiment is devoted to establishing the size of the initial population, Δ, and the Elite Set, δ, for the SPR. In order to do so, we have fixed $\delta = 10$ and vary Δ in the range $[10, 50]$ with a step of 10. Table 3 shows the results obtained.

It can be derived from the results that even considering the smallest initial population of $\Delta = 10$ results in high-quality solutions. However, it can be seen how the efficacy of the algorithm

Table 3
Comparison of different sizes for the initial population of Static Path Relinking when fixing the Elite Set size to $\delta = 10$.

Δ	Avg.	Dev. (%)	Time (s)	#Best
10	3799.50	0.51	**86.61**	15
20	3845.00	0.59	94.93	15
30	3845.61	0.01	115.51	17
40	**3846.78**	**0.00**	165.64	**18**
50	**3846.78**	**0.00**	206.36	**18**

Table 4
Comparison of GRASP isolated and coupled with SPR to analyze the contribution of each part of the algorithm.

Algorithm	Avg.	Dev. (%)	Time (s)	#Best
GRASP	3637.94	1.97	**122.42**	15
SPR	**3846.78**	**0.00**	165.64	**18**

Table 5
Comparison between different size of the Dynamic Path Relinking Elite Set and the new solutions.

Γ	γ	Avg.	Dev. (%)	Time (s)	#Best
10	10	3842.78	25.05	**192.66**	16
10	20	**3846.78**	**0.00**	417.30	**18**
20	10	**3846.78**	**0.00**	411.66	**18**
20	20	**3846.78**	**0.00**	1098.50	**18**

increases with the size of the initial population, stagnating when $\Delta = 40$. In particular, $\Delta = 50$ provides the same results but requires almost 30% more computing time. Therefore, we select $\Delta = 40$ for the final configuration of SPR. We have performed the same experimentation for the value of δ fixing the initial population size to $\Delta = 40$, but there are no differences in quality among the different values. For this reason, we have selected an Elite Set size of $\delta = 10$.

Having selected the best GRASP configuration, i.e., constructive method using the greedy function g_{dg} and ALS as a local search, and the best SPR configuration, i.e., initial population size of 40, it is necessary to analyze the contribution of SPR to GRASP. In order to do so, an experiment is performed comparing the results obtained by GRASP isolated and then coupled with SPR. The results are shown in Table 4.

As expected, SPR is able to reach all the best solutions of the experiments, with a deviation of 0%. Furthermore, it does not considerably increase the computing time, being an efficient method for the TSS. On the contrary, GRASP fails to reach 3 out of 18 best solutions, remaining at a deviation of nearly 2% with respect to the solution found by SPR. These results confirms the contribution of SPR to the final algorithm.

The last preliminary experiment is designed to configure the dynamic variant of Path Relinking, DPR. In this case, there are two input parameters to adjust: Γ, the size of the Elite Set, and γ, the number of new solutions generated to combine with each solution of the Elite Set. The tested values are $\Gamma = \{10, 20\}$ and $\gamma = \{10, 20\}$, evaluating all possible combinations of these values. The results are shown in Table 5.

As expected, the computing time increases drastically with the size of the Elite Set and the initial population. However, the quality does not vary when considering pairs $(\Delta, \delta) = \{(10, 20), (20, 10), (20, 20)\}$. The combination (10, 10) can be directly discarded due to the large deviation of 25% provided. Therefore, we select the configuration that provides the smallest computing time, which is $\Delta = 20$ and $\delta = 10$.

5.2. Final results

Since the dataset has been increased with more complex and challenging instances, it is necessary to establish a time limit for the exact algorithm. If the Gurobi solver reaches that time limit, it returns the best solution found so far, which is not necessarily the optimal value. In particular, the time limit given to the Gurobi solver is set to 108 000 s (approximately 30 h).

Table 6
Comparison of SPR, DPR, SA, CELF and Gurobi solver when considering the original dataset in which Gurobi is able to reach the optimal value.

Algorithm	Avg.	Dev. (%)	Time (s)	#Optimal
Gurobi	**45.38**	**0.00**	117.14	**82**
DPR	44.34	2.58	**0.01**	76
SPR	44.54	1.82	**0.01**	79
SA	46.31	**4.86**	**0.01**	76
CELF	42.07	12.19	**0.01**	61

The best previous approach is an exact method which shows its limits when dealing with larger and more complex instances. In order to evaluate the contribution of our proposal, we have also included an additional metaheuristic algorithm for performing a comparison with SPR and DPR. In particular, we have selected Simulated Annealing (SA), which is a metaheuristic based on the analogy between an optimization process and a thermodynamic process known as annealing. It is a search method which tries to escape from local optima allowing to explore worse solutions if those solutions satisfy certain criteria. SA was originally proposed by Kirkpatrick et al. [44] and it has been successfully applied in a wide variety of hard combinatorial optimization problems. SA has been successfully applied in several works related to influence maximization problems [45,46].

Additionally, the well-known Cost-Effective Lazy Forward (CELF) selection algorithm [47], which has been widely used in the context of influence maximization problems and, in particular, in TSS [48], is included in the comparison. CELF is a greedy procedure which leverages the submodularity property of the network to considerably reduce the computational effort of the greedy hill-climbing algorithm. The main objective of this optimization is to scale to large problems, reaching near optimal placements. This improvement makes CELF approximately 700 times faster than the original procedure.

There exists several implementation of SA which are publicly available. For this work, we have selected the one provided by the Metaheuristic Optimization FramewoRK [49], which has been tested over several hard optimization problems [50,51] SA requires from several parameters, which has been set to the values recommended in the literature. The parameters used are: cooldown (C), which indicates the temperature variation, is set to 0.98; the initial temperature (T_i), which represents the worst value that can be found in the neighborhood, set to 100 000; the maximum number of iterations (I), set to 100; and the neighborhood considered during the search, which is based on the move operator defined in this work. The complexity of this implementation is divided into the construction phase and the proper SA algorithm. The complexity of the constructive phase is equal to GRASP complexity described in Section 4.3, while the complexity of SA is evaluated as $O\left(T_i \cdot I \cdot n^2\right)$. We refer the reader to [52] to a deeper analysis of the complexity of SA.

The results are divided into two different experiments. First of all, SPR and DPR are tests when considering the set of original instances in which the exact method is able to reach the optimal value. Table 6 shows the results obtained.

As it can be derived from the results, SPR performs slightly better than DPR in this set of instances, being able to reach 79 out of 82 optimal solutions, while DPR reaches 76. It is important to remark that the average deviation of both methods, smaller than 0.05, indicates that in those instances in which neither SPR nor DPR are able to reach the optimal value, they stay really close to it. In order to confirm this hypothesis, we have conducted a

Table 7
Comparison of CELF, SA, SPR and DPR over the set of largest and most complex instances.

Instance	CELF				SA			
	Avg.	Dev. (%)	Time (s)	#Best	Avg.	Dev. (%)	Time (s)	#Best
PRISON*	240	11.11	0.02	0	**270**	**0.00**	0.34	**1**
EMAIL-EU-CORE*	4672	1.79	22.71	0	**4757**	**0.00**	267.78	**1**
EGO-FACEBOOK	**19462**	**0.00**	3609.75	**1**	**19462**	**0.00**	1044.35	**1**
CA-GRQC	22487	5.05	12441.33	0	**23684**	**0.00**	5364.14	**1**
TWITCH_EN	25060	0.22	16833.33	0	23853	5.03	5885.88	0
LASTFM_ASIA	**25005**	**0.00**	26017.00	**1**	23000	8.02	6160.10	0
CA-HEPTH	44451	1.16	165624.79	0	**44972**	**0.00**	9105.73	**1**
BLOG_CATALOG3	**46732**	**0.00**	44674.80	**1**	46418	0.67	8407.40	0
Summary	23514.13	2.23	33652.97	3	23302.00	1.71	4534.97	5

Instance	SPR				DPR			
	Avg.	Dev. (%)	Time (s)	#Best	Avg.	Dev. (%)	Time (s)	#Best
PRISON*	**270**	**0.00**	**0.07**	**1**	**270**	**0.00**	0.16	**1**
EMAIL-EU-CORE*	**4757**	**0.00**	**84.48**	**1**	**4757**	**0.00**	262.59	**1**
EGO-FACEBOOK	**19462**	**0.00**	**279.47**	**1**	**19462**	**0.00**	769.70	**1**
CA-GRQC	23630	0.23	**442.07**	0	**23684**	**0.00**	2166.29	**1**
TWITCH_EN	24853	1.06	**680.77**	0	**25116**	**0.00**	2230.12	**1**
LASTFM_ASIA	24556	1.80	**760.97**	0	24780	0.90	2409.55	0
CA-HEPTH	44909	0.14	**2140.92**	0	**44972**	**0.00**	7743.06	**1**
BLOG_CATALOG3	46595	0.29	**1336.91**	0	46692	0.09	8705.44	0
Summary	23629.00	0.44	**715.71**	3	**23716.63**	**0.12**	3035.87	**6**

pairwise non-parametric Wilcoxon statistical test between SPR and Gurobi solver, obtaining a p-value equal to 0.109, which indicates that, with a confidence interval of 95%, there are not statistically significant differences between those methods. Regarding the SA, it is worth mentioning that it is able to reach 76 out of 82 instances with a deviation of 4.86%, requiring negligible time such as SPR and DPR. With respect to CELF, the algorithm requires from negligible computing times as DPR, SPR and SA, but it only reaches 61 out of 82 optimal solutions, with a deviation of 12.19%. From these results, we can obtain two main conclusions: SA is a competitive algorithm for the TSS, and the proposed DPR and SPR significantly contribute to the quality of the obtained solutions, as it can be seen in the smaller deviation with respect to the optimal value.

The last experiment is devoted to evaluate the performance of the proposed algorithms and the Gurobi solver when considering the most challenging and realistic instances. Table 7 shows the results obtained in the set of large instances. In this case, we show the results disaggregated, since it is conformed with 8 instances that can be individually analyzed.

It is worth mentioning that the Gurobi solver is only able to provide the optimal solution for 2 out of 8 instances derived from the new set of complex instances marked with an asterisk in the corresponding instance name. For the remaining instances, Gurobi is not even able to load the model in memory, which highlights the need to consider metaheuristic algorithms for this set of challenging instances. In particular, in those instances where Gurobi reaches the optimal value, SA, SPR and DPR are also able to find it. However, CELF is not able to reach the optimal value for these two instances. Additionally, for the instance EMAIL-EU-CORE, Gurobi requires almost 30 h to find the optimal value, while SA requires 268 s, DPR 262 s and SPR only 85 s.

Analyzing the instances in which Gurobi is not able to even load the model, SPR requires from smaller computing time than DPR in general, but it provides worse results in terms of quality. Regarding SA, it is able to provide competitive results in these challenging instances. Specifically, SPR reaches the best solution in 3 out of 8 instances, SA reaches 5 out of 8 best solutions, and, finally, DPR reaches all the best solutions but for two instances in which CELF is able to provide slightly better results. It is worth mentioning that CELF requires from approximately five times the computing time required by DPR, thus being DPR much more scalable for large scale networks. In terms of deviation, CELF provides the worst results with a 2.23%, followed by SA with 1.71%, but it is considerably larger than the one obtained by SPR and DPR. Specifically, the average deviation obtained by SPR is considerably small (0.44%), and DPR is able to reach a deviation of 0.12%. Since the deviation of SPR is really close to 0%, we conduct a pairwise non-parametric Wilcoxon statistical test to evaluate if there are statistically significant differences between SPR and DPR. The resulting p-value of 0.04, smaller than 0.05, indicates that DPR is statistically better than SPR. These results highlights the contribution of SPR and DPR to the state of the art of TSS.

6. Conclusions

This research presents two different Path Relinking approaches for solving the Target Set Selection problem. In particular, the Static Path Relinking variant is compared with the Dynamic Path Relinking variant over a set of challenging instances derived from real-life social networks. Both methods are compared with the best approach found in the literature, which is an exact algorithm implemented in the Gurobi commercial solver. In the comparison, the limits of the exact approach are shown, being unable to even load the most complex instances, while SPR and DPR are able to provide high-quality solutions in reasonable computing time. Additionally, a complexity analysis has been included for each algorithm with the aim of analyzing the computational effort required to execute each one of them.

As a conclusion, both SPR and DPR are able to provide promising solutions for the TSS, each one of them being suitable for different situations. On the one hand, if the computing time is a hard constraint, we do recommend considering SPR since the quality of the solutions is not drastically worse. On the other hand, if the maximum computing time is not a critical part of the problem, DPR is able to provide better results in terms of quality.

The proposed algorithms have been compared with a Simulated Annealing implementation, which has been successfully applied in several influence maximization problems, and with CELF, which is a widely used method in the context of influence maximization and, particularly, in TSS. The results obtained highlight the appropriateness of designing an specific algorithm for solving the TSS such as the proposal of this research.

Table 8
Size of each instance considered in this work.

ID	Datasets	N	ID	Datasets	N
1	Knoke Bureaucracies KNOKI	10	73	Zachary Karate Club ZACHE	34
2	**Knoke Bureaucracies KNOKM**	10	74	Zachary Karate Club ZACHC	34
3	Roethlisberger & Dickson Bank RDGAM	14	**75**	**Bernard & Killworth Technical BKTECC**	34
4	Roethlisberger & Dickson Bank RDPOS	14	76–77	Kapferer Tailor Shop KAPFTI1 and KAPFTI2	39
5	**Roethlisberger & Dickson Bank RDHLP**	14	**78–79**	**Kapferer Tailor Shop KAPFTS1 and KAPFTS2**	39
6	Kapferer Mine KAPFMU	15	80	Bernard & Killworth Office BKOFFC	40
7	Kapferer Mine KAPFMM	15	81	Bernard & Killworth Ham Radio BKHAMC	44
8	Thurman Office THURA	15	**82**	**Bernard & Killworth Fraternity BKFRAC**	58
9	Thurman Office THURM	15	83	Gagnon & Macrae Prision	67
10–24	**Newcomb Fraternity NEWC1...NEWC15**	17	84	email-Eu-core network	1005
25	Davis Southern Club Women DAVIS	18	85	Social circles: Facebook	4039
26–28	Sampson Monastery SAMPLK1...SAMPLK3	18	**86**	**General Relativity and Quantum Cosmology collaboration network**	5242
29	Sampson Monastery SAMPES	18	87	Twitch EN	7126
30	Sampson Monastery SAMPIN	18	88	LastFM Asia Social Network	7624
31–51	**Krackhardt Office css KRACKAD1...KRACKAD21**	21	**89**	**High Energy Physics Theory collaboration network**	9877
52–72	**Krackhardt Office css KRACKFR1...KRACKFR21**	21	90	BlogCatalog3	10312

The successful application of Path Relinking metaheuristic to TSS lead us to propose several future lines of research. First of all, it would be interesting to evaluate the Path Relinking proposal over the opposite problem: minimization of information spreading, which is interesting in topics such as misinformation diffusion or disease control, among others. Another interesting line of research is the adaptation of Path Relinking to other well-known influence maximization problems, such as Budgeted Influence Maximization or Influence Spectrum Problem.

CRediT authorship contribution statement

Isaac Lozano-Osorio: Validation, Resources, Conceptualization. **Andrea Oliva-García:** Software, Investigation, Data curation. **Jesús Sánchez-Oro:** Writing – review & editing, Writing – original draft, Supervision, Methodology.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Jesus Sanchez-Oro reports financial support was provided by Spain Ministry of Science and Innovation. Isaac Lozano-Osorio reports financial support was provided by Spain Ministry of Science and Innovation. Andrea Oliva-Garcia reports financial support was provided by Spain Ministry of Science and Innovation.

Data availability

Data availability is public at https://grafo.etsii.urjc.es/TSS/.

Acknowledgments

The authors acknowledge support from the Spanish Ministry of "Ciencia, Innovación MCIN/AEI/10.13039/501100011033/ FEDER, UE) under grant ref. PID2021-126605NB-I00 and PID2021-125709OA-C22, the "Comunidad de Madrid, Spain" and "Fondos Estructurales" of the European Union with grant reference S2018/TCS-4566.

Appendix

This Appendix shows the detailed size of each considered social network in Table 8. In particular, each instance is associated with an identification number (column ID), a name (column Datasets), and the number of nodes (column N). The subset of instances used in the preliminary experiments are those highlighted in bold font, specifically: [2, 5, 10, 20, 24, 31, 38, 43, 47, 51, 57, 65, 70, 75, 78, 82, 86, 89].

Instances with ID from 1 to 82 are directly derived from the best method found in the literature [19]. The remaining instances, which will also be included in our public repository https://grafo.etsii.urjc.es/TSS, have been derived from the following social network repositories:

- Instance 83: Same repository as the original dataset, http://vlado.fmf.uni-lj.si/pub/networks/data/UciNet/UciData.htm
- Instances from 84 to 89: SNAP dataset, https://snap.stanford.edu/data/
- Instance 90: BlogCatalog, http://datasets.syr.edu/datasets/BlogCatalog3.html

References

[1] J.V. Chen, B.c. Su, A.E. Widjaja, Facebook C2C social commerce: A study of online impulse buying, Decis. Support Syst. 83 (2016) 57–69, http://dx.doi.org/10.1016/j.dss.2015.12.008.

[2] S. Ryu, J. Park, The effects of benefit-driven commitment on usage of social media for shopping and positive word-of-mouth, J. Retail. Consum. Serv. 55 (2020) 102094, http://dx.doi.org/10.1016/j.jretconser.2020.102094.

[3] D.M.J. Lazer, M.A. Baum, Y. Benkler, A.J. Berinsky, K.M. Greenhill, F. Menczer, M.J. Metzger, B. Nyhan, G. Pennycook, D. Rothschild, M. Schudson, S.A. Sloman, C.R. Sunstein, E.A. Thorson, D.J. Watts, J.L. Zittrain, The science of fake news, Science 359 (6380) (2018) 1094–1096, http://dx.doi.org/10.1126/science.aao2998.

[4] P.A. Dreyer, F.S. Roberts, Irreversible k-threshold processes: Graph-theoretical threshold models of the spread of disease and of opinion, Discrete Appl. Math. 157 (7) (2009) 1615–1627, http://dx.doi.org/10.1016/j.dam.2008.09.012.

[5] K. Sörensen, F. Glover, Metaheuristics, Encycl. Oper. Res. Manag. Sci. 62 (2013) 960–970.

[6] M. Richardson, R. Agrawal, P. Domingos, Trust management for the semantic web, in: D. Fensel, K. Sycara, J. Mylopoulos (Eds.), The Semantic Web - ISWC 2003, Springer Berlin Heidelberg, Berlin, Heidelberg, 2003, pp. 351–368.

[7] D. Kempe, J. Kleinberg, E. Tardos, Maximizing the spread of influence through a social network, in: KDD '03: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2003, http://dx.doi.org/10.1145/956750.956769.

[8] N. Chen, On the approximability of influence in social networks, SIAM J. Discrete Math. 23 (3) (2009) 1400–1415, http://dx.doi.org/10.1137/08073617X.

[9] C.C. Centeno, M.C. Dourado, L.D. Penso, D. Rautenbach, J.L. Szwarcfiter, Irreversible conversion of graphs, Theoret. Comput. Sci. 412 (29) (2011) 3693–3700, http://dx.doi.org/10.1016/j.tcs.2011.03.029.

[10] F. Cicalese, G. Cordasco, L. Gargano, M. Milanič, J. Peters, U. Vaccaro, Spread of influence in weighted networks under time and budget constraints, Theoret. Comput. Sci. 586 (2015) 40–58, http://dx.doi.org/10.1016/j.tcs.2015.02.032.

[11] L. Narayanan, K. Wu, How to choose friends strategically, Theoret. Comput. Sci. 811 (2020) 99–111, http://dx.doi.org/10.1016/j.tcs.2018.07.013.

[12] C.Y. Chiang, L.H. Huang, B.J. Li, J. Wu, H.G. Yeh, Some results on the target set selection problem, J. Comb. Optim. 25 (4) (2012) 702–715, http://dx.doi.org/10.1007/s10878-012-9518-3.

[13] A. Nichterlein, R. Niedermeier, J. Uhlmann, M. Weller, On tractable cases of target set selection, Soc. Netw. Anal. Min. 3 (2) (2012) 233–256, http://dx.doi.org/10.1007/s13278-012-0067-7.

[14] E. Ackerman, O. Ben-Zwi, G. Wolfovitz, Combinatorial model and bounds for target set selection, Theoret. Comput. Sci. 411 (44–46) (2010) 4017–4022, http://dx.doi.org/10.1016/j.tcs.2010.08.021.

[15] M. Fardad, G. Kearney, On a linear programming approach to the optimal seeding of cascading failures, in: 2017 IEEE 56th Annual Conference on Decision and Control, CDC, IEEE, 2017, http://dx.doi.org/10.1109/CDC.2017.8263650.

[16] S. Raghavan, R. Zhang, A branch-and-cut approach for the weighted target set selection problem on social networks, INFORMS J. Optim. 1 (4) (2019) 304–322, http://dx.doi.org/10.1287/ijoo.2019.0012.

[17] F.G. Baghbani, M. Asadpour, H. Faili, Integer linear programming for influence maximization, Iran. J. Sci. Technol. Trans. Electr. Eng. 43 (3) (2019) 627–634, http://dx.doi.org/10.1007/s40998-019-00178-7.

[18] G. Cordasco, L. Gargano, M. Mecchia, A.A. Rescigno, U. Vaccaro, Discovering small target sets in social networks: A fast and effective algorithm, Algorithmica 80 (6) (2017) 1804–1833, http://dx.doi.org/10.1007/s00453-017-0390-5.

[19] S.V. Ravelo, C.N. Meneses, Generalizations, formulations and subgradient based heuristic with dynamic programming procedure for target set selection problems, Comput. Oper. Res. 135 (2021) 105441, http://dx.doi.org/10.1016/j.cor.2021.105441.

[20] M. Chopin, A. Nichterlein, R. Niedermeier, M. Weller, Constant thresholds can make target set selection tractable, Theory Comput. Syst. 55 (1) (2013) 61–83, http://dx.doi.org/10.1007/s00224-013-9499-3.

[21] F. Cicalese, G. Cordasco, L. Gargano, M. Milanič, U. Vaccaro, Latency-bounded target set selection in social networks, Theoret. Comput. Sci. 535 (2014) 1–15, http://dx.doi.org/10.1016/j.tcs.2014.02.027.

[22] M. Charikar, Y. Naamad, A. Wirth, On Approximating Target Set Selection, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik GmbH, Wadern/Saarbruecken, Germany, 2016, http://dx.doi.org/10.4230/LIPICS.APPROX-RANDOM.2016.4.

[23] X. Liu, Z. Yang, W. Wang, Exact solutions for latency-bounded target set selection problem on some special families of graphs, Discrete Appl. Math. 203 (2016) 111–116, http://dx.doi.org/10.1016/j.dam.2015.09.005.

[24] S.V. Ravelo, C.N. Meneses, E.A. Anacleto, NP-hardness and evolutionary algorithm over new formulation for a Target Set Selection problem, in: 2020 IEEE Congress on Evolutionary Computation, CEC, IEEE, 2020, http://dx.doi.org/10.1109/CEC48606.2020.9185558.

[25] A.L. Serrano, C. Blum, A biased random key genetic algorithm applied to target set selection in viral marketing, in: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '22, Association for Computing Machinery, New York, NY, USA, 2022, pp. 241–250, http://dx.doi.org/10.1145/3512290.3528785.

[26] R. Olivares, F. Muñoz, F. Riquelme, A multi-objective linear threshold influence spread model solved by swarm intelligence-based methods, Knowl.-Based Syst. 212 (2021) 106623, http://dx.doi.org/10.1016/j.knosys.2020.106623.

[27] S. Banerjee, M. Jenamani, D.K. Pratihar, A survey on influence maximization in a social network, Knowl. Inf. Syst. 62 (9) (2020) 3417–3455, http://dx.doi.org/10.1007/s10115-020-01461-4.

[28] Z. Aghaee, M.M. Ghasemi, H.A. Beni, A. Bouyer, A. Fatemi, A survey on meta-heuristic algorithms for the influence maximization problem in the social networks, Computing 103 (11) (2021) 2437–2477, http://dx.doi.org/10.1007/s00607-021-00945-7.

[29] V. Campos, R. Martí, J. Sánchez-Oro, A. Duarte, GRASP with path relinking for the orienteering problem, J. Oper. Res. Soc. 65 (12) (2014) 1800–1813, http://dx.doi.org/10.1057/jors.2013.156.

[30] A. Duarte, J. Sánchez-Oro, M.G. Resende, F. Glover, R. Martí, Greedy randomized adaptive search procedure with exterior path relinking for differential dispersion minimization, Inform. Sci. 296 (2015) 46–60, http://dx.doi.org/10.1016/j.ins.2014.10.010.

[31] D. Cuellar-Usaquén, C. Gomez, D. Álvarez-Martínez, A GRASP/Path-Relinking algorithm for the traveling purchaser problem, Int. Trans. Oper. Res. 30 (2) (2021) 831–857, http://dx.doi.org/10.1111/itor.12985.

[32] M.G. Resende, C.C. Ribeiro, GRASP with path-relinking: Recent advances and applications, in: T. Ibaraki, K. Nonobe, M. Yagiura (Eds.), Metaheuristics: Progress As Real Problem Solvers, Springer US, Boston, MA, 2005, pp. 29–63, http://dx.doi.org/10.1007/0-387-25383-1_2.

[33] F. Glover, M. Laguna, Tabu Search, Springer, 1998.

[34] M. Laguna, R. Marti, GRASP and path relinking for 2-layer straight line crossing minimization, INFORMS J. Comput. 11 (1) (1999) 44–52, http://dx.doi.org/10.1287/ijoc.11.1.44.

[35] C.C. Ribeiro, M.G. Resende, Path-relinking intensification methods for stochastic local search algorithms, J. Heuristics 18 (2) (2012) 193–214, http://dx.doi.org/10.1007/s10732-011-9167-1.

[36] M. Resende, R.M.M. Gallego, A. Duarte, GRASP and path relinking for the max–min diversity problem, Comput. Oper. Res. 37 (3) (2010) 498–508, http://dx.doi.org/10.1016/j.cor.2008.05.011, Hybrid Metaheuristics.

[37] S. Pérez-Peló, J. Sánchez-Oro, A. Duarte, Finding weaknesses in networks using Greedy Randomized Adaptive Search Procedure and Path Relinking, Expert Syst. 37 (6) (2020) e12540, http://dx.doi.org/10.1111/exsy.12540.

[38] I. Lozano-Osorio, J. Sánchez-Oro, A. Duarte, Ó. Cordón, A quick GRASP-based method for influence maximization in social networks, J. Ambient Intell. Humaniz. Comput. (2021) http://dx.doi.org/10.1007/s12652-021-03510-4.

[39] T.A. Feo, M.G. Resende, A probabilistic heuristic for a computationally difficult set covering problem, Oper. Res. Lett. 8 (2) (1989) 67–71, http://dx.doi.org/10.1016/0167-6377(89)90002-3.

[40] T.A. Feo, M.G.C. Resende, S.H. Smith, A greedy randomized adaptive search procedure for maximum independent set, Oper. Res. 42 (5) (1994) 860–878, http://dx.doi.org/10.1287/opre.42.5.860.

[41] I. Lozano-Osorio, J. Sanchez-Oro, M.A. Rodriguez-Garcia, A. Duarte, Optimizing computer networks communication with the band collocation problem: A variable neighborhood search approach, Electronics 9 (11) (2020) http://dx.doi.org/10.3390/electronics9111860.

[42] I. Lozano-Osorio, A. Martínez-Gavara, R. Martí, A. Duarte, Max–min dispersion with capacity and cost for a practical location problem, Expert Syst. Appl. 200 (2022) 116899, http://dx.doi.org/10.1016/j.eswa.2022.116899.

[43] J. Sánchez-Oro, M. Laguna, R. Martí, A. Duarte, Scatter search for the bandpass problem, J. Global Optim. 66 (4) (2016) 769–790, http://dx.doi.org/10.1007/s10898-016-0446-0.

[44] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by simulated annealing, Science 220 (4598) (1983) 671–680, http://dx.doi.org/10.1126/science.220.4598.671.
[45] Q. Jiang, G. Song, C. Gao, Y. Wang, W. Si, K. Xie, Simulated annealing based influence maximization in social networks, Proc. AAAI Conf. Artif. Intell. 25 (1) (2011) 127–132, http://dx.doi.org/10.1609/aaai.v25i1.7838.
[46] S.J. Liu, C.Y. Chen, C.W. Tsai, An effective simulated annealing for influence maximization problem of online social networks, Procedia Comput. Sci. 113 (2017) 478–483, http://dx.doi.org/10.1016/j.procs.2017.08.306, The 8th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN 2017) / The 7th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (ICTH-2017) / Affiliated Workshops.
[47] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, N. Glance, Cost-effective outbreak detection in networks, in: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2007, pp. 420–429, http://dx.doi.org/10.1145/1281192.1281239.
[48] C. Wang, L. Deng, G. Zhou, M. Jiang, A global optimization algorithm for target set selection problems, Inform. Sci. 267 (2014) 101–118, http://dx.doi.org/10.1016/j.ins.2013.09.033.
[49] R. Martín-Santamaría, S. Cavero, A. Herrán, A. Duarte, J.M. Colmenar, A practical methodology for reproducible experimentation: an application to the double-row facility layout problem, Evol. Comput. (2022) 1–35, http://dx.doi.org/10.1162/evco_a_00317.
[50] I. Lozano-Osorio, J. Sánchez-Oro, A. Martínez-Gavara, A.D. López-Sánchez, A. Duarte, An efficient fixed set search for the covering location with interconnected facilities problem, in: L. Di Gaspero, P. Festa, A. Nakib, M. Pavone (Eds.), Metaheuristics, Springer International Publishing, Cham, 2023, pp. 485–490.
[51] J. Yuste, E.G. Pardo, A. Duarte, Variable neighborhood descent for software quality optimization, in: L. Di Gaspero, P. Festa, A. Nakib, M. Pavone (Eds.), Metaheuristics, Springer International Publishing, Cham, 2023, pp. 531–536.
[52] G.H. Sasaki, B. Hajek, The time complexity of maximum matching by simulated annealing, J. ACM 35 (2) (1988) 387–403, http://dx.doi.org/10.1145/42282.46160.

Isaac Lozano-Osorio was born in Quintanar de la Orden (Spain) on August 11, 1997. He is a Ph.D. student at the University Rey Juan Carlos, Madrid, where he is part of the Group for Research in Algorithms For Optimization (GRAFO). His main research interests focus on the applicability of Artificial Intelligence (AI), focus specially on heuristics and metaheuristics, for solving hard optimization problems related to Social Network Influence Maximization Problems.

Andrea Oliva-García is a MEng student at Menéndez Pelayo International University, Madrid. She is part of the Group for Research in Algorithms For Optimization (GRAFO) at University Rey Juan Carlos. Her research interests focus on the applicability of Artificial Intelligence (AI) techniques to solve problems related to social network.

Jesús Sánchez-Oro was born in Madrid (Spain) on December 31, 1987. He holds a degree in Computer Science from the Universidad Rey Juan Carlos (2010), his Master's degree in Computer Vision from the Universidad Rey Juan Carlos in 2011, and his Ph.D. in Computer Science in 2016 from the Universidad Rey Juan Carlos. He is visiting professor at the Computer Science Department, and he is a member of the Group for Research on Algorithms For Optimization (GRAFO). His main research interests focus on Artificial Intelligence and Operations Research, specially in heuristics and metaheuristics for solving hard optimization problems.