NP-hardness and evolutionary algorithm over new formulation for a Target Set Selection problem

Santiago V. Ravelo
Institute of Informatics
Federal University of Rio Grande do Sul
Porto Alegre, Brazil
santiago.ravelo@inf.ufrgs.br

Cláudio N. Meneses
Center of Mathematics, Computation and Cognition
Federal University of ABC
São Paulo, Brazil
claudio.meneses@ufabc.edu.br

Eduardo A.J. Anacleto
Center of Mathematics, Computation and Cognition
Federal University of ABC
São Paulo, Brazil
eduardo.anacleto@ufabc.edu.br

***Abstract*—This work considers the Target Set Selection problem, which can be used to model the propagation and consumption of information, data, ideas and products through networks, with applications in marketing, medicine, sociology and bioinformatics. We propose a new version of the problem and prove it belongs to the NP-hard class. We also design an evolutionary algorithm that uses, in the crossover and mutation operators, exact solutions of sub-problems which were modeled by a new mathematical formulation. We test our approach over a benchmark of instances constructed from real-world data sets.**

***Index Terms*—Target Set Selection, NP-hard, Binary Linear Program, Evolutionary algorithm**

I. Introduction

The growth of social networks in the last decades have shown the critical role they play in society, being used as communication platforms between individuals, political campaign podiums, advertising staging, scientific and technological cooperation resources, teaching and learning interfaces, among other information and content facilitators. Some of them influence relations between the individuals or the entities, e.g., the friendship relations in Facebook, the followers on Instagram, Twitter and YouTube channels, the groups of WhatsApp and Telegram, or the profiles and solutions in Stack-Exchange and Stack-Overflow. Given the influences between the network individuals, one can model the spread of information, ideas or product acquisition through the network with diverse applications in several areas such as economy, sociology or medicine. A class of problems that studies those spreads is denoted as Target Set Selection.

Target set selection problems receive a network and an influence relation between the individuals, where each person can be in an influenced (activated) state or not. Whenever an individual is influenced, he/she becomes an influencer and helps to propagate the information through the network. The objective is to select a minimum group of initial influencers or to maximize the influenced sub-network size. Some practical applications arise from marketing, where it is required to promote the selling of certain products. In such cases, a group of persons is selected to receive the products at no cost (or very cheap) in exchange of giving publicity and promoting the sales as much as possible. Since every new individual that buys or acquires the products may act as an influencer (kind of "snow ball" effect), the objective is to reduce the costs by minimizing the group of individuals receiving the free products while guaranteeing the whole network (or big part of it) will be influenced.

Recently, the spread of "fake news" had reached receptive audiences in some communities, resulting in disinformation with direct impact in politics, public opinion and the population believes [13]. Solutions of the target set selection problem may help to mitigate the misinformation effect, by recognizing a group of individuals that will help to spread to the network accurate news and "good practices" for information consumption. Some other possible applications of target set selection problems occur in combating crime, by identifying informers and obtaining information on criminal organizations, in medicine by studying the evolution of contagious diseases, or in bioinformatics by regulating the behavior of bacterial consortia.

The work of Domingos and Richardson [10], [16] motivated Kempe et al. [12] to introduce a definition for a target set selection as a combinatorial optimization problem and, since then, several proofs of NP-hardness and inapproximability were given for the problem variants [5], [6], [8], [11]. The computational difficulty justifies a continuous development of different approaches as mathematical formulations and bounds proofs [2], approximation algorithms [12], heuristic or metaheuristic methods [9], [10], [16] and exact techniques in polynomial time for special cases or exponentially for general cases [3], [7].

In this work we study and propose the maximum effort-reward GAP Target Set Selection problem (MAX-GAP-TSS), a new NP-hard version of the Target Set Selection problem. We also propose a binary linear programming formulation and an evolutionary algorithm which uses our model to exactly solve sub-problems in the crossover and mutation operations, and we test our approach on large instances from the literature,

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, the Rio Grande do Sul Research Foundation - FAPERGS (grant 19/25510001906-8 PqG), the São Paulo Research Foundation - FAPESP (grant 2018/03819–4) and CNPq (grant 312206/2015-1).

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

based in real-world datasets.

The rest of the paper is organized as follows. In section II, we introduce and formally define the MAX-GAP-TSS and, in section III, we demonstrate the problem is NP-hard. In section IV, we propose a binary linear programming formulation to the problem, which is used in section V, to design an evolutionary algorithm hybridized with the mathematical formulation. The computational experiments are presented in section VI and, in section VII, we give final remarks and future woks.

II. Problem definition

Target set selection problems model propagation processes in social, biological and computer networks based on interactions of nodes. These problems receive as input a graph or a digraph where the nodes may have one of two states: activated or not activated. Many studies consider that once a node is activated it never returns to a not activated state and each node, say u, has associated a positive activation threshold value $\tau(u)$. So, a non-activated node u becomes activated iff it has at least $\tau(u)$ already activated neighbors [2], [5], [6], [9], [11]. The propagation process from an initial set A of activated nodes, consists of activating all the non-activated nodes that become activated from A, repeating the process from the new set of activated nodes until no new nodes can become activated. Since the activated nodes never return to a non-activated state, then each new set of activated nodes contains the previous set and, if the process is repeated k times, it will generate the following sequence of activated sets of nodes, being A^* the **result of the propagation process** from A:

$$A = A_0 \subset A_1 \subset A_2 \subset \cdots \subset A_k = A^*.$$

The scheme above, termed **adjacency-threshold propagation scheme**, assumes for each node all of its neighbors have the same influence to activate the node, which is unrealistic in diverse scenarios where there exist individuals with different values of influences over the same person. For that reason, we propose a more general strategy to model influence propagation: for any two nodes u and v, the influence of u over v is given by $\psi(u,v) \in [0,1]$ and, a non-activated node v becomes activated iff the influence sum of the already activated nodes over v is at least 1 (i.e., given the set A of activated nodes, v becomes activated iff $\sum_{u\in A}\psi(u,v) \geq 1$). We denote this scheme by **weighted-influence propagation scheme** and the following fact shows that the previous scheme is easily modeled by ours.

Fact 1: Given a graph (or digraph) $G = (V, E)$, a positive threshold function $\tau : V \to \mathbb{Q}^+$ over the nodes and an initially activated set of nodes $A \subseteq V$, the activated sets sequence generated by a propagation process from A using the adjacency-threshold propagation scheme is exactly the same sequence generated when using the weighted-influence propagation scheme with the influence function over every pair of nodes $u, v \in V$ defined as:

$$\psi(u,v) = \begin{cases} \frac{1}{\tau(v)}, & \text{if } \langle u,v\rangle \in E \\ 0, & \text{otherwise.} \end{cases}$$

Proof: First, we show that for any set of activated nodes $A' \subseteq V$, the set of non-activated nodes that can become activated from A' using the adjacency-threshold propagation scheme is exactly the same that the one using the weighted-influence propagation scheme.

Consider a non-activated node u that becomes activated from the activated nodes in A' using the adjacency-threshold propagation scheme. By definition, u has at least $\tau(u)$ neighbors already activated in A', so if we denote by $A'(u)$ the subset of activated nodes of A' neighbors of u, we obtain:

$$\begin{aligned} \tau(u) &\leq |A'(u)| = \sum_{v\in A'(u)} 1 \\ 1 &\leq \sum_{v\in A'(u)} \frac{1}{\tau(u)} = \sum_{v\in A'(u)} \psi(v,u). \end{aligned}$$

Implying that u also becomes activated using the weighted-influence propagation scheme with the influence function ψ as defined in the fact. The proof in the other direction is analogously but read backwards.

Hence, any non-activated node u becomes activated from A' by using the adjacency-threshold propagation scheme iff becomes activated from A' by using the weighted-influence propagation scheme with the influence function ψ. Then, we conclude that the sets of non-activated nodes that become activated from A' with any of both propagation schemes are exactly the same. So, given any initially activated set of nodes A, the sequence of activated sets generated in a propagation process from A is also the same when using any of the schemes. Thus, the proof is completed. □

Fact 1 shows that the weighted-influence propagation scheme is able to model the same situations as the adjacency-threshold propagation scheme. Furthermore, by setting different values to the influence function ψ we are able to model the propagation process of a larger variety of scenarios in which one individual may be influenced with different weights by the others (e.g., followers of celebrities or religious acolyte may be more influenced by the celebrity or the priest, respectively, than by another individuals of their network).

Besides considering a more general propagation scheme, we also consider that each node u requires a non-negative effort value to be initially activated, say $\alpha(u)$, and provides a non-negative reward value if activated at some point of the propagation process, say $\beta(u)$. These values are set to model the diversity of scenarios where different individuals require different costs to promote or use some product and also provide different profits after the product acquisition (e.g., a famous person may require a higher price to show the usage of a product than a non-famous person and, a company owner who buys a product to be used by all the employees may provide more profit than an individual buying the product for personal use).

Regarding the problem objective, some studies were done in order to activate the whole graph while minimizing the number of initially activated nodes, which may be translated

to minimize the effort in order to guarantee a lower bound reward value [5]–[7], [9], [11]. In contrast, other works were done in order to maximize the number of activated nodes, while the initially activated set cardinality is bounded by some threshold, that is to maximize the reward guaranteeing an upper bound effort value [8], [12]. Joining both cases, what the problem seeks is to minimize the effort while maximizing the reward, then the problem could be seen as a bi-objective problem. Since in many cases the effort and reward values may be expressed in the same terms (e.g., money or energy), our approach is to unify both objectives in one, seeking for the profit maximization which is given by the reward of the activated nodes at the end of the propagation process minus the effort to activate the nodes at the beginning of the process, i.e., we want to maximize the gap between the final reward and the initial effort.

Problem 1: Maximum effort-reward GAP Target Set Selection problem (MAX-GAP-TSS)

Input: A tuple $\langle V, \psi, \alpha, \beta \rangle$, where:

- V is a finite non-empty set of nodes,
- $\psi : V \times V \rightarrow [0, 1]$ is an influence function over each pair of nodes,
- $\alpha : V \rightarrow \mathbb{Q}^+$ is a non-negative effort function over the nodes,
- $\beta : V \rightarrow \mathbb{Q}^+$ is a non-negative reward function over the nodes.

Output: A set of initially activated nodes $A \subseteq V$ that maximizes the overall activated reward: the sum of the activated nodes' rewards at the end of the propagation process minus the sum of the initially activated nodes effort (i.e., maximizes $\sum_{u \in A^*} \beta(u) - \sum_{u \in A} \alpha(u)$, where A^* is the result of the propagation process from A with weighted-influence propagation scheme).

To the best of our knowledge, we are the first to propose this version of the Target Set Selection problem. For that reason, in next section we discuss over the complexity of solving MAX-GAP-TSS.

III. NP-HARDNESS

In order to prove MAX-GAP-TSS belongs to the NP-hard computational complexity class, we propose a polynomial transformation from another version of the Target Set Selection problem, which was proved to be NP-hard [5], [11] and we define as follows:

Problem 2: Minimum Target Set Selection problem (MIN-TSS)

Input: A tuple $\langle G = (V, E), \tau \rangle$, where:

- G is a graph with finite non-empty set of nodes V and set of edges E,
- $\tau : V \rightarrow \mathbb{N}^+$ is a positive activation threshold function over the nodes.

Output: A minimum cardinality set of initially activated nodes $A \subseteq V$, that activates all the nodes of the graph (i.e., if A^* is the result of the propagation process from A with adjacency-threshold propagation scheme, then $A^* = V$ and A has minimum cardinality).

Now, we prove the following theorem:

Theorem 1: MAX-GAP-TSS is NP-hard.

Proof: Consider an instance $I = \langle G = (V, E), \tau \rangle$ of the MIN-TSS. From this instance we construct the following instance of the MAX-GAP-TSS:

$$I' = \left\langle V, \psi(u,v) = \begin{cases} \frac{1}{\tau(v)}, & \text{if } \langle u, v \rangle \in E \\ 0, & \text{otherwise} \end{cases}, \alpha = 1, \beta = 2 \right\rangle$$

Instance I' has the same set of nodes as instance I, the influence function is the same as defined by Fact 1 and, for all the nodes, the effort and reward functions are, respectively, the constant values 1 and 2. Notice that I' size is $\mathcal{O}\left(|V|^2\right)$ and also I' can be constructed from I in $\mathcal{O}\left(|V|^2\right)$ time.

First, we prove that any optimal solution for MAX-GAP-TSS with instance I' activates all the nodes at the end of the propagation process. In that sense, consider A to be an optimal solution for MAX-GAP-TSS with instance I', A^* to be the result of the propagation process from A and also, that there exists a node $u \in V$ such that $u \notin A^*$ (i.e., u is not activated at the end of the propagation process from A). Then, the objective value of A is:

$$obj(A) = \sum_{v \in A^*} \beta(v) - \sum_{v \in A} \alpha(v) = 2\,|A^*| - |A|\,.$$

If we add u in A, then A^* will have at least one more node (u) and the objective value of the new solution ($A \cup \{u\}$) results:

$$obj\,(A \cup \{u\}) \geq 2\,|A^*| - |A| + \beta(u) - \alpha(u) = 2\,|A^*| - |A| + 1.$$

Then, $obj(A \cup \{u\}) > obj(A)$ and since A is optimal, there should not exist any solution with greater objective value, implying that such node u does not exist and that any optimal solution for MAX-GAP-TSS with instance I' activates all the nodes at the end of the propagation process. Moreover, V is the result of the propagation process from any optimal solution A for MAX-GAP-TSS with instance I' and objective value:

$$obj(A) = 2\,|V| - |A|\,.$$

Thus, to maximize the objective value of A is equivalent to minimize the cardinality of A, which is the objective value of MIN-TSS with instance I. Since Fact 1 shows that both propagation schemes activate the same sets of nodes, then an optimal solution A of MAX-GAP-TSS with instance I' activates all the nodes at the end of the propagation process and also has minimum cardinality, so A is also optimal for MIN-TSS with instance I. In the other direction, any optimal solution of MIN-TSS with instance I is a minimum cardinality solution that activates all the nodes at the end of the propagation process, so it is also an optimal solution for MAX-GAP-TSS with instance I'.

Finally, since the transformation complexity from I to I' is $\mathcal{O}\left(|V|^2\right)$ time, we conclude there exists a polynomial time reduction from MIN-TSS to MAX-GAP-TSS and, since MIN-TSS is NP-hard [5], [11], then MAX-GAP-TSS is also NP-hard. Therefore, the proof is completed. □

IV. Mathematical formulation

We propose a binary linear programming model to be used in the construction of different approaches to solve the MAX-GAP-TSS. In particular, we use our formulation to exactly solve small instances as part of the evolutionary algorithm we present in the next section.

In the direction of formalizing our formulation, from now on consider we are dealing with an instance $I = \langle V, \psi, \alpha, \beta \rangle$ of the MAX-GAP-TSS.

First, we define the variables of the formulation. We considered three different types of binary variables. The first type of binary variables are associated with each node $u \in V$, taking value 1 if the node belongs to the initially activated set A:

$$x_u = \begin{cases} 1, & \text{if } u \text{ is initially activated} \\ 0, & \text{otherwise.} \end{cases}$$

The second type of binary variables are also associated with each node $u \in V$, taking value 1 if the node is activated at some point of the propagation process from the initially activated nodes:

$$y_u = \begin{cases} 1, & \text{if } u \text{ is activated} \\ 0, & \text{otherwise.} \end{cases}$$

The last type of binary variables are associated with each pair of different nodes $u, v \in V$, taking value 1 if u is activated before v in the propagation process:

$$z_{uv} = \begin{cases} 1, & \text{if } u \text{ is activated before } v \\ 0, & \text{otherwise.} \end{cases}$$

Notice that variables of types x_u and y_u give us, respectively, the nodes whose effort and reward must be considered in the objective function, so we can express the objective function as:

$$\max \sum_{u \in V} \beta(u) y_u - \sum_{u \in V} \alpha(u) x_u.$$

Now, we need to determine the formulation constraints. Observe that the variables of type z_{uv} determine the activation order. Then some constraints are needed to guarantee such order is not violated. One of the order constraints must guarantee that if node u is activated before v, then v cannot be activated before u (i.e., only one of z_{uv} or z_{vu} can be equals to 1), the other constraint must guarantee that if node u is activated before v and v is activated before w, then u also must be activated before w (i.e., if $z_{uv} = z_{vw} = 1$, then $z_{uw} = 1$). Both constraints are easily modeled by the following inequalities:

$$z_{uv} + z_{vu} \leq 1 \qquad \forall u, v \in V$$
$$z_{uv} + z_{vw} - z_{uw} \leq 1 \qquad \forall u, v, w \in V.$$

Since the variables of types z_{uv} and x_u indicate an activated state, then we must guarantee the node $u \in V$ is actually activated whenever $x_u = 1$ or $z_{uv} = 1$ for any other node $v \in V$. Constraints that consider those conditions are:

$$z_{uv} \leq y_u \qquad \forall u, v \in V$$
$$x_u \leq y_u \qquad \forall u \in V.$$

If we consider the weight-influence propagation scheme, to a node u be activated (i.e., $y_u = 1$), it is required that the node belongs to the initially activated set (i.e., $x_u = 1$) or the sum of the preceding activated nodes influences over u should be greater or equal to 1 (i.e., $\sum_{v \in V} \psi(v, u) z_{vu} \geq 1$). Such conditions can be expressed by the following inequalities:

$$y_u \leq x_u + \sum_{v \in V} \psi(v, u) z_{vu} \qquad \forall u \in V.$$

We also include constraints to enforce the activation of the node u (i.e., $y_u = 1$) if the sum of the influences of the previous activated nodes over u is greater or equal to 1 (i.e., $\sum_{v \in V} \psi(v, u) z_{vu} \geq 1$):

$$\sum_{v \in V} \psi(v, u) \left(z_{vu} - y_u \right) < 1 \qquad \forall u \in V.$$

The above inequalities are strict, implying that the set of solutions they describe is not convex. A strategy to avoid strict inequalities is to include a positive constant, say ϵ, smaller than 1 (e.g, 10^{-3}):

$$\sum_{v \in V} \psi(v, u) \left(z_{vu} - y_u \right) \leq 1 - \epsilon \qquad \forall u \in V.$$

The last set of constraints we define are to avoid some symmetries and they force that any activated node u (i.e., $y_u = 1$) must be activated before or after any other node v (i.e., $z_{uv} = 1$ or $z_{vu} = 1$):

$$y_u \leq z_{uv} + z_{vu} \qquad \forall u, v \in V.$$

Finally, we propose the following binary linear formulation with $\mathcal{O}\left(|V|^2\right)$ variables and $\mathcal{O}\left(|V|^3\right)$ constraints:

Model 1: Binary linear program for instance $\langle V, \psi, \alpha, \beta \rangle$ of MAX-GAP-TSS:

$$\max \sum_{u \in V} \beta(u) y_u - \sum_{u \in V} \alpha(u) x_u$$

s.t.

$$z_{uv} + z_{vu} \leq 1 \qquad \forall u, v \in V$$
$$z_{uv} + z_{vw} - z_{uw} \leq 1 \qquad \forall u, v, w \in V$$
$$z_{uv} - y_u \leq 0 \qquad \forall u, v \in V$$
$$x_u - y_u \leq 0 \qquad \forall u \in V$$
$$y_u - x_u - \sum_{v \in V} \psi(v, u) z_{vu} \leq 0 \qquad \forall u \in V$$
$$\sum_{v \in V} \psi(v, u) z_{vu} - \sum_{v \in V} \psi(v, u) y_u \leq 1 - \epsilon \qquad \forall u \in V$$
$$y_u - z_{uv} - z_{vu} \leq 0 \qquad \forall u, v \in V$$
$$x_u, y_u, z_{uv} \in \{0, 1\} \qquad \forall u, v \in V.$$

By changing the objective function to $\min \sum_{u \in V} \alpha(u) x_u$ and adding the constraint $\sum_{u \in V} \beta(u) y_u \geq K$, for a given positive constant K, we obtain a new formulation for a generalized MIN-TSS. By considering as objective function $\max \sum_{u \in V} \beta(u) y_u$ and adding the constraint $\sum_{u \in V} \alpha(u) x_u \leq K$, we obtain a new formulation for a generalization of another NP-hard Target Set Selection problem [8], [12].

V. EVOLUTIONARY ALGORITHM

Our proposal is to hybridize a classical evolutionary algorithm [15] with other approaches as mathematical programming and local search. Evolutionary algorithms have been able to produce satisfactory solutions for several complex and difficult problems [4], [17], [19], and improvements could been achieved by solving sub-problems of large instances exactly, using those results to obtain "better" solutions. With that objective, during the crossover and mutation operations, we leave at most K free nodes (fixing the rest of the nodes in the solution or out of the solution) and we calculate an exact solution to the sub-problem with the K free nodes. The exact solutions of the generated sub-problems are computed with a given solver and the sub-problem formulation using Model 1. Finally, each solution of the larger instances pass through a local search process, in order to improve the quality. Algorithm 1 shows our evolutionary approach and we describe it in the sequence.

```
    Input  : ⟨V, ψ, α, β⟩, instance of the MAX-GAP-TSS.
    Output : Ā, solution of the MAX-GAP-TSS with instance ⟨V, ψ, α, β⟩.
1   begin
2       Ā ← ∅
3       S_0 ← ∅
4       while |S_0| < POOL_SIZE do
5           A ←GENERATE(V, ψ, α, β)
6           S_0 ← S_0 ∪ {A}
7           if obj(A) > obj(Ā) then
8               Ā ← A
9           end
10      end
11      i ← 0
12      j ← 0
13      while i ≤MAX_ITERATIONS and j ≤MAX_NO_UPDATE do
14          for each A ∈ S_i do
15              fitness[A] ← ρ × obj(A) + θ ×HAMMING(A, S_i)
16          end
17          S_{i+1} ← ∅
18          for l ← 1 to CROSSOVER_NUMBER do
19              A_1, A_2 ← pair of different solutions from S_i, randomly selected according to the
                  fitness value
20              A ←CROSSOVER(A_1, A_2, V, ψ, α, β, K)
21              S_{i+1} ← S_{i+1} ∪ {A}
22              if obj(A) > obj(Ā) then
23                  Ā ← A
24                  j ← 0
25              end
26          end
27          S_{i+1} ← S_i ∪ S_{i+1}
28          S' ← ∅
29          for l ← 1 to MUTATION_NUMBER do
30              A ← random solution from S_{i+1}
31              A ← MUTATION(A, V, ψ, α, β, K)
32              S' ← S' ∪ {A}
33              if obj(A) > obj(Ā) then
34                  Ā ← A
35                  j ← 0
36              end
37          end
38          S' ← S_{i+1} ∪ S'
39          S'' ← μ×POOL_SIZE solutions from S' with greater objective value
40          S_{i+1} ← (1 − μ) ×POOL_SIZE solutions from S' with greater HAMMING distance
              to S''
41          S_{i+1} ← S_{i+1} ∪ S''
42          i ← i + 1
43          j ← j + 1
44      end
45      return Ā
46  end
```

Algorithm 1: Evolutionary algorithm for MAX-GAP-TSS.

Algorithm 1 requires the specification of some parameters, that can be received as part of the input (besides the instance of MAX-GAP-TSS) or defined in the algorithm body. Those parameters are:

- `POOL_SIZE`, to indicate the size of the population at the beginning/ending of each iteration;
- `MAX_ITERATIONS` and `MAX_NO_UPDATE`, to indicate, respectively, the maximum number of generations and the maximum number of generations without updating the best solution found;
- `CROSSOVER_NUMBER`, to indicate the number of new solutions generated by crossover operations at each generation;
- `MUTATION_NUMBER`, to indicate the number of new solutions generated by the mutation operation at each generation;
- ρ and θ, weights of the objective value and the average distance of a solution to the pool, used for fitness evaluation;
- K, to indicate the maximum number of free nodes for the definition of the sub-problems to exactly be solved;
- μ, to indicate the percentage of the solutions with best objective value to be part of the next generation, being $(1-\mu)$ the percentage of the solution selected to the next generation based on a distance criteria.

Algorithm 1 begins with an empty solution $\bar{A}$ at line 2, which is updated in lines 7-9, 22-25 and 33-36 every time e better solution is obtained (i.e., a solution with greater objective value), being the last value of $\bar{A}$ the algorithm output. The first generation of solutions, denoted S_0, is initialized as an empty collection in line 3 and `POOL_SIZE` randomly generated solutions are added between lines 4 and 10. Then, to reference the current number of generations and the number of generations without updating the best solution found, in lines 11 and 12 are initialized counter variables i and j, whose values are updated between lines 42 and 43 of the algorithm main loop (lines 13 to 44) and whenever a new best solution is found, variable j is reset (lines 24 and 35). At the beginning of every iteration of the main loop, between lines 14 and 16, each solution receives a fitness value, composed by a weighted sum of the objective function value and the average Hamming distance of the solution to the rest of the pool. Then, between lines 18 and 26 new solutions are constructed by a crossover operator over two parents selected from the pool. The parents are randomly selected in line 19 by applying a probabilistic distribution, which depends on the previously calculated fitness, where the probability of the solution A be selected as parent is given by $\frac{fitness[A]}{\sum_{A' \in S_i} fitness[A']}$ (i.e., the fitness of A divided by the fitness sum of every solution of the pool). Thus, solutions with greater fitness values have greater probabilities of being parents of new solutions. In line 27, we also include as possible candidates for the next generation all the solutions of the current one, and between lines 29 and 37 we apply a mutation operator to randomly selected solutions that are also added as candidates for the next generation. Finally, between lines 38 and 41, we select the $\mu\times$POOL_SIZE solutions with greater objective function values to the next generation, and, in order to provide some diversity, we complete the new pool with the $(1-\mu)\times$POOL_SIZE solutions farther from the already selected solutions.

Now we describe the procedures used by Algorithm 1. First, we have the `GENERATE` function which constructs a random initial solution. The `GENERATE` function pseudo-code is given

by Algorithm 2.

```
    Input  : ⟨V, ψ, α, β⟩, instance of the MAX-GAP-TSS.
    Output : A, random initial solution.
1   begin
2       A ← ∅
3       A* ← ∅
4       n ← random integer between 0 and |V|
5       while |A*| < n do
6           u ← random node from V \ A*
7           A ← A ∪ {u}
8           A* ← propagation process result from A
9       end
10      A ← LOCAL_SEARCH(A)
11      return A
12  end
```

Algorithm 2: Algorithm to generate an initial solution.

Observe that Algorithm 2 begins with two empty sets A and A^*, for the solution being constructed and for the result of the propagation process from that solution (lines 2 and 3). Then, in line 4, a random integer n is uniformly generated in the interval $[0, |V|]$, to determine a minimum number of nodes that must be activated as result of the propagation process from A. At each iteration of the loop from line 5 to 9, a node is randomly selected with an uniform distribution among the not activated nodes resulting of the propagation process from A (i.e., selected from $V \setminus A^*$). Then, the selected node is added to the solution A and the result of the propagation process from A is updated. After constructing a random solution A that guarantees the activation of at least n nodes, the algorithm returns an improved solution by executing a local search procedure from A, where two solutions are neighbors if they differ in at most one node. Formally, we defined the neighborhood of A as:

$$N(A) = \{A' : A' \subseteq V \text{ and } |A - A'| \leq 1 \text{ and } |A' - A| \leq 1\}$$

Another procedure Algorithm 1 uses is the CROSSOVER, described by Algorithm 3.

```
    Input  : ⟨V, ψ, α, β⟩, instance of the MAX-GAP-TSS; A1, A2, parents of the new solution;
             K, number of free nodes.
    Output : A, solution generated by the crossover of A1 and A2.
1   begin
2       A ← A1 ∩ A2
3       A* ← propagation process result from A
4       while |(A1 ∪ A2) \ A*| > K do
5           u ← random node from (A1 ∪ A2) \ A*
6           A ← A ∪ {u}
7           A* ← propagation process result from A
8       end
9       if 1 ≤ |V \ A*| ≤ K then
10          A ← SOLVER(V, ψ, α, β, A, (A1 ∪ A2) \ A*)
11      end
12      A ← LOCAL_SEARCH(A)
13      return A
14  end
```

Algorithm 3: Crossover operator algorithm.

Besides the instance of MAX-GAP-TSS, Algorithm 3 receives as input two parent solutions A_1 and A_2, and the positive integer K. The new solution begins as the intersection of the parents (line 2) and while there exist more than K initially activated nodes of the parents that are not activated at the end of the propagation process from the new solution A, one of those nodes is randomly selected and added to A (lines 4-8). Thus, at line 10, there are at most K initially activated nodes of the parents that are not activated in the propagation process from A, we select those nodes to define the set of free nodes used by the exact solver (denote such set by A'). Then, at line 10, we call a solver (e.g., CPLEX, Gurobi) to exactly compute an optimal solution for the sub-problem defined by the instance $\langle V, \psi, \alpha, \beta \rangle$, the solution being constructed A and the set of free nodes A'. For the sub-problem formulation we construct Model 1 from the instance $\langle V, \psi, \alpha, \beta \rangle$, we obtain the propagation sequence from A: $A = A_0 \subset A_1 \subset A_2 \subset \cdots \subset A_m = A^*$, and we fix the following variables:

- For each node u in A (solution being constructed, which is the set of initially activated nodes) we fix variable x_u equals to 1;
- For each node u in A^* (result of the propagation process from A) we fix variable y_u equals to 1;
- For every $0 \leq i < j \leq m$, we fix each z_{uv} equals to 1, where $u \in A_i$ and $v \in A_j$ (every node in A_i was activated before any node of A_j);
- For every $0 \leq i \leq m$, we sort A_i and set z_{uv} equals 1, for each pair of nodes $u, v \in A_i$ where u is before v in the sorting (nodes activated in the same step can be in any activation order);
- For each pair of nodes $u \in A^*$ and $v \notin A^*$, we fix variable z_{uv} equals to 1 (nodes in A^* are already activated, so the activation moment could be considered before than any node out of A^*);
- For each z_{uv} fixed to 1, we fix z_{vu} equals to 0 (if node u is activated before v, then v cannot be activated before u);
- For each node u in $A^* \setminus A$, we fix the variable x_u equals to 0 (nodes that are activated during the propagation process from A have no need to be initially activated);
- For each node u in $V \setminus (A \cup A')$, we fix the variable x_u equals to 0 (only the free nodes can be added to the initially activated set).

After fixing those values, the number of variables of Model 1 is reduced to at most $(K \times |V| + K + |V \setminus A^*|)$, from the originally $\left(|V|^2 + 2 \times |V|\right)$ variables, which for small values of K implies in almost the square root of the original number of variables. Furthermore, before calling the solver, we check at line 9 if the number of variables of the reduced model is small enough (depending only on the constant K). Thus, the sub-problem generated is much simpler and the specific solvers could be able to compute an exact solution in "reasonable time". Succeeding the solver call, Algorithm 3 returns an improved solution obtained by a local search procedure (similar to the one of Algorithm 2) over the resulting solution from the solver.

The last procedure Algorithm 1 uses is the MUTATION operation. That operator receives an instance of the MAX-GAP-TSS, a solution A to be mutated and the integer K. The idea is to select K random nodes from the set of nodes A' to leave as free nodes for the sub-problem. An uniformly random integer number ℓ is generated between 0 and K , to select ℓ free nodes from A to A' and $K - \ell$ free nodes from V to A'. Then, we remove from the solution A the selected free nodes and, similarly as in Algorithm 3, we exactly solve the

sub-problem obtained by the instance, the initially activated nodes A and the free nodes A'. The solution of the solver is also submitted to the same local search procedure and returned to the main algorithm. The pseudo-code for the MUTATION procedure is given by Algorithm 4.

```
Input  : ⟨V, ψ, α, β⟩, instance of the MAX-GAP-TSS; A, solution to be mutated; K, number
         of free nodes.
Output : A, mutated solution.
begin
    ℓ ← random integer number between 0 and K
    A_1 ← ℓ random nodes from A
    A_2 ← K − ℓ random nodes from V
    A' ← A_1 ∪ A_2
    A ← A \ A'
    A* ← propagation process result from A
    if 1 ≤ |V \ A*| ≤ K then
        A ← SOLVER(V, ψ, α, β, A, A' \ A*)
    end
    A ← LOCAL_SEARCH(A)
    return A
end
```

Algorithm 4: Mutation operator algorithm.

VI. Experiments

With the objective of analyzing the applicability of our evolutionary approach, this section presents computational experiments over several real-world datasets. The larger datasets were obtained from *Standford Large Dataset Collection* [14], while the smaller datasets were obtained from *UCINET IV DATASETS* [1]. All the selected datasets (larger and smaller) are described by Table I.

Name	Symmetrical	# of nodes	# of influence arcs
Krackhardt office css [1]	No	21	278
Zachary Karate club [1]	Yes	34	155
B&K office [1]	No	40	1.560
B&K fraternity [1]	No	58	3.306
Gagnon&Macrae prison [1]	No	67	182
ego-Facebook [14]	Yes	4.039	88.234
CA-HepPh [14]	Yes	12.008	118.521
CA-AstroPh [14]	Yes	18.772	198.110
CA-CondMat [14]	Yes	23.133	93.497
Cit-HepTh [14]	No	27.770	352.807

TABLE I
SELECTED DATASETS FROM [14] AND [1].

Some of the datasets were also used in a previous work for the MIN-TSS [9] and, since the existing datasets do not provide information to compute the influence, reward nor effort values, we constructed instances for the MIN-TSS following the threshold criteria proposed by the authors of [9]:

- *Constant threshold*, each node u receives as threshold the minimum between $d(u)$ and a constant value, where $d(u)$ is the node degree. For each dataset and each value of τ in $\{2, 4, 6, 8, 10\}$, an instance was constructed.
- *Proportional thresholds*, each node receives as threshold $\eta \times d(u)$, for a given constant η. For each dataset and each value of η in $\{0.1, 0.2, 0.3, 0.4, 0.5\}$, an instance was constructed. For $\eta = 0.5$ the nodes are activated under majority threshold [2].
- *Random threshold*, each node u receives as threshold activation value a random integer number in $[1, d(u)]$.

The above description is for instances of MIN-TSS and, to obtain instances for the MAX-GAP-TSS, we applied the reduction described in section III.

In order to execute the tests, we implemented our algorithms in the C++ programming language using the compiler g++ 7.4.0, being the algorithms parameters: POOL_SIZE = 10, CROSSOVER_NUMBER = 10, MUTATION_NUMBER = 10, MAX_ITERATIONS = 300, MAX_NO_ITERATIONS = 50, $\rho = 1$, $\theta = 1$, $K = 10$, $\mu = 1$. We used the Mersenne Twister algorithm [18] with seed value 211612 for the pseudo-random numbers generation and the `gettimeofday()` function to obtain the CPU times. For each instance, our implementations were executed in a processor Intel Xeon(R) CPU E5-1620v2 with 4 cores of 3,70GHz each, and 16 GB of RAM, under Ubuntu Linux 18.04.3 LTS 64 bits.

In the direction of analyzing the quality of the solutions, we used the CPLEX to solve the instances constructed from the small datasets selected from [1]. For all these instances, our algorithms were able to find solutions in a very short time, with the same objective values as the solutions found by the CPLEX. Hence, our approach found almost optimal solutions for all the small instances and, for the instances the CPLEX was able to prove optimality, our algorithms found also optimal solutions. Table II gives the results of the small instances experiments. For the large instances from [14], the CPLEX was not able to find feasible solutions, however our approach found interesting solutions where the cardinality of the initial set that activates the whole network was at most an order of magnitude smaller than the network size. Table III resumes the experiments for the large set of instances.

VII. Conclusions

We introduced a new version of the Target Set Selection problem, suitable for several practical applications. We proved the problem belongs to the NP-hard class and also we presented a new binary linear formulation, which can be easily adapted to other Target Set Selection problems. Besides that, we designed an evolutionary algorithm hybridized with our mathematical formulation and we run tests over a benchmark of instances based on real-world data-sets. Our results seem satisfactory and very competitive, furthermore, for several of the tested instances, we obtained "almost" optimal solutions in a "small" amount of time. Future works will consider to hybridize evolutionary algorithms with primal-dual formulations, Lagrangian relaxation or results from polyhedral and approximation theories. Also, despite the instances were constructed from real-world datasets, some parameters (influence, reward and effort) were artificial since they were not considered by the existing datasets. Thus, another line of work would be to create and make available more realistic and meaningful datasets for the MAX-GAP-TSS.

References

[1] UCINET IV DATASETS. http://vlado.fmf.uni-lj.si/pub/networks/data/UciNet/UciData.htm. Accessed: 2020-05-09.

[2] E. Ackerman, O. Ben-Zwi, and G. Wolfovitz. Combinatorial model and bounds for target set selection. *Theoretical Computer Science*, 411(44):4017 – 4022, 2010.

[3] I. Bliznets and D. Sagunov. Solving Target Set Selection with Bounded Thresholds Faster than 2^n. In *13th International Symposium on Parameterized and Exact Computation (IPEC 2018)*, volume 115 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 22:1–22:14, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

Instances			Evolutionary Algorithm							
			Information of the best solution found					CPLEX	Overall	
			Value	$\|A\|$	$\|A^*\|$	Iter.	Seconds	Calls	Iter.	Seconds
Krackhardt office css	constant	2	40	2	21	0	0.0000	0	50	0.0020
		4	38	4	21	0	0.0000	0	50	0.0026
		6	36	6	21	0	0.0000	0	50	0.0030
		8	34	8	21	0	0.0000	2	50	0.0097
		10	32	10	21	0	0.0001	0	50	0.0034
	proportional	0.1	41	1	21	0	0.0000	0	50	0.0017
		0.2	40	2	21	0	0.0000	0	50	0.0021
		0.3	40	2	21	0	0.0001	0	50	0.0039
		0.4	39	3	21	0	0.0002	0	50	0.0056
		0.5	38	4	21	0	0.0001	0	50	0.0034
	random		40*	2	21	0	0.0000	210	50	3.0441
Zachary Karate club	constant	2	66*	2	34	0	0.0001	0	50	0.0031
		4	57*	11	34	1	0.1763	230	51	2.5693
		6	54*	14	34	1	0.0423	115	51	0.5228
		8	52*	16	34	35	0.2339	96	85	0.4949
		10	51*	17	34	1	0.0924	133	51	1.0042
	proportional	0.1	67*	1	34	0	0.0000	0	50	0.0023
		0.2	67*	1	34	0	0.0000	0	50	0.0025
		0.3	67*	1	34	0	0.0002	0	50	0.0048
		0.4	65	3	34	0	0.0002	7	50	0.0356
		0.5	64	4	34	0	0.0001	7	50	0.0364
	random		64*	4	34	1	0.0003	2	51	0.0162
B&K office	constant	2	78	2	40	0	0.0001	0	50	0.0055
		4	76	4	40	0	0.0001	0	50	0.0070
		6	74	6	40	0	0.0001	0	50	0.0071
		8	72	8	40	0	0.0001	0	50	0.0078
		10	70	10	40	0	0.0001	0	50	0.0089
	proportional	0.1	76	4	40	0	0.0001	0	50	0.0091
		0.2	72	8	40	0	0.0001	0	50	0.0086
		0.3	68	12	40	0	0.0001	0	50	0.0162
		0.4	65	15	40	0	0.0001	0	50	0.0100
		0.5	63	17	40	0	0.0001	0	50	0.0114
	random		77	3	40	0	0.0001	0	50	0.0067
B&K fraternity	constant	2	114	2	58	0	0.0001	0	50	0.0141
		4	112	4	58	0	0.0001	0	50	0.0155
		6	110	6	58	0	0.0001	0	50	0.0169
		8	108	8	58	0	0.0002	0	50	0.0176
		10	106	10	58	0	0.0001	0	50	0.0118
	proportional	0.1	110	6	58	0	0.0001	0	50	0.0169
		0.2	104	12	58	0	0.0002	0	50	0.0186
		0.3	99	17	58	0	0.0002	0	50	0.0200
		0.4	96	20	58	0	0.0002	0	50	0.0204
		0.5	91	25	58	0	0.0002	0	50	0.0214
	random		114	2	58	0	0.0001	0	50	0.0101
Gagnon&Macrae prison	constant	2	120	14	67	0	0.0004	17	50	0.1968
		4	99	35	67	11	0.2543	174	61	0.9826
		6	97	37	67	12	0.3435	253	62	1.1223
		8	96	38	67	26	0.3243	330	76	1.0606
		10	96	38	67	11	0.3226	174	61	0.9906
	proportional	0.1	129	5	67	0	0.0002	0	50	0.0098
		0.2	129*	5	67	0	0.0002	254	50	2.3519
		0.3	128	6	67	0	0.0441	96	50	1.0640
		0.4	126	8	67	1	0.0141	129	51	0.7299
		0.5	126	8	67	0	0.0360	9	50	0.0886
	random		122*	12	67	8	0.0020	20	58	0.0570

TABLE II

EXPERIMENTS RESUME FOR SMALL INSTANCES FROM [1]. THE * INDICATES THAT THE SOLUTION WAS PROVED TO BE OPTIMAL.

Instances			Evolutionary Algorithm							
			Information of the best solution found					CPLEX	Overall	
			Value	$\|A\|$	$\|A^*\|$	Iter.	Seconds	Calls	Iter.	Seconds
ego-Facebook	constant	2	7870	208	4039	7	0.30	70	57	5.10
		4	7645	433	4039	46	6.10	45	96	12.22
		6	7415	663	4039	89	15.71	99	139	24.97
		8	7201	877	4039	74	16.45	31	124	26.70
		10	6997	1081	4039	167	46.49	102	217	61.89
	proportional	0.1	7930	148	4039	91	9.51	0	141	14.60
		0.2	7539	539	4039	299	56.35	1	300	56.35
		0.3	7087	991	4039	294	83.92	88	300	85.55
		0.4	6406	1672	4039	69	29.37	2	119	51.40
		0.5	5817	2261	4039	260	156.92	81	300	180.39
	random		6424	1654	4039	161	69.78	349	211	91.10
CA-HepPh	constant	2	22010	2006	12008	93	32.58	246	143	87.77
		4	18871	5145	12008	62	215.89	253	112	373.75
		6	17343	6673	12008	75	341.30	204	125	547.58
		8	16417	7599	12008	142	730.03	275	192	981.29
		10	15770	8246	12008	179	987.53	378	229	1271.33
	proportional	0.1	23686	330	12008	12	3.75	0	62	18.10
		0.2	23213	803	12008	169	150.47	47	219	190.99
		0.3	21367	2649	12008	291	755.66	16	300	780.56
		0.4	19166	4850	12008	286	1234.46	36	300	1296.90
		0.5	17970	6046	12008	236	1142.46	212	286	1370.10
	random		18081	5935	12008	252	618.10	422	300	829.26
CA-AstroPh	constant	2	35686	1858	18772	101	55.26	308	151	138.10
		4	32413	5131	18772	75	353.85	196	125	596.46
		6	30295	7249	18772	110	719.67	232	160	1046.67
		8	28839	8705	18772	82	677.47	133	132	1100.97
		10	27726	9818	18772	100	952.21	202	150	1450.01
	proportional	0.1	37219	325	18772	42	27.04	7	92	56.09
		0.2	36521	1023	18772	231	407.68	56	281	490.48
		0.3	34549	2995	18772	281	1264.19	24	300	1349.85
		0.4	30772	6772	18772	274	2933.47	6	300	3230.34
		0.5	27969	9575	18772	274	3607.12	137	300	3950.60
	random		28789	8755	18772	299	3049.41	807	300	3059.38
CA-CondMat	constant	2	42654	3612	23133	18	17.26	5	68	46.53
		4	36333	9933	23133	9	16.94	23	59	54.16
		6	32509	13757	23133	158	1232.59	253	208	1761.24
		8	30069	16197	23133	163	2082.80	285	213	2772.39
		10	28441	17825	23133	193	2737.02	395	243	3416.41
	proportional	0.1	45655	611	23133	67	59.84	104	117	100.19
		0.2	44520	1746	23133	217	735.66	68	267	894.67
		0.3	40523	5743	23133	197	2313.72	6	247	2894.48
		0.4	35688	10578	23133	298	5407.88	51	300	5446.20
		0.5	33085	13181	23133	296	5935.88	279	300	6013.20
	random		34246	12020	23133	243	3870.14	434	293	4744.98
Cit-HepTh	constant	2	50924	4616	27770	108	345.51	322	158	595.70
		4	48917	6623	27770	136	1280.98	561	186	1685.39
		6	48880	6660	27770	187	2156.70	434	237	2738.04
		8	47957	7583	27770	239	3181.69	253	289	3870.10
		10	46453	9087	27770	261	3681.16	445	300	4222.45
	proportional	0.1	50937	4603	27770	49	174.49	0	99	370.47
		0.2	50937	4603	27770	32	224.95	0	82	581.36
		0.3	50937	4603	27770	44	387.70	20	94	817.61
		0.4	50935	4605	27770	41	518.33	88	91	990.53
		0.5	50935	4605	27770	58	755.61	125	108	1311.57
	random		50923	4617	27770	130	1902.14	326	180	2571.71

TABLE III

EXPERIMENTS RESUME FOR LARGE INSTANCES FROM [1].

[4] M. R. Bonyadi and D. C. Reutens. Optimal-margin evolutionary classifier. *IEEE Transactions on Evolutionary Computation*, 23(5):885–898, 2019.

[5] C. C. Centeno, M. C. Dourado, L. D. Penso, D. Rautenbach, and J. L. Szwarcfiter. Irreversible conversion of graphs. *Theoretical Computer Science*, 412(29):3693 – 3700, 2011.

[6] N. Chen. On the approximability of influence in social networks. *SIAM Journal on Discrete Mathematics*, 23(3):16, 2009.

[7] C. Y. Chiang, L. H. Huang, B. J. Li, J. Wu, and H. G. Yeh. Some results on the target set selection problem. *Journal of Combinatorial Optimization*, 25(4):702–715, 2013.

[8] F. Cicalese, G. Cordasco, L. Gargano, M. Milanič, J. Peters, and U. Vaccaro. Spread of influence in weighted networks under time and budget constraints. *Theoretical Computer Science*, 586:40 – 58, 2015. Fun with Algorithms.

[9] G. Cordasco, L. Gargano, and A. A. Rescigno. On finding small sets that influence large networks. *Social Network Analysis and Mining*, 6(1):94, 2016.

[10] P. Domingos and M. Richardson. Mining the network value of customers. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, pages 57–66, New York, NY, USA, 2001. ACM.

[11] P. A. Dreyer and . S. Roberts. Irreversible k-threshold processes: Graph-theoretical threshold models of the spread of disease and of opinion. *Discrete Applied Mathematics*, 157(7):1615 – 1627, 2009.

[12] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, pages 137–146, New York, NY, USA, 2003. ACM.

[13] 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, and J. L. Zittrain. The science of fake news. *Science*, 359(6380):1094–1096, 2018.

[14] J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014. Accessed: 2020-05-09.

[15] A. Pétrowski and S. Ben-Hamida. *Evolutionary Algorithms*, volume 9. Wiley, 2017.

[16] M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 61–70, New York, NY, USA, 2002. ACM.

[17] S. Y. Huang and Y. Chen. Proving theorems by using evolutionary search with human involvement. In *2017 IEEE Congress on Evolutionary Computation (CEC)*, pages 1495–1502, 2017.

[18] M. Saito and M. Matsumoto. SIMD-oriented fast Mersenne Twister: a 128-bit pseudorandom number generator. *In Monte Carlo and Quasi-Monte Carlo Methods*, 64(2):607–622, 1993.

[19] A. Shirazi, J. Ceberio, and J. A. Lozano. Evolutionary algorithms to optimize low-thrust trajectory design in spacecraft orbital precession mission. In *2017 IEEE Congress on Evolutionary Computation (CEC)*, pages 1779–1786, 2017.