







# Minimizing the Cost of Leveraging Influencers in Social Networks: IP and CP Approaches<sup>\*</sup>

Felipe de C. Pereira<sup>1</sup>, Pedro J. de Rezende<sup>1</sup>, and Tallys Yunes<sup>2</sup>

<sup>1</sup> Institute of Computing, University of Campinas (UNICAMP), Campinas, SP, Brazil  
pereira\_felipe@ic.unicamp.br, pjr@unicamp.br

<sup>2</sup> Miami Herbert Business School, University of Miami, Coral Gables, FL, USA  
tallys@miami.edu

**Abstract.** In this paper, we introduce and study mathematical programming formulations for the Least Cost Directed Perfect Awareness Problem (LDPAP), an NP-hard optimization problem that arises in the context of influence marketing. In the LDPAP, we seek to identify influential members of a given social network that can disseminate a piece of information and trigger its propagation throughout the network. The objective is to minimize the cost of recruiting the initial spreaders while ensuring that the information reaches everyone. This problem has been previously modeled as two different integer programming formulations that were tested on a collection of 300 small synthetic instances. In this work, we propose two new integer programming models and three constraint programming formulations for the LDPAP. We also present pre-processing techniques capable of significantly reducing the sizes of these models. To investigate and compare the efficiency and effectiveness of our approaches, we perform a series of experiments using the existing small instances and a new publicly available benchmark of 14 large instances. Our findings yield new optimal solutions to 185 small instances that were previously unsolved, tripling the total number of instances with known optima. Regarding both small and large instances, our contributions include a comprehensive analysis of the experimental results and an evaluation of the performance of each formulation in distinct scenarios, further advancing our understanding of the LDPAP toward the design of exact approaches for the problem.

**Keywords:** Least Cost Directed Perfect Awareness Problem · Influence Marketing · Social Networks · Integer Programming · Constraint Programming.

---

<sup>\*</sup> Supported in part by grants from: *Santander Bank*, Brazil; *Brazilian National Council for Scientific and Technological Development* (CNPq), Brazil, #313329/2020-6, #314293/2023-0; *São Paulo Research Foundation* (FAPESP), Brazil, #2023/04318-7, #2023/14427-8; *Fund for Support to Teaching, Research and Outreach Activities* (FAEPEX), Brazil; *Coordination for the Improvement of Higher Education Personnel* (CAPES), Brazil – Finance Code 001.

## 1 Introduction

The phenomenon of influence spreading on social networks has been modeled in combinatorial optimization since the beginning of the century [11]. One of the most studied problems within this topic is the Target Set Selection Problem (TSSP) [5], whose introduction gave rise to IP formulations [1,20,22] and a collection of variants that deal with different scenarios [2].

In the original formulation of the TSSP, the objective is to select a set of individuals of minimum size (or minimum cost) capable of starting a widespread propagation of a piece of information (news, media, opinion, etc.). Typically, it is assumed that during a propagation, inactive people only start to actively pass on the information when they receive sufficient influence from their acquaintances.

A more realistic scenario is addressed in the Perfect Awareness Problem (PAP), which was proposed in [6] and also studied in [7,13,16]. In the PAP, an additional intermediary state (being aware) is introduced. In this problem, people are considered aware of the information as soon as they are influenced for the first time. The objective is to select a minimum set of seminal spreaders so that all individuals end up in the aware state at the end of the propagation.

In [15], the Least Cost Directed Perfect Awareness Problem (LDPAP) is introduced as an extension of the PAP to consider nonreciprocal relations observed in social networks like  $\mathbb{X}$  (formerly known as Twitter) and Instagram. The LDPAP also includes distinct degrees of influence between individuals (which neither TSSP nor PAP consider) as well as different costs for recruiting initial spreaders.

Although there are important theoretical and practical results regarding the LDPAP in the literature, which includes two integer programming (IP) formulations and a heuristic [15], there remains significant work to be done to solve the problem from an exact perspective. So far, only small instances can be solved to optimality. In this paper, we propose new formulations and exact techniques for the LDPAP that allow us to solve considerably larger instances.

### Our contributions

The main contributions of this paper for the LDPAP are:

- two novel IP formulations;
- three new constraint programming (CP) formulations;
- preprocessing techniques that reduce the size of the models;
- comparative experiments using the new and existing models on small synthetic instances and large instances built from crawled  $\mathbb{X}$  networks.

This text is organized as follows. In Section 2, we formally define the LDPAP and review the literature on the problem. The new IP and CP models are presented in Section 3 and Section 4, respectively. In Section 5, we introduce the preprocessing techniques for these formulations. Later, in Section 6, we report a set of computational experiments and analyze the results with the objective of empirically evaluating the exact models. Lastly, in Section 7, we close the paper with concluding remarks and address future work.

## 2 Object of Study

Although the LDPAP has just recently been introduced into the literature, a number of theoretical and practical results are already known. Next, we formally present the LDPAP and summarize previous work.

### 2.1 Problem Statement

The LDPAP, as first introduced in [15], can be described as follows. Consider a portion of an online social network represented by a directed graph  $D = (V, E)$ , where  $V$  and  $E$  are the sets of vertices and directed edges (arcs) of  $D$ . Each vertex in  $V$  represents an individual, and each  $(u, v) \in E$  indicates that  $u$  can exert influence over  $v$ . The *in-* and *out-neighborhoods* of each  $v \in V$  are indicated by  $N_{\text{in}}(v) = \{u \in V : (u, v) \in E\}$  and  $N_{\text{out}}(v) = \{u \in V : (v, u) \in E\}$ .

A set of vertices selected to first disseminate the information is called a *seed set* and it contains the *seeds*. When the seeds transmit the information to their out-neighbors, some vertices may be influenced to the point that they forward the information, triggering a propagation. In this process, each vertex assumes at least one of three possible states regarding the information being disseminated:

- *ignorant*: the vertex is not a seed and has not received the information yet;
- *aware*: the vertex is a seed or has received the information at least once;
- *spreader*: the vertex is a seed or has received the information enough times to enable it to forward the information to others.

The *cost* of selecting a vertex  $v$  as a seed is denoted by  $c_v \in \mathbb{R}^+$  and the cost of a seed set  $S \subseteq V$  is  $c_S = \sum_{v \in S} c_v$ . Moreover, the *threshold* of  $v$ , denoted by  $t_v \in \mathbb{R}^+$ , quantifies the need for  $v$  to be sufficiently influenced to the point where  $v$  begins to pass on the information being propagated. In other words, if  $v$  is not a seed, then  $v$  becomes a spreader only if the amount of influence received by  $v$  is at least  $t_v$ . Furthermore, the *weight* of an edge  $(u, v) \in E$ , denoted by  $w_{u,v} \in \mathbb{R}^+$ , measures the degree of influence  $u$  can exert on  $v$ .

The passage of time in a propagation is divided into rounds. In each round  $\tau \geq 0$  of the propagation  $\mathcal{P}_S$  started from a seed set  $S$ , the sets of vertices that are in the spreader and aware states are denoted by  $S_\tau$  and  $A_\tau$ , respectively. In the beginning of  $\mathcal{P}_S$ ,  $S_0 = A_0 = S$ , and for each  $\tau \geq 1$  we have:

$$\begin{aligned} A_\tau &= A_{\tau-1} \cup \{v \in V \setminus A_{\tau-1} : |N_{\text{in}}(v) \cap S_{\tau-1}| \geq 1\}; \\ S_\tau &= S_{\tau-1} \cup \{v \in V \setminus S_{\tau-1} : \sum_{u \in N_{\text{in}}(v) \cap S_{\tau-1}} w_{u,v} \geq t_v\}. \end{aligned}$$

Note that every vertex in the spreader state is also in the aware state, i.e.,  $S_\tau \subseteq A_\tau$  for any  $\tau \geq 0$ , but the reverse is not necessarily true. The propagation ends when  $S_{\rho-1} = S_\rho$  for some  $\rho \geq 1$ . If all vertices are aware at the end of  $\mathcal{P}_S$ , i.e.,  $A_\rho = V$ ,  $S$  is called a *perfect seed set*.

Formally, in the LDPAP, we are given an instance  $I = (D, c, t, w)$ , where  $D = (V, E)$  is a directed graph, and  $c : V \rightarrow \mathbb{R}^+$ ,  $t : V \rightarrow \mathbb{R}^+$  and  $w : E \rightarrow \mathbb{R}^+$  are cost, threshold, and weight functions, respectively. The problem's objective is to find a perfect seed set of minimum cost.

## 2.2 Previous Work

In [15], it is shown that the LDPAP is NP-hard and cannot be approximated within a ratio of  $\mathcal{O}(2^{\log^{1-\varepsilon} n})$ , for any  $\varepsilon > 0$ , unless  $\text{NP} \subseteq \text{DTIME}(n^{\text{polylog}(n)})$ . These results also hold for the PAP [6], which is a special case of the LDPAP where  $(u, v) \in E$  iff  $(v, u) \in E$ ,  $w_{u,v} = 1$  for every  $(u, v) \in E$ , and  $t_v \in \mathbb{Z}^+$ ,  $c_v = 1$  for every  $v \in V$ . Moreover, the LDPAP remains NP-hard for acyclic graphs, which can be proved by a polynomial reduction from the *Hitting Set Problem* [10,15].

One may obtain a trivial lower bound for the objective function of the LDPAP by computing the sum of the costs of all vertices that are sources, if there exists any, or the least cost among all vertices, otherwise. An algorithm to obtain an alternative lower bound that is at least as good as the trivial one is proposed in [15]. The strategy is to contract the strongly connected components of the original graph to form a (probably smaller) new instance. The optimal value for the new instance provides a lower bound for the original one.

There are two IP models for the LDPAP called IP-ROUNDS and IP-ARCS, which were proposed in [15]. In that same paper, these formulations were tested with a commercial IP solver on 300 small instances containing up to 100 vertices. Within an hour of execution for each instance, the IP-ROUNDS and IP-ARCS models obtained optimal solutions for 90 and 54 instances, respectively. The authors also concluded that the IP-ARCS formulation appears to be more adequate for solving instances with sparse graphs, while the IP-ROUNDS model is a better fit for instances with graphs of intermediate and higher densities.

Due to the LDPAP's complexity, a natural way to tackle large instances is to employ heuristic algorithms. In [15], the authors introduce a heuristic for the LDPAP called RGR, which is based on the metaheuristic Reactive GRASP [19]. The RGR heuristic was tested on large instances with up to 50,000 vertices and proved to be greatly superior to a baseline greedy algorithm.

## 3 Integer Programming Formulations

In this section, we introduce two new IP formulations for the LDPAP, namely, IP-ARCS-POLY and IP-ORDERING. For ease of reference, we also present the two existing formulations from [15] namely, IP-ROUNDS and IP-ARCS. We remark that these models can be simplified to obtain TSSP formulations and they may be seen as adaptations from the IP models for the TSSP from [1,20,22]. Future TSSP models might lead to the design of new formulations for LDPAP.

Let  $I = (D = (V, E), c, t, w)$  be an instance of the LDPAP, where  $|V| = n$ . Every formulation in this section contains a set  $\{s_v : v \in V\}$  of binary variables such that  $s_v = 1$  iff  $v$  is a seed. The models also share the same objective function (1), which minimizes the cost of the seed set.

$$\min \sum_{v \in V} c_v s_v \tag{1}$$

### 3.1 The Existing IP-ROUNDS Formulation

In the IP-ROUNDS model, there exists a set  $\{x_{v,\tau} : v \in V, \tau \in \{0, 1, \dots, n\}\}$  of binary variables such that  $x_{v,\tau} = 1$  iff  $v$  is a spreader in round  $\tau$ . Note that  $n+1$  is the maximum number of rounds it takes for a propagation to end. The rest of the formulation comprises constraints (2)–(4).

$$x_{v,0} = s_v \quad \forall v \in V \quad (2)$$

$$\sum_{u \in N_{\text{in}}(v)} w_{u,v} x_{u,\tau-1} + t_v x_{v,0} \geq t_v x_{v,\tau} \quad \forall v \in V \quad \forall \tau \in \{1, 2, \dots, n\} \quad (3)$$

$$x_{v,0} + \sum_{u \in N_{\text{in}}(v)} x_{u,n-1} \geq 1 \quad \forall v \in V \quad (4)$$

Constraints (2) determine that the seed set is formed by the vertices that are spreaders in round  $\tau = 0$ . Constraints (3) forbid each vertex  $v$  from being a spreader in round  $\tau$  (i.e., make  $x_{v,\tau} = 0$ ) if  $v$  is neither a seed ( $x_{v,0} = 0$ ) nor receives enough influence from its in-neighbors that are spreaders in round  $\tau - 1$  (i.e.,  $\sum_{u \in N_{\text{in}}(v)} w_{u,v} x_{u,\tau-1} < t_v$ ). Lastly, constraints (4) ensure that every vertex  $v$  is either a seed or has an in-neighbor that is spreader in round  $\tau = n - 1$  and, consequently, is aware in round  $n$ . The IP-ROUNDS formulation has  $\mathcal{O}(|V|^2)$  binary variables and  $\mathcal{O}(|V|^2)$  constraints.

### 3.2 The Existing IP-ARCS Formulation

The IP-ARCS model has a set  $\{y_{u,v} : (u, v) \in E\}$  of binary variables such that  $y_{u,v} = 1$  iff  $u$  influences  $v$  during the propagation. Let  $\Xi$  be the collection of all directed cycles of  $D$ . The IP-ARCS formulation includes constraints (5)–(8).

$$y_{u,v} + s_v \leq 1 \quad \forall (u, v) \in E \quad (5)$$

$$\sum_{i \in N_{\text{in}}(u)} w_{i,u} y_{i,u} + t_u s_u \geq t_u y_{u,v} \quad \forall (u, v) \in E \quad (6)$$

$$s_v + \sum_{u \in N_{\text{in}}(v)} y_{u,v} \geq 1 \quad \forall v \in V \quad (7)$$

$$\sum_{(u,v) \in \xi} y_{u,v} \leq |\xi| - 1 \quad \forall \xi \in \Xi \quad (8)$$

Constraints (5) establish that, if a vertex is a seed, it cannot be influenced by any of its in-neighbors. Constraints (6) guarantee that, for any  $(u, v) \in E$ , if  $u$  influences  $v$  during the propagation ( $y_{u,v} = 1$ ), then  $u$  necessarily enters the spreader state before that happens, either by being a seed ( $s_u = 1$ ) or by receiving an amount of influence of at least  $t_u$ . Constraints (7) enforce that each vertex is a seed or is influenced by one of its in-neighbors, and hence aware.

Lastly, (8) forbid the occurrence of a circular sequence of influences during the propagation by avoiding directed cycles in the directed subgraph of  $D$  induced by each  $(u, v) \in E$  with  $y_{u,v} = 1$ . The IP-ARCS formulation has  $\mathcal{O}(|V| + |E|)$  variables and  $\mathcal{O}(|V| + |E|)$  constraints, except for (8), which can be exponential.

### 3.3 The New IP-ARCS-POLY Formulation

The new IP-ARCS-POLY formulation is obtained by substituting constraints (8) in IP-ARCS by an alternative set of constraints with polynomial size. The idea is similar to the subtour elimination constraints in the Miller–Tucker–Zemlin formulation for the Travelling Salesman Problem [12].

Let  $\{\ell_v : v \in V\}$  be a set of integer variables with domain  $\{0, 1, \dots, n-1\}$ . For each  $v \in V$ ,  $\ell_v$  is the length of the longest path that starts from a seed and ends at  $v$  in the directed subgraph of  $D$  induced by the edges in  $\{(u, v) \in E : y_{u,v} = 1\}$ . The IP-ARCS-POLY model comprises constraints (5)–(7), (9), and (10).

$$(n-1) \cdot (1 - s_v) \geq \ell_v \quad \forall v \in V \quad (9)$$

$$n(y_{u,v} - 1) + 1 \leq \ell_v - \ell_u \quad \forall (u, v) \in E \quad (10)$$

Constraints (9) determine that if  $v$  is a seed ( $s_v = 1$ ), then the length of the longest path that starts from a seed and ends at  $v$  is 0 (i.e.,  $\ell_v = 0$ ). On the other hand, constraints (10) enforce that if  $u$  influences  $v$  (i.e.,  $y_{u,v} = 1$ ), then the length of the longest path that starts from a seed and ends at  $v$  must be larger than the length of the longest path that starts from a seed and ends at  $u$  (i.e.,  $\ell_v > \ell_u$ ). The IP-ARCS-POLY model has  $\mathcal{O}(|V| + |E|)$  binary variables,  $\mathcal{O}(|V|)$  integer (non-binary) variables, and  $\mathcal{O}(|V| + |E|)$  constraints.

### 3.4 The New IP-ORDERING Formulation

In the new IP-ORDERING formulation, there are three sets of binary variables, namely,  $\{p_v : v \in V\}$ ,  $\{q_v : v \in V\}$ ,  $\{h_{u,v} : u, v \in V, u \neq v\}$ , such that:

- $p_v = 1$  iff  $v$  is not a seed, but becomes a spreader during the propagation;
- $q_v = 1$  iff  $v$  is not a seed and does not become a spreader, but is in the aware state at the end of the propagation;
- $h_{u,v} = 1$  iff  $u$  and  $v$  are both spreaders during the propagation, but  $u$  assumes that state before  $v$  does.

The model comprises constraints (11)–(17). The role of the  $h$  variables is to induce the order in which the vertices assume their spreader state.

$$s_v + p_v + q_v = 1 \quad \forall v \in V \quad (11)$$

$$\sum_{u \in N_{\text{in}}(v)} w_{u,v} h_{u,v} \geq t_v p_v \quad \forall v \in V \quad (12)$$

$$\sum_{u \in N_{\text{in}}(v)} (s_u + p_u) \geq q_v \quad \forall v \in V \quad (13)$$

$$s_u + p_u \geq h_{u,v} \quad \forall (u, v) \in V \times V, u \neq v \quad (14)$$

$$h_{u,v} \leq p_v \quad \forall (u, v) \in V \times V, u \neq v \quad (15)$$

$$h_{u,v} + h_{v,u} + q_v \leq 1 \quad \forall (u, v) \in V \times V, u \neq v \quad (16)$$

$$h_{i,j} + h_{j,k} \leq h_{i,k} + 1 \quad \forall (i, j, k) \in V \times V \times V, i \neq j \neq k \quad (17)$$

Constraints (11) guarantee that each vertex is either a seed or assumes the spreader or aware state. Constraints (12) enforce that, if  $v$  is not a seed but becomes a spreader, the amount of influence received by  $v$  from its in-neighbors that are spreaders before  $v$  must be at least  $t_v$ . Constraints (13) determine that if  $v$  is neither a seed nor becomes a spreader but is otherwise aware at the end of the propagation, then  $v$  must have at least one in-neighbor that is either a seed or becomes a spreader during the propagation.

Constraints (14) and (15) establish that, if  $u$  becomes a spreader before  $v$  does, then either  $u$  is a seed or it becomes a spreader later, and  $v$  has to become a spreader as well. Constraints (16) imply that at most one of the following can be true:  $u$  assumes the spreader state before  $v$  does;  $v$  assumes the spreader state before  $u$  does;  $v$  is neither a seed nor becomes a spreader, but becomes aware during the propagation. Lastly, (17) ensure transitivity in the ordering of the vertices that assume the spreader state during the propagation. The IP-ORDERING model has  $\mathcal{O}(|V|^2)$  binary variables and  $\mathcal{O}(|V|^3)$  constraints.

## 4 Constraint Programming Formulations

In this section, we rewrite the IP-ROUNDS, IP-ARCS-POLY, and IP-ORDERING formulations by converting some of their linear constraints into propositional logic constraints, which are more natural and adequate for a CP solver. The converted constraints represent logical implications originally formulated with big-M type coefficients. This is the case for constraints (3), (6), (9), and (10). As a result of this rewriting, we derive three CP formulations for the LDPAP, namely, CP-ROUNDS, CP-ARCS-POLY, and CP-ORDERING. Before describing them, we explain why we did not rewrite IP-ARCS.

Because of the exponentially many constraints (8) in the IP-ARCS model, one would generally use a row generation algorithm in which (8) are regarded as *lazy constraints* that get introduced only as they become violated. The CP solvers we are familiar with do not allow lazy constraints since they require all constraints to be loaded a priori, which is not doable for (8). Although it is possible to use a CP solver to repeatedly solve the problem and add violated constraints (8), we tested such an implementation and concluded it is too time consuming and not effective in practice. Some CP solvers, such as Gecode [8], include a global constraint called DAG that avoids induced directed cycles, but that constraint basically consists of using (9) and (10). Therefore, substituting DAG for (8) in IP-ARCS would simply lead to the same IP-ARCS-POLY model that we are already studying.

We now introduce the CP-ROUNDS, CP-ARCS-POLY and CP-ORDERING formulations, which share the same objective function (1).

### 4.1 The CP-ROUNDS Formulation

The CP-ROUNDS model is derived from IP-ROUNDS by substituting (18) for (3).

$$\text{IF } (x_{v,\tau} \text{ AND } \neg x_{v,0}) \text{ THEN } \sum_{u \in N_{\text{in}}(v)} w_{u,v} x_{u,\tau-1} \geq t_v \quad \forall v \in V \quad \forall \tau \in \{1, 2, \dots, n\} \quad (18)$$

For every vertex  $v$ , (18) say that if  $v$  is a spreader in round  $\tau$  but not a seed, then the amount of influence it receives in round  $\tau - 1$  is at least  $t_v$ . Like IP-ROUNDS, this model has  $\mathcal{O}(|V|^2)$  binary variables and  $\mathcal{O}(|V|^2)$  constraints.

#### 4.2 The CP-ARCS-POLY Formulation

The CP-ARCS-POLY model is derived from IP-ARCS-POLY by substituting (19), (20), and (21) for (6), (9), and (10), respectively.

$$\text{IF } (y_{u,v} \text{ AND } \neg s_u) \text{ THEN } \sum_{i \in N_{\text{in}}(u)} w_{i,u} y_{i,u} \geq t_u \quad \forall (u, v) \in E \quad (19)$$

$$\text{IF } s_v \text{ THEN } \ell_v = 0 \quad \forall v \in V \quad (20)$$

$$\text{IF } y_{u,v} \text{ THEN } \ell_u < \ell_v \quad \forall (u, v) \in E \quad (21)$$

Constraints (19) guarantee that, for any  $(u, v) \in E$ , if  $u$  influences  $v$  during a propagation but  $u$  is not a seed, then  $u$  must have become a spreader by receiving enough influence. Constraints (20) establish that, if  $v$  is a seed, the longest path from a seed to  $v$  has length 0. Lastly, (21) indicate that, if  $u$  influences  $v$ , the longest path from a seed to  $u$  must be shorter than the longest path from a seed to  $v$ . The CP-ARCS-POLY model has  $\mathcal{O}(|V| + |E|)$  binary variables,  $\mathcal{O}(|V|)$  (non-binary) integer variables, and  $\mathcal{O}(|V| + |E|)$  constraints.

#### 4.3 The CP-ORDERING Formulation

The CP-ORDERING formulation is derived from IP-ORDERING by substituting (22) and (23) for (12) and (17), respectively.

$$\text{IF } p_v \text{ THEN } \sum_{u \in N_{\text{in}}(v)} w_{u,v} h_{u,v} \geq t_v \quad \forall v \in V \quad (22)$$

$$\text{IF } (h_{i,j} \text{ AND } h_{j,k}) \text{ THEN } h_{i,k} \quad \forall (i, j, k) \in V \times V \times V, i \neq j \neq k \quad (23)$$

Constraints (22) determine that, if a non-seed vertex  $v$  becomes a spreader, the amount of influence received by  $v$  must be at least  $t_v$ . Constraints (23) enforce transitivity on the spreading order implied by the  $h$  variables. The CP-ORDERING model has  $\mathcal{O}(|V|^2)$  binary variables and  $\mathcal{O}(|V|^3)$  constraints.

## 5 Preprocessing Methods for the LDPAP Formulations

In this section, we present useful preprocessing techniques that can be applied to the proposed models to reduce their sizes either by removing unnecessary variables or constraints, or by reducing some variable domains.

Let  $I = (D = (V, E), c, t, w)$  be an instance of the LDPAP with  $|V| = n$ , and let  $Q \subseteq V$  be the set of source vertices in  $D$  (i.e., vertices with an empty in-neighborhood). Clearly, any feasible solution for  $I$  must contain  $Q$ , otherwise the sources would not become aware during the propagation. Now, let  $\mathcal{P}_Q$  be



a propagation started from  $Q$  and let  $\rho \geq 1$  be the last round of  $\mathcal{P}_Q$ . Then,  $Q_\rho = Q_{\rho-1}$ , where  $Q_i$  denotes the set of spreaders in the  $i$ -th round. Take  $\kappa = \rho + n + 1 - |Q_{\rho-1}|$ . Our first preprocessing method relies on Theorem 1.

**Theorem 1.** *If  $S$  is a feasible solution for  $I$ , then the propagation  $\mathcal{P}_S$  started from  $S$  takes no more than  $\kappa$  rounds.*

*Proof.* Let  $S$  be a feasible solution for  $I$  so that  $\mathcal{P}_S$  takes at least  $\kappa + 1$  rounds. Since  $S$  is feasible,  $Q \subseteq S$  and, therefore,  $Q_{\rho-1} \subseteq S_{\rho-1}$ . Hence, at the end of round  $\rho - 1$  of  $\mathcal{P}_S$ , there are at least  $|Q_{\rho-1}|$  spreaders and, consequently, at most  $n - |Q_{\rho-1}|$  non-spreaders. Therefore, there can be at most  $n - |Q_{\rho-1}| + 1$  rounds after  $\rho - 1$ , totaling  $\rho + n + 1 - |Q_{\rho-1}| = \kappa$  rounds (because there are  $\rho$  rounds from 0 to  $\rho - 1$ ), which contradicts the choice of  $S$ .  $\square$

We can compute both  $\rho$  and  $|Q_{\rho-1}|$  (and thus  $\kappa$ ) by simulating  $\mathcal{P}_Q$  using the `CompletePropagation` algorithm, introduced in [15], which runs in  $\mathcal{O}(|V| + |E|)$  time. To do so, we keep a counter for the number of rounds and another for the number of spreaders per round.

So far in this paper, we have used  $n + 1$  as an upper bound for the total number of rounds that a propagation can take. Indeed, some variables in the `IP-ROUNDS`, `IP-ARCS-POLY`, `CP-ROUNDS`, and `CP-ARCS-POLY` formulations were defined by assuming that, in the worst-case scenario, a propagation ends in round  $n$ . We now tighten the  $n + 1$  upper bound to  $\kappa$ . In doing so, we are assuming that, in the worst case, a propagation ends in round  $\kappa - 1$ . Next, we describe how we adjust the models accordingly.

For the `IP-ROUNDS` and `CP-ROUNDS` models, we remove all variables  $x_{v,\tau}$  with  $v \in V$  and  $\tau \geq \kappa$ , as well as every constraint of type (3) and (18) in which they occur. Then, we substitute (24) for (4).

$$x_{v,0} + \sum_{u \in N_{\text{in}}(v)} x_{u,\kappa-2} \geq 1 \quad \forall v \in V \quad (24)$$

For the `IP-ARCS-POLY` and `CP-ARCS-POLY` models, we simply change the domain of the  $\ell$  variables from  $\{0, 1, \dots, n - 1\}$  to  $\{0, 1, \dots, \kappa - 2\}$ .

Now, we present a preprocessing approach for the `IP-ORDERING` formulation. Recall that each binary variable  $h_{u,v}$  indicates whether  $u$  assumes the spreader state before  $v$  does. Moreover, (16) and (17) guarantee that the directed graph defined by the edge set  $\{(u, v) : u, v \in V, u \neq v, h_{u,v} = 1\}$  is acyclic. In other words, these constraints forbid any circular sequence of vertices with respect to the order in which they assume the spreader state.

During a propagation, however, a circular sequence of influence induced by  $h$  can only occur between vertices belonging to the same strongly connected component of  $D$ . Thus, instead of ensuring a non-circular ordering of the vertices for the entire graph, we can rewrite the constraints to focus separately on each strongly connected component of  $D$ .

We redefine the set of  $h$  variables from  $\{h_{u,v} : u, v \in V, u \neq v\}$  to  $\{h_{u,v} : (u, v) \in E \cup E'\}$  where  $E'$  contains edges  $(u, v)$  and  $(v, u)$  for every distinct

vertices  $u$  and  $v$  that belong to the same strongly connected component of  $D$ . Next, we substitute constraints (25) to (28) for constraints (14) to (17).

$$s_u + p_u \geq h_{u,v} \quad \forall (u, v) \in E \cup E' \quad (25)$$

$$h_{u,v} \leq p_v \quad \forall (u, v) \in E \cup E' \quad (26)$$

$$h_{u,v} + h_{v,u} + q_v \leq 1 \quad \forall (u, v) \in E' \quad (27)$$

$$h_{i,j} + h_{j,k} \leq h_{i,k} + 1 \quad \forall \text{ distinct } i, j, k \in V \text{ s.t. } (i, j), (j, k), (i, k) \in E' \quad (28)$$

To consider the CP-ORDERING formulation with the preprocessing approach we have described for the IP-ORDERING model, we apply the same modifications proposed above and also substitute (29) for (23).

$$\text{IF } (h_{i,j} \text{ AND } h_{j,k}) \text{ THEN } h_{i,k} \quad \forall \text{ distinct } i, j, k \in V \text{ s.t. } (i, j), (j, k), (i, k) \in E' \quad (29)$$

## 6 Computational Experiments

In this section, we describe the experiments we carried out with the formulations presented in Sections 3 and 4. We used a machine equipped with an Intel<sup>®</sup> Xeon<sup>®</sup> E5-2630 v4 processor, 64 GB of RAM, and the Ubuntu 22.04.1 LTS operating system. We employed Gurobi v10.0.3 [9] as the IP solver and CP-SAT v9.7.2996 (from Google OR-Tools) [18] as the CP solver. The instances in our experiments were divided into two datasets, outlined in the next section.

We refer the reader to a publicly available repository [17] that accompanies this paper and includes the source code, problem instances, the solutions obtained, and an appendix with additional details about our experimental results.

### 6.1 Datasets

The first dataset, denoted by  $\Delta_{\text{syn}}$ , was introduced in [15] and contains 30 synthetic instances with  $n$  vertices for each  $n \in \{10, 20, \dots, 100\}$ , totaling 300 instances. The graphs in these instances were generated using a well known algorithm proposed in [3] for the creation of scale-free graphs that capture crucial characteristics of social networks. Particularly, the  $\Delta_{\text{syn}}$  dataset was purposely designed to be diverse regarding the densities of the graphs, calculated as  $m/(n^2 - n)$ , where  $m$  is the number of edges. For each fixed  $n$ , the 30 instances in  $\Delta_{\text{syn}}$  with  $n$  vertices vary from very sparse graphs to nearly complete graphs.

For each instance  $(D, w, c, t)$  in  $\Delta_{\text{syn}}$ , the weight, cost, and threshold functions are given by (30), (31) and (32), which were originally presented in [15].

$$w_{u,v} = |N_{\text{out}}(u)| / \left( \sum_{w \in N_{\text{in}}(v)} |N_{\text{out}}(w)| \right) \quad (30)$$

$$c_v = \begin{cases} 5 \cdot |N_{\text{out}}(v)|, & \text{if } |N_{\text{out}}(v)| > 0 \\ +\infty, & \text{otherwise} \end{cases} \quad (31)$$

$$t_v = \begin{cases} \frac{|N_{\text{in}}(v)|}{2} \cdot \text{median}_{u \in N_{\text{in}}(v)} \{w_{u,v}\}, & \text{if } |N_{\text{in}}(v)| > 0 \\ +\infty, & \text{otherwise.} \end{cases} \quad (32)$$

Formula (30) establishes that the amount of influence that  $u$  exerts on  $v$  is proportional to the popularity of  $u$  on the network compared to the popularity of the other in-neighbors of  $v$ . Formula (31) is based on the fact that, in 2020, influencers with up to 1 million followers on some online social networks were charging about 5 cents of a dollar per follower for a single paid post [4]. Note that when  $v$  has no influence over other users, assigning  $+\infty$  as its cost prevents  $v$  from being picked as an ineffective seed.

Formula (32) determines that a vertex  $v$  begins to forward the information when it receives an amount of influence equivalent to the total influence coming from half of its in-neighbors, assuming that all of them were exerting a median level of influence on  $v$ . This is an extension of the well-studied *majority threshold function*, where, in a given unweighted and undirected graph, a non-seed vertex spreads the information when at least half of its neighbors are spreaders [5]. For more details on the generation of the  $\Delta_{\text{syn}}$  dataset, we refer the reader to [15].

With the objective of evaluating the scalability of the proposed models for larger but still solvable instances, we designed a second set of instances, denoted by  $\Delta_{\mathbb{X}}$ , containing large graphs. The  $\Delta_{\mathbb{X}}$  dataset was built using a set of users of the  $\mathbb{X}$  social network and their online interactions between August 2019 and March 2020 [21]. For each of 14 selected topics, from soccer to politics, the authors identified the users who actively posted, retweeted or quoted, and then crawled their followers that were also active users on the same topic.

These crawled relationships between “follower” and “followed” were previously used in [15] to obtain a benchmark of instances used to test a heuristic for the LDPAP (see Section 2.2). We first attempted to test our exact models on that benchmark, but preliminary results indicated that the formulations could not be solved within the established time limit of one hour and sometimes could not even be loaded into memory due to the very large sizes of the instances, specifically their number of edges. Thus, we were compelled to reduce the number of edges in these graphs in order to stress our models on large, yet solvable, instances.

Therefore, we designed the  $\Delta_{\mathbb{X}}$  dataset so that its instances contain a subset of the user relationships present in the original instances from [15]. This resulted in 14 large instances, one for each crawled topic. Next, we outline the details of the  $\Delta_{\mathbb{X}}$  dataset construction.

First, we grouped the users with respect to their popularity in the network, in accordance with the approach proposed in [4], where the influential users were classified according to their number of followers. To do so, we took  $r$  as the largest integer such that there exists a user with at least  $2^r$  followers, and then partitioned the users into the following equivalence classes:

- *Group A*: at least  $2^r$  followers;
- *Group B*: at least  $2^{r-1}$  and fewer than  $2^r$  followers;
- *Group C*: at least  $2^{r-2}$  and fewer than  $2^{r-1}$  followers;
- *Group D*: fewer than  $2^{r-2}$  followers.

We then assumed that the information typically flows from more popular users to less popular ones or between users within the same class, except for users within the least popular Group D. Next, we constructed a graph that

contains an edge  $(u, v)$ , meaning that  $u$  exerts influence over  $v$ , for each pair of users  $u$  and  $v$  such that  $v$  follows  $u$ , and either  $u$  is more popular than  $v$  or they both belong to the same group A, B or C. This construction purposely forbids outgoing edges from vertices in group D, since they correspond to users with very few followers and are not significantly influential in practice. Lastly, we used (30), (31) and (32) to compute the remaining parameters required to create an instance of the LDPAP. Table 1 shows properties of the graphs that make up the instances in  $\Delta_{\mathbb{X}}$ , where SCCs denotes the number of strongly connected components (SCC) and LSCC is the number of vertices in the largest SCC.

**Table 1.** Quantifying topological characteristics of  $\Delta_{\mathbb{X}}$ .

Instance	Vertices	Edges	Density	SCCs	LSCC
$X_{01}$	6529	69385	0.00163	6344	170
$X_{02}$	16976	646190	0.00224	16123	719
$X_{03}$	29693	250162	0.00028	29515	172
$X_{04}$	275	1244	0.01651	250	26
$X_{05}$	1637	11744	0.00439	1520	99
$X_{06}$	6537	310694	0.00727	5725	664
$X_{07}$	8068	138701	0.00213	7362	572
$X_{08}$	7857	164710	0.00267	6994	730
$X_{09}$	927	5371	0.00626	884	35
$X_{10}$	9271	58828	0.00068	9209	50
$X_{11}$	3964	24296	0.00155	3903	56
$X_{12}$	1520	13287	0.00575	1388	104
$X_{13}$	993	16828	0.01708	685	173
$X_{14}$	1405	8181	0.00415	1370	32

## 6.2 Results for Instances in $\Delta_{\text{syn}}$

In this section, we report the results for each of the 300 instances from  $\Delta_{\text{syn}}$ . We configured both solvers, Gurobi and CP-SAT, to run on a single thread of execution within a time limit of 1 hour for each pair of formulation and instance.

The preprocessing methods that we proposed in Section 5 to reduce the size of the models were employed whenever they were applicable. They proved to be much more effective than the built-in preprocessing phase inherent to the solver, so the time spent running them was fully compensated by the performance gains.

Moreover, for each instance, we provided the solvers with an initial feasible solution that corresponds to the best known perfect seed set for that instance thus far. These seed sets were obtained from experiments conducted in [15] and are publicly available in [14]. We also provided the solvers with an initial lower bound for the objective function, which was obtained by the algorithm proposed in [15] (see Section 2.2).

It is important to highlight that, as shown in Section 3, the IP-ARCS model has an exponential number of constraints (8). As a result, we followed the traditional lazy constraint strategy: whenever the IP solver found an integer solution, we performed a complete depth-first search on the directed graph induced by the

integer solution and, for each cycle found, we added the corresponding violated constraint to the formulation.

We remark that, in [15], the IP-ROUNDS and IP-ARCS models were tested with the  $\Delta_{\text{syn}}$  dataset and found optimal solutions for 90 and 54 instances, respectively, totalling 93 instances solved.

Regarding the experiments conducted in the present paper, Table 2 shows the number of provably optimal solutions obtained by each model, where the instances are grouped by their number of vertices (recall that there are 30 instances for each value of  $|V|$ ).

**Table 2.** Number of solved instances from  $\Delta_{\text{syn}}$ .

Formulation	$ V $										Total
	10	20	30	40	50	60	70	80	90	100	
IP-ROUNDS	<b>30</b>	28	9	4	3	3	3	3	4	4	91
IP-ARCS	<b>30</b>	9	4	3	2	2	1	1	1	1	54
IP-ARCS-POLY	<b>30</b>	14	5	3	2	2	1	1	1	1	60
IP-ORDERING	<b>30</b>	26	6	3	2	2	1	1	1	1	73
CP-ROUNDS	<b>30</b>	<b>30</b>	<b>30</b>	<b>30</b>	<b>30</b>	<b>30</b>	<b>30</b>	<b>29</b>	<b>22</b>	<b>17</b>	<b>278</b>
CP-ARCS-POLY	<b>30</b>	22	9	5	3	3	1	2	1	1	77
CP-ORDERING	<b>30</b>	<b>30</b>	<b>30</b>	28	12	6	6	2	3	2	149

We first observe that the results for the IP-ROUNDS and IP-ARCS models were similar to the ones obtained in [15], since they solved almost the exact same instances they did before. The IP-ARCS-POLY model had slightly superior performance than its exponential-size version, IP-ARCS, by solving 6 more instances with 20 and 30 vertices. The IP-ORDERING formulation outperformed both IP-ARCS and IP-ARCS-POLY formulations by solving a total of 73 instances. Among the IP models, IP-ROUNDS remained the best formulation for the  $\Delta_{\text{syn}}$  dataset, solving a total of 91 instances.

When it comes to the CP formulations, we observe that each model outperformed its own IP version. More specifically, the CP-ORDERING and CP-ROUNDS models solved substantially more instances than their IP counterparts for each group of instances (with at least 20 vertices). Indeed, the CP-ORDERING and CP-ROUNDS models were the ones that solved the largest numbers of instances, 149 and 278, respectively, surpassing all the IP models by a large margin. Next, we analyze the results obtained by these two formulations in more detail.

Because the CP-ROUNDS model was able to solve all 149 instances solved by the CP-ORDERING model, we compare the performance of these formulations by the time it took them to find these optimal solutions. Table 3 reports statistics on the running times for those 149 instances.

From the median values in Table 3, we see that both models solved at least half of the 149 instances in less than 2 seconds. However, the average times indicate that the CP-ROUNDS formulation was about 38 times faster than the CP-ORDERING model. The intervals between minimum and maximum running times, together with the standard deviations, suggest that the CP-ROUNDS model had more stable running times that grew more gradually.

**Table 3.** Running times (in seconds) for the 149 instances solved by both CP-ROUNDS and CP-ORDERING formulations.

Formulation	Min	Max	Median	Average	Std Dev
CP-ROUNDS	0.006	56.572	0.617	3.622	8.453
CP-ORDERING	0.004	3362.744	1.450	138.469	458.432

Regarding the 129 instances solved by the CP-ROUNDS model but not by CP-ORDERING (because it exceeded the 1-hour time limit), CP-ORDERING obtained an average optimality gap of 78.31%, with a standard deviation of 24.26. On the other hand, the CP-ROUNDS formulation proved optimality within 469 seconds on average, with a standard deviation of 746.

With respect to the 22 instances for which all formulations exceeded the time limit, the IP-ROUNDS and CP-ROUNDS models obtained the smallest optimality gaps, on average, namely 92.60% and 92.83%, respectively.

### 6.3 Results for Instances in $\Delta_{\mathbb{X}}$

In this section, we report the results of the experiments conducted with the  $\Delta_{\mathbb{X}}$  dataset. We maintained the same experimental settings used for the  $\Delta_{\text{syn}}$  set (see Section 6.2), with the exceptions noted below.

We configured the solvers to run on at most 10 threads of execution for each pair of formulation and instance. We also provided the solvers with initial feasible solutions that were obtained by running the only existing heuristic for the LDPAP [15] for 10 minutes per instance. Due to the large sizes of the instances from  $\Delta_{\mathbb{X}}$ , the combinatorial lower bound proposed in [15] could not be computed. Instead, we provided the solvers with trivial lower bounds (see Section 2.2).

Table 4 reports the optimality gaps obtained for each formulation within the 1-hour time limit. Entries containing the ‘-’ symbol indicate the run could not be completed due to lack of memory.

**Table 4.** Optimality gaps (%) obtained for the instances in  $\Delta_{\mathbb{X}}$ .

Formulation	$X_{01}$	$X_{02}$	$X_{03}$	$X_{04}$	$X_{05}$	$X_{06}$	$X_{07}$	$X_{08}$	$X_{09}$	$X_{10}$	$X_{11}$	$X_{12}$	$X_{13}$	$X_{14}$
IP-ROUNDS	88.87	-	-	<b>0.00</b>	<b>0.00</b>	-	-	-	<b>0.00</b>	-	20.22	31.32	<b>85.28</b>	<b>0.00</b>
IP-ARCS	88.87	<b>93.63</b>	<b>0.00</b>	<b>0.00</b>	<b>0.00</b>	<b>96.06</b>	76.36	52.46	<b>0.00</b>	64.31	<b>0.00</b>	31.32	98.73	<b>0.00</b>
IP-ARCS-POLY	88.87	98.65	<b>0.00</b>	<b>0.00</b>	<b>0.00</b>	<b>96.06</b>	76.36	52.46	<b>0.00</b>	64.31	<b>0.00</b>	31.32	98.73	<b>0.00</b>
IP-ORDERING	88.80	-	<b>0.00</b>	<b>0.00</b>	<b>0.00</b>	-	-	-	<b>0.00</b>	<b>0.00</b>	<b>0.00</b>	31.32	98.73	<b>0.00</b>
CP-ROUNDS	-	-	-	<b>0.00</b>	<b>0.00</b>	-	-	-	<b>0.00</b>	-	<b>0.00</b>	<b>0.00</b>	98.73	<b>0.00</b>
CP-ARCS-POLY	<b>88.46</b>	98.65	<b>0.00</b>	<b>0.00</b>	<b>0.00</b>	<b>96.06</b>	<b>0.00</b>	<b>0.00</b>	<b>0.00</b>	64.31	<b>0.00</b>	31.32	98.73	<b>0.00</b>
CP-ORDERING	-	-	-	<b>0.00</b>	<b>0.00</b>	-	-	-	<b>0.00</b>	<b>0.00</b>	<b>0.00</b>	<b>0.00</b>	-	<b>0.00</b>

We first analyze the results for the smallest instances from  $\Delta_{\mathbb{X}}$  with respect to the number of vertices in their largest SCCs (see Table 1). Instances  $X_{04}$ ,  $X_{05}$ ,  $X_{09}$ , and  $X_{14}$  were solved by each of the formulations in at most 200 seconds per execution, although most of these executions took only a few seconds.

Instance  $X_{10}$  was solved only by the IP-ORDERING and CP-ORDERING models in 532 and 17 seconds, respectively. Instance  $X_{11}$  was solved by all models, except for IP-ROUNDS, and optimal solutions were obtained in less than 2 minutes, except for CP-ROUNDS, which took 18 minutes. Instance  $X_{12}$  was solved by

the CP-ROUNDS and CP-ORDERING models in 278 and 73 seconds, respectively. The CP-ORDERING model was the only one capable of solving all the instances mentioned so far, always needing less than 2 minutes to prove optimality.

The remaining instances are larger than the previous ones with respect to their largest SCCs. In this case, the executions with CP-ORDERING were halted due to lack of memory, most likely due to the large number of constraints (29).

Instance  $X_{03}$  was solved by the CP-ARCS-POLY, IP-ORDERING, IP-ARCS and IP-ARCS-POLY models, taking less than 3 minutes for the first three. Instances  $X_{07}$  and  $X_{08}$  were solved only by the CP-ARCS-POLY model in less than 3 minutes. The remaining instances, namely  $X_{01}$ ,  $X_{02}$ ,  $X_{06}$ , and  $X_{13}$ , were not solved by any model. For these instances, the smallest optimality gaps were achieved, most of the time, by either the IP-ARCS or the CP-ARCS-POLY formulation.

In conclusion, for the  $\Delta_{\mathbb{X}}$  dataset, the CP-ORDERING formulation had the best overall performance on instances with relatively small strongly connected components. For instances whose LSCCs had more than 104 vertices, CP-ORDERING ran into memory limitations, and for these cases the IP-ARCS and CP-ARCS-POLY formulations were the most effective ones.

## 7 Concluding Remarks and Future Work

In this paper, we study different formulations and preprocessing approaches for the LDPAP. We propose two new IP models and three new CP formulations. We also design and perform experiments to test the existing and new models on a set of small synthetic instances and a new collection of large instances obtained from the  $\mathbb{X}$  social network.

Based on our results, the new CP-ROUNDS formulation is more suitable, both in terms of efficiency and effectiveness, to solve LDPAP instances with up to 100 vertices. With this model, we managed to increase the number of known solved instances from the  $\Delta_{\text{syn}}$  dataset from 93 to 278 (out of 300). For this group of instances, every CP model outperformed its IP counterpart.

Regarding the 14 large instances from the  $\Delta_{\mathbb{X}}$  dataset, we learned that the CP-ROUNDS model does not scale well with respect to the number of vertices in the graph. Our results suggest that CP-ORDERING is the most efficient and effective formulation for large instances with fairly small SCCs. For instances containing larger SCCs, the IP-ARCS and CP-ARCS-POLY formulations appear to be better suited.

As for future research directions, we believe that both the CP-ROUNDS and CP-ORDERING formulations can be employed in the development of heuristics for the LDPAP based on large neighborhood search or even on techniques grounded in the decomposition of large instances into smaller ones. For the second case, the smaller instances can, in turn, be quickly solved with these models depending on their topological characteristics. Also, it is our view that the possibility of applying Benders decomposition to the IP-ARCS model is worth investigating.

**Disclosure of Interests.** The authors have no competing interests to declare that are relevant to the content of this article.

## References

1. Ackerman, E., Ben-Zwi, O., Wolfowitz, G.: Combinatorial model and bounds for target set selection. *Theoretical Computer Science* **411**(44), 4017–4022 (2010). <https://doi.org/10.1016/j.tcs.2010.08.021>
2. Banerjee, S., Jenamani, M., Pratihari, D.K.: A survey on influence maximization in a social network. *Knowledge and Information Systems* **62**(9), 3417–3455 (2020). <https://doi.org/10.1007/s10115-020-01461-4>
3. Bollobás, B., Borgs, C., Chayes, J., Riordan, O.: Directed scale-free graphs. In: *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. p. 132–139. SODA '03, Society for Industrial and Applied Mathematics, USA (2003)
4. Campbell, C., Farrell, J.R.: More than meets the eye: The functional components underlying influencer marketing. *Business Horizons* **63**(4), 469–479 (2020). <https://doi.org/10.1016/j.bushor.2020.03.003>
5. Chen, N.: On the approximability of influence in social networks. *SIAM Journal on Discrete Mathematics* **23**(3), 1400–1415 (2009). <https://doi.org/10.1137/08073617X>
6. Cordasco, G., Gargano, L., Rescigno, A.A.: Active influence spreading in social networks. *Theoretical Computer Science* **764**, 15–29 (2019). <https://doi.org/10.1016/j.tcs.2018.02.024>
7. Gautam, R.K., Kare, A.S., Durga Bhavani, S.: Centrality measures based heuristics for perfect awareness problem in social networks. In: *Multi-disciplinary Trends in Artificial Intelligence*. pp. 91–100. Springer Nature Switzerland, Cham (2023). [https://doi.org/10.1007/978-3-031-36402-0\\_8](https://doi.org/10.1007/978-3-031-36402-0_8)
8. Gecode Team: Gecode — generic constraint development environment (2023), available from <http://www.gecode.org>
9. Gurobi Optimization, LLC: Gurobi Optimizer Reference Manual (2023), <https://www.gurobi.com>
10. Karp, R.M.: Reducibility among Combinatorial Problems, pp. 85–103. Springer US, Boston, MA (1972). [https://doi.org/10.1007/978-1-4684-2001-2\\_9](https://doi.org/10.1007/978-1-4684-2001-2_9)
11. Kempe, D., Kleinberg, J., Tardos, E.: Maximizing the spread of influence through a social network. In: *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. p. 137–146. KDD '03, ACM, New York, NY, USA (2003). <https://doi.org/10.1145/956750.956769>
12. Miller, C., Tucker, A., Zemlin, R.: Integer programming formulation of traveling salesman problems. *J. ACM* **7**(4), 326–329 (1960). <https://doi.org/10.1145/321043.321046>
13. Pereira, F.C.: A computational study of the Perfect Awareness Problem. Master's thesis, University of Campinas, Brazil (2021), <http://hdl.handle.net/20.500.12733/1641217>
14. Pereira, F.C., de Rezende, P.J.: The Least Cost Directed Perfect Awareness Problem – Benchmark Instances and Solutions. *Mendeley Data*, V2 (2023). <https://doi.org/10.17632/xgtjgzf28r>
15. Pereira, F.C., de Rezende, P.J.: The Least Cost Directed Perfect Awareness Problem: complexity, algorithms and computations. *Online Social Networks and Media* **37-38** (2023). <https://doi.org/10.1016/j.osnem.2023.100255>
16. Pereira, F.C., de Rezende, P.J., de Souza, C.C.: Effective heuristics for the perfect awareness problem. *Procedia Computer Science* **195**, 489–498 (2021). <https://doi.org/10.1016/j.procs.2021.11.059>, proceedings of the XI Latin and American Algorithms, Graphs and Optimization Symposium.



17. Pereira, F.C., de Rezende, P.J., Yunes, T.: Minimizing the Cost of Leveraging Influencers in Social Networks: IP and CP Approaches - Complementary Data. Mendeley Data, V2 (2023). <https://doi.org/10.17632/tkk5pdswtj>
18. Perron, L., Didier, F.: CP-SAT (Google OR-Tools) (2023), [https://developers.google.com/optimization/cp/cp\\_solver/](https://developers.google.com/optimization/cp/cp_solver/)
19. Prais, M., Ribeiro, C.C.: Reactive grasp: An application to a matrix decomposition problem in tdma traffic assignment. *INFORMS Journal on Computing* **12**(3), 164–176 (2000). <https://doi.org/10.1287/ijoc.12.3.164.12639>
20. Raghavan, S., Zhang, R.: A branch-and-cut approach for the weighted target set selection problem on social networks. *INFORMS Journal on Optimization* **1**(4), 304–322 (2019). <https://doi.org/10.1287/ijoo.2019.0012>
21. Schweimer, C., Gfrerer, C., Lugstein, F., Pape, D., Velimsky, J.A., Elsässer, R., Geiger, B.C.: Generating simple directed social network graphs for information spreading. In: *Proceedings of the ACM Web Conference 2022*. p. 1475–1485. WWW '22, ACM, New York, USA (2022). <https://doi.org/10.1145/3485447.3512194>
22. Shakarian, P., Eyre, S., Paulo, D.: A scalable heuristic for viral marketing under the tipping model. *Social Network Analysis and Mining* **3**(4), 1225–1248 (Dec 2013). <https://doi.org/10.1007/s13278-013-0135-7>