Maximizing Influence Blocking with Competing Cascades using Integer Linear Programming

Guilherme Carbonari Boneti
Department of Computer Science, Federal University of Paraná
Curitiba, Brazil
gcb19@inf.ufpr.br

Renato Silva de Melo
Department of Computer Science, Federal University of Paraná
Curitiba, Brazil
rsmelo@inf.ufpr.br

André Luis Vignatti
Department of Computer Science, Federal University of Paraná
Curitiba, Brazil
vignatti@inf.ufpr.br

ABSTRACT

In the *influence blocking maximization* problem, we aim to minimize the spread of disinformation in a network. More specifically, given a set of nodes that initiate the spread of disinformation and an integer k, we must find k nodes in the network to serve as the starting point for competing (say, correct) information so that the misinformation spread is minimized. In this paper, we propose a version of the problem under a dissemination model derived from the Competitive Linear Threshold model. We present an integer linear programming formulation for the problem that, as far as we know, inaugurates the use of mathematical programming in dissemination problems with two competing cascades. We performed experiments to verify the quality of our formulation, evaluating the integrality gap and the scalability. Such results can serve as a baseline for future solutions using integer linear programming.

KEYWORDS. Influence Blocking Maximization. Misinformation. Integer Linear Programming.

Paper topics: Combinatorial Optimization. Mathematical Programming.

1. Introduction

In recent years, there has been a growing interest in the propagation of influence through social networks, owing to the rapid growth of social media and their increasing importance in our society. In these environments, any piece of information can be disseminated rapidly and achieve significant scale. Unfortunately, this has also led to the proliferation of misinformation on social media platforms.

Misinformation has always been a problem, but it seems to have gained momentum with the rise of social media [Lazer et al., 2018]. People tend to believe information that aligns with their social narratives and discredit information that challenges their beliefs [Lewandowsky et al., 2012]. Social media's structure and methods of spreading information can amplify the spread of misinformation. [Vosoughi et al., 2018] found that on Twitter, fake news is 70% more likely to be shared than real news. Several recent studies have demonstrated the potential for misinformation to influence societal behavior. For example, Allcott and Gentzkow [Allcott e Gentzkow, 2017] analyzed the potential impact of misinformation on the outcome of the 2016 United States presidential election. Another instance is the proliferation of questionable sources on major social media platforms during the COVID-19 outbreak, as documented by Cinelli et al [Cinelli et al., 2020].

1.1. Contributions

We formalize and investigate a problem motivated by the issue of misinformation, which we call *Influence Blocking Maximization in the SCLT model*, drawing on previous research on this subject [He et al., 2012]. This problem arises when two information flows compete for dominance in a network, and our approach can help identify optimal strategies for blocking or mitigating the impact of unwanted information flows.

Our contribution comprises the formal definition of a version of the problem based on the SCLT dissemination model, an integer linear programming formulation for the problem and experiments to verify the quality of the proposed formulation. As far as we know, this is the first ILP formulation for the influence blocking class of problems. ILP formulations are only known for problems with a single information flow [Fischetti et al., 2018; Ghayour et al., 2019], but without considering two opposite flows, that occurs when modeling misinformation situations.

The rest of this paper is organized as follows. Section 2 contains the related work. Section 3 presents the diffusion models and the model we call the Simplified Competitive Linear Threshold Model, which we use in this work. Section 4 presents the problem definition we deal with, as well as the process of deriving the definition from the problem definition originally presented by [He et al., 2012]. Section 5 presents the formulation of the integer linear program that models the problem, and the proof of the correctness of such formulation. Section 6 describes the computational experiments performed, and discusses the results. We conclude the paper in Section 7.

2. Related Works

Various models have been developed to capture the characteristics of real-world information propagation. Among them, the independent cascade and linear threshold models, originally described by [Kempe et al., 2003], are well-known. To address influence propagation problems under these models, several techniques have been proposed. Approximation algorithms, such as CELF [Leskovec et al., 2007] and CELF++ [Goyal et al., 2011], exploit submodularity to find near-optimal node selections for the Influence Maximization problem. To significantly reduce CELF's runtime, [de Melo, 2021] develops a preprocessing heuristic algorithm. Mathematical programming methods were used in [Fischetti et al., 2018] and [Ghayour et al., 2019]. The former introduces the Generalized Least Cost Influence Propagation (GLCIP) and an integer linear programming (ILP) for this problem, while the latter proposes ILP formulations to handle the Influence Maximization

(IM) and Target Set Selection (TSS) problems. [He et al., 2012] introduced a new problem called Influence Blocking Maximization (IBM), which was studied under the Competitive Linear Threshold (CLT) model, considering two competing information flows within the network. The authors developed a heuristic algorithm to address the IBM problem.

While there are other studies that tackle the IBM problem, none of them present an ILP approach to address it with competing cascades. Therefore, in this study, we introduce what we believe to be the first ILP formulation for this problem. Given the different variations of the IBM problem that have been studied, we believe that our work can serve as a valuable foundation for developing ILP models to tackle these variations.

3. Diffusion Models

In real world situations, information is transmitted through a variety of channels, including social media, television, and face-to-face conversations. The precise way by which a particular idea spreads and the factors that drive individuals to accept or reject it can be difficult to discern. Nevertheless, diffusion models aim to capture the dynamics of information propagation in social networks, providing valuable mathematical frameworks for studying influence dissemination, despite limitations in capturing real-world nuances. Next, we present two important diffusion models and a simplified version we use for our model.

3.1. Linear Threshold Model

The Linear Threshold (LT) Model, as defined by [Kempe et al., 2003], models a social network as a directed graph $G = (V, E)$, where V represents the set of vertices (individuals) and E represents the set of edges (connections) between them. Each edge (u, v) in the graph connecting vertices u and v is assigned a weight $w_{u,v}$. Additionally, each vertex v is assigned a threshold value, t_v, that is chosen uniformly at random from the interval $[0, 1]$. In this model, a vertex, or node, can either be active or inactive, depending on whether it has been influenced by information or not, respectively. In this model, a node v is considered influenced if the sum of the weights of its incoming neighbors surpasses its threshold, i.e.,

$$\sum_{\forall (u,v) \in E} w_{u,v} > t_v.$$

The diffusion process starts with an initial set of nodes carrying the information. At each discrete time step, an inactive node becomes active if the sum of the weights of its active in-neighbors exceeds its threshold. The diffusion process continues until the entire network is influenced.

3.2. Competitive Linear Threshold Model

The Competitive Linear Threshold (CLT) Model is an extension of the LT model, which considers two competing ideas for influence in the network, represented by positive and negative diffusion, as proposed in [He et al., 2012]. Once a node is influenced by either diffusion, it cannot change the value of its activation (positive or negative) and be activated by the opposite behavior. Similar to the LT model, in CLT we have negative and positive seed sets, denoted by S^- and S^+, respectively. Additionally, each node in the network has positive and negative thresholds, denoted by t_v^- and t_v^+, and each edge has positive and negative weights, denoted by w_v^- and w_v^+.

At each time step, negative and positive influences propagate independently using negative thresholds and weights, and positive thresholds and weights, respectively. Since a node can only be activated by one diffusion, if both negative and positive thresholds are exceeded, the negative diffusion is assumed to win, and the node is negatively activated.

3.3. Simplified Competitive Linear Threshold Model

In this work, we use a simplified version of the Competitive Linear Threshold (CLT) model, which we refer to as the Simplified Competitive Linear Threshold (SCLT) model. Let $\delta_{\text{in}}(v)$ be the number of in-neighbors of a node v. In the SCLT model, we set both the positive and negative thresholds to half of the number of incoming neighbors for each node, i.e., $t_v^- = t_v^+ = \lfloor \delta_{\text{in}}(v)/2 \rfloor$. Such value is often referred to as the majority threshold [Chen, 2009]. Moreover, all edge weights are assigned a fixed value of 1. This simplification allows us to focus on the fundamental aspects of the diffusion model.

4. Problem Definition

In this section, we define the problem we deal with in this article, called the *Influence Blocking Maximization in the SCLT model*. However, before we define our problem, we present the definition of the original problem under the CLT model, as done by [He et al., 2012]. The change is only in the diffusion model, from CLT to SCLT, but this changes the definition of the problem, so that the simplest way to understand how this change occurs is by presenting the original problem first, and only then pointing out the modifications.

Let S^-_{end} be the set of negative nodes at the end of the diffusion process, and $s^-_{\text{end}} = |S^-_{\text{end}}|$. In the CLT model, as presented by [He et al., 2012], the thresholds values are set according to a probability distribution. Thus, the value s^-_{end} is a random variable. Clearly, s^-_{end} depends on the fixed input data and also the chosen solution S^+, which is the only variable. Thus, our notation only makes explicit the dependency on S^+, and we write $\Pr\big(s^-_{\text{end}} = \ell \,\big|\, S^+\big)$ to denote the conditional probability that the set S^-_{end} has size ℓ given that the set S^+ was chosen as the solution. Given a solution S^+, the expected size of the negative nodes s^-_{end} is,

$$\mathbb{E}\big[s^-_{\text{end}} \mid S^+\big] = \sum_{\ell=0}^{|V|} \ell \cdot \Pr\big(s^-_{\text{end}} = \ell \,\big|\, S^+\big).$$

To measure the impact of a solution S^+, [He et al., 2012] consider the difference between two scenarios, when the solution is indeed S^+, and when the solution set is empty (i.e., letting the negative spread without blocking it). This is called the *expected blocked negative influence* of S^+, and is formally defined as

$$\sigma(S^+) = \mathbb{E}\big[s^-_{\text{end}} \mid \{\emptyset\}\big] - \mathbb{E}\big[s^-_{\text{end}} \mid S^+\big],$$

and we want to maximize this quantity. We can now define the problem, as presented by [He et al., 2012].

Problem (Influence Blocking Maximization in the CLT model [He et al., 2012])**.**
Input: graph $G = (V, E)$ with thresholds t_v^+ and t_v^- for each $v \in V$, weights w^+ and w^-, a negative seed set S^-, and a positive integer k.
Output: a positive set of nodes S^+ of size k such that maximizes $\sigma(S^+)$.

It has been proven by [He et al., 2012] that IBM is NP-hard under the CLT model.

In this paper we consider the same problem, but in the SCLT model. This causes some modifications to the original definition. First, the positive and negative thresholds are the same, so we can use a unified t_v notation. Also, the thresholds are fixed at $\lfloor \delta_{\text{in}}(v)/2 \rfloor$. This means that the optimization function become deterministic rather than probabilistic. Thus, we replace the mathematical expectation with deterministic functions $s^-_{\text{end}}(\{\emptyset\})$ and $s^-_{\text{end}}(S^+)$. Note that $s^-_{\text{end}}(\{\emptyset\})$ becomes a fixed deterministic value, which can be computed in polynomial time (just simulate the diffusion process without selecting positive nodes for the solution). So, if before the objective was to maximize the difference $s^-_{\text{end}}(\{\emptyset\}) - s^-_{\text{end}}(S^+)$ but the first term is fixed, now it is the same as maximizing $-s^-_{\text{end}}(S^+)$. Another equivalent way is to say that we want to minimize $s^-_{\text{end}}(S^+)$, i.e., the objective is to choose S^+ in order to minimize the negative spread. Now we are ready to define the problem we deal with in this article.

Problem (Influence Blocking Maximization in the SCLT model)**.**
Input: graph $G = (V, E)$ with thresholds $t_v = \lfloor \delta_{\text{in}}(v)/2 \rfloor$ for each $v \in V$, a negative seed set S^-, and a positive integer k.
Output: a positive set of nodes S^+ of size k that minimizes the negative spread.

5. Integer Linear Programming Formulation

This section presents an integer linear programming formulation to solve the IBM Problem in the SCLT diffusion model. The key idea of the formulation is to use the concept of time as a step-by-step way of simulating the diffusion process. To achieve this, we introduce the variable a_v. This variable denotes the elapsed time since the activation of node v. This ensures that the activation time of the source node is always greater than that of the target node when one node activates another. Thus, a node v can only influence u if $a_v > a_u$, otherwise node v would not have been activated yet. By managing activation times in this manner, we can effectively handle cascades and prevent the occurrence of cycles. Next, we present the ILP formulation.

Input Parameters

G		the input graph $G = (V, E)$
S^-_v	$v \in V$	node v belongs to S^-
t_v	$v \in V$	threshold of node v
k		the number of nodes to be selected for S^+

Variables

S^+_v	$v \in V$	node v belongs to S^+
x^-_v	$v \in V$	node v is negatively activated
x^+_v	$v \in V$	node v is positively activated
$y^-_{u,v}$	$(u, v) \in E$	u exerts negative influence over v
$y^+_{u,v}$	$(u, v) \in E$	u exerts positive influence over v
a_v	$v \in V$	elapsed time since the activation of node v

Objective Function

$$\min \sum_{v \in V} x^-_v \tag{1}$$

Constraints

$$x_v^- \geq 1 - \frac{t_v}{\sum\limits_{u \in \text{in}(v)} y_{u,v}^-} \qquad \forall v \in V \tag{2}$$

$$x_v^+ - S_v^+ < \frac{\sum\limits_{u \in \text{in}(v)} y_{u,v}^+}{t_v} \qquad \forall v \in V \tag{3}$$

$$a_u + n(1 - y_{u,v}^- - y_{u,v}^+) > a_v \qquad \forall (u,v) \in E \tag{4}$$

$$y_{u,v}^- \leq x_u^- \qquad \forall (u,v) \in E \tag{5}$$

$$y_{u,v}^+ \leq x_u^+ \qquad \forall (u,v) \in E \tag{6}$$

$$\sum_{v \in V} S_v^+ = k \tag{7}$$

$$S_v^- \leq x_v^- \qquad \forall v \in V \tag{8}$$

$$S_v^+ \leq x_v^+ \qquad \forall v \in V \tag{9}$$

$$a_v + S_v^+ + S_v^- \geq 1 \qquad \forall v \in V \tag{10}$$

$$y_{u,v}^+ + y_{u,v}^- = 1 \qquad \forall (u,v) \in E \tag{11}$$

$$x_v^+ + x_v^- = 1 \qquad \forall v \in V \tag{12}$$

$$x_v^+, x_v^- = \{0,1\} \qquad \forall v \in V \tag{13}$$

$$y_{u,v}^+, y_{u,v}^- = \{0,1\} \qquad \forall (u,v) \in E \tag{14}$$

$$S_v^+, S_v^- = \{0,1\} \qquad \forall v \in V \tag{15}$$

$$a_v = \{0, 1, 2, ..., n\} \qquad \forall v \in V \tag{16}$$

The correctness of the presented formulation is not obvious, thus Theorem 2 presents a proof of the correctness. Before presenting Theorem 2, we need a technical lemma, presented in Lemma 1, which is used later in the proof of Theorem 2.

Lemma 1. *If $\sum_{u \in in(v)} y_{u,v}^+ > t_v$, then the ILP formulation sets $x_v^+ = 1$.*

Proof. The SCLT Model sets the threshold to be half the number of the incoming edges. So,

$$\sum_{u \in \text{in}(v)} y_{u,v}^+ > t_v = \frac{\delta_{\text{in}}(v)}{2}. \tag{17}$$

Furthermore, as all nodes are expected to be activated,

$$\sum_{u \in \text{in}(v)} y_{u,v}^+ + \sum_{u \in \text{in}(v)} y_{u,v}^- \leq \delta_{\text{in}}(v).$$

Rearranging,

$$\sum_{u \in \text{in}(v)} y_{u,v}^- \leq \delta_{\text{in}}(v) - \sum_{u \in \text{in}(v)} y_{u,v}^+.$$

Combining with Inequality (17),

$$\sum_{u \in \text{in}(v)} y_{u,v}^- \leq \delta_{\text{in}}(v) - \sum_{u \in \text{in}(v)} y_{u,v}^+ < \delta_{\text{in}}(v) - \frac{\delta_{\text{in}}(v)}{2} = \frac{\delta_{\text{in}}(v)}{2}$$

i.e., $\sum_{u \in \text{in}(v)} y^-_{u,v} = \delta_{\text{in}}(v)/2$. So, the right side of Constraint (3) becomes 0, meaning that x^-_v can be either 0 or 1. However, the objective function minimizes the negative nodes, so $x^-_v = 0$. By Constraint (12), we conclude that $x^+_v = 1$. $\square$

Theorem 2. *The ILP formulation presented correctly models the IBM Problem.*

Proof. Constraint (2) states that a node is negatively activated when the number of income edges from negative neighbors exceeds its threshold. More specifically, when $\sum_{u \in \text{in}(v)} y^-_{u,v} > t_v$, the right-hand side of the inequality becomes a strictly greater than 0. Since x^-_v is a binary variable, it must be set to 1 to satisfy the constraint, indicating the negative activation of node v. So, this constraint ensures that the activation condition is met when there is a sufficient number of negative neighbors contributing to the activation of node v. Constraint (3) specifies that for a node to be positively activated, it must meet either one of two conditions. Firstly, the number of incoming edges from its positive neighbors must exceed its threshold. Secondly, it must be included in the positive seed set. If neither of these conditions is met, the node is not positively activated. Note that, when $\sum_{u \in \text{in}(v)} y^+_{u,v} > t_v$, node v should be activated, i.e. $x^+_v = 1$. However, by the formulation, the right-hand side of the inequality becomes greater than 1, allowing x^+_v to be either 0 or 1. That will not be a problem as, according to Lemma 1, x^+_v will be set to 1. Constraint (4) stipulates that if node v is activated by node u, then $a_v < a_u$. This ensures that the dissemination cascades remain acyclic, as a node cannot be activated by another node with a lower activation time. Constraints (5) and (6) ensure that a node can only be activated if its source node is also activated. By Constraint (7), k nodes should be selected to be in S^+. Constraints (8) and (9) require seed nodes to be activated. Constraint (10) determines that if a node is not in any seed set, its activation time has to be greater than 1. Constraint (11) states that an edge can only belong to either the negative propagation or the positive propagation, but not both. Constraint (12) says that a node can only be negative or positive, but not both. Constraints (13), (14), and (15) guarantee that x^-_v, x^+_v, $y^-_{u,v}$, $y^+_{u,v}$, S^-_v and S^+_v are binary variables. Constraint (16) determines that a_v is an integer variable not greater than n. $\square$

6. Computational Experiments

In this section, we describe our experiments and discuss the obtained results.

6.1. Runtime Environment

Our computational experiments were run in an Intel Core i7-8565U 1.80GHz with 12GB of RAM, using Gurobi Optimizer 9.1.2 as the underlying LP-solver. To generate the graphs used in the experiments, we employed the NetworkX library (version 2.5.1) [Hagberg et al., 2008] in Python (version 2.7.17).

6.2. Metrics

We performed experiments to evaluate the proposed formulation. The metrics we consider are the *maximum and average empirical integrality gap* and the *running time to obtain an integer solution* using a generic branch-and-bound technique already provided by the solver.

The *integrality gap* for a minimization problem is defined as the maximum ratio between the solution value of the integer program and of its relaxation. The integrality gap is an interesting metric as it suggests how tight the relaxed formulation is compared to the convex hull of the integer solutions. Also, in some cases, the integrality gap translates into the approximation ratio of an approximation algorithm (e.g. when using variable rounding) [Young, 1995; Raghavan e Tompson, 1987]. The integrality gap must be obtained analytically, but this is not always possible or simple, so we perform an empirical analysis of it. Furthermore, the integrality gap is a worst case definition, but "typical" cases may better reflect reality. This justifies our empirical *average* analysis of the integrality gap.

The running time when using a generic branch-and-bound technique serves as a baseline for other solutions, as it is an upper bound that must not be exceeded. We note that several integer linear programming solution methods can be used on our formulation, but such methods are not within the scope of this work, our focus is on the quality of the formulation itself.

6.3. Instances

We use power-law graphs obtained by the Barabási-Albert model [Barabási e Albert, 1999] generated by NetworkX library as input for the Gurobi solver. Since graphs representing real networks (complex networks) are sparse, then the graphs were created with average degree 5. Each edge in the graphs was randomly assigned a direction. Additionally, each node had a 10% chance of being selected as part of the set S^-. The parameter k, determining the size of S^+, was set to half the size of S^-. The idea of using synthetic graphs in the experiments allows us to compute the average of the proposed metrics, in addition to making it possible to measure the scalability of our formulation.

6.4. Results and Discussion

We solved the ILP and compared its results with the linear relaxation, where the variables in the model are allowed to take continuous values. The linear relaxation is useful to evaluate the quality of the ILP formulation, either through the execution time or the integrality gap. More specifically, we compute the average and maximum integrality gap, as well as the average time required for each model to solve the problem. For each graph size, we perform 10 executions, and for each execution, a new graph is generated. In order to obtain more reliable results, we disabled the preprocessing, cutting planes and heuristics in Gurobi, relying only on the default branch-and-bound scheme, which is the standard technique for Gurobi's integer programming solution.

Table 1: ILP and linear relaxation performance.

	Average execution time (s)		Integrality gap	
Number of vertices	ILP	Linear relaxation	Maximum	Average
50	0.0501238	0.0335631	1.24138	1.02413
75	0.0577051	0.0394926	1.31506	1.07191
100	0.0898999	0.0523067	1.07692	1.0224
125	0.133355	0.0614975	1.10344	1.03172
150	0.248824	0.0811536	1.09164	1.05086
200	0.262325	0.0996094	1.0844	1.03416
400	0.316307	0.221297	1.06194	1.01856
800	1.07569	0.517779	1.0909	1.04631
1600	2.82034	2.17428	1.07035	1.0434
3200	5.85123	5.83748	1.05813	1.04229
6400	19.3689	18.7263	1.04426	1.03945
12800	163.72	134.685	1.05726	1.04090

The results presented in Table 1 demonstrate that the ILP consistently produces solutions that are similar to the relaxation. The average gap remained consistently close to 1 for all instance sizes, never exceeding 1.1. Furthermore, on average, the maximum gap is only 7% larger than

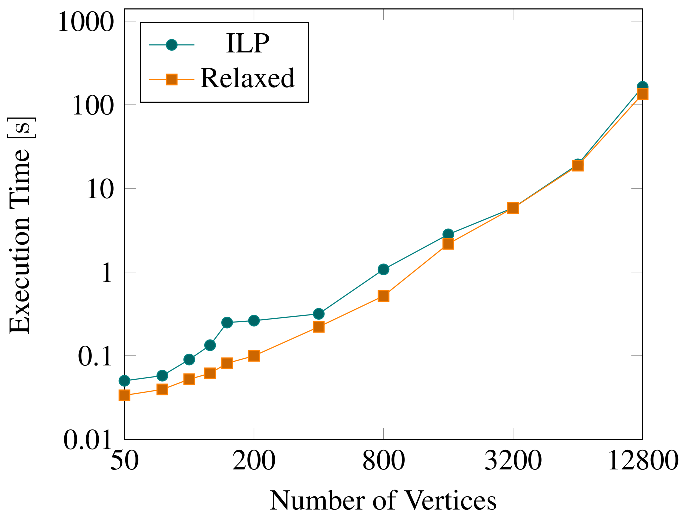

Figure 1: ILP and linear relaxation average execution time

the corresponding average gap, indicating that the ILP formulation performs well and maintains a relatively small variation in the quality of solutions across different instances.

In Figure 1 , we plot the values of Table 1 related to the execution time in order to get a better view, showing a comparison of the execution times between the ILP and the relaxed model. It suggests an exponential growth pattern as the graph size increases. On average, the relaxed model was approximately 79% faster than the ILP model. However, this difference tends to decrease for larger instances.

7. Conclusion

In this work, we present a version of the Influence Blocking Maximization problem described under the SCLT model, and we present an integer linear programming formulation for the problem. Our experimental results demonstrate that the formulation performs well across a wide range of instances, including both small and large ones, keeping a close proximity to the results obtained using its relaxed version.

There is no trivial or direct way to obtain an integer linear programming formulation for this problem, and we believe that this can serve as a starting point for the study of problems related to misinformation from the point of view of mathematical programming. This naturally motivates future directions, such as alternative formulations for this problem or variants, the use of integer linear program solving techniques for scalability and efficiency, and theoretical analyzes in polyhedral combinatorics.

References

Allcott, H. e Gentzkow, M. (2017). Social media and fake news in the 2016 election. *Journal of Economic Perspectives*, 31:211–236.

Barabási, A.-L. e Albert, R. (1999). Emergence of scaling in random networks. *Science*, 286(5439): 509–512.

Chen, N. (2009). On the approximability of influence in social networks. *SIAM Journal on Discrete Mathematics*, 23(3):1400–1415.

Cinelli, M., Quattrociocchi, W., Galeazzi, A., Valensise, C. M., Brugnoli, E., Schmidt, A. L., Zola, P., Zollo, F., e Scala, A. (2020). The covid-19 social media infodemic. *ArXiv*, abs/2003.05004.

de Melo, R. S. (2021). *Exact Algorithms For Influence Propagation In Complex Networks*. PhD thesis, Department of Computer Science, Federal University of Paraná.

Fischetti, M., Kahr, M., Leitner, M., Monaci, M., e Ruthmair, M. (2018). Least cost influence propagation in (social) networks. *Mathematical Programming*, 170(1):293–325.

Ghayour, F., Baghbani, Asadpour, M., e Faili, H. (2019). Integer linear programming for influence maximization. *Iranian Journal of Science and Technology, Tran. of Elec. Eng.*, 43(3):627–634.

Goyal, A., Lu, W., e Lakshmanan, L. V. (2011). Celf++: Optimizing the greedy algorithm for influence maximization in social networks. In *Proc. of the 20th Int. Conf. Companion on WWW*, p. 47–48.

Hagberg, A. A., Schult, D. A., e Swart, P. J. (2008). Exploring network structure, dynamics, and function using networkx. In Varoquaux, G., Vaught, T., e Millman, J., editors, *Proceedings of the 7th Python in Science Conference*, p. 11 – 15, Pasadena, CA USA.

He, X., Song, G., Chen, W., e Jiang, Q. (2012). Influence blocking maximization in social networks under the competitive linear threshold model. In *Proc. of the 2012 SIAM Int. Conf. on Data Mining (SDM)*, p. 463–474.

Kempe, D., Kleinberg, J., e Tardos, E. (2003). Maximizing the spread of influence through a social network. In *Proc. of the 9th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, p. 137–146.

Lazer, D. M. J., Baum, M. A., Benkler, Y., Berinsky, A. J., Greenhill, K. M., Menczer, F., Metzger, M. J., Nyhan, B., Pennycook, G., Rothschild, D., Schudson, M., Sloman, S. A., Sunstein, C. R., Thorson, E. A., Watts, D. J., e Zittrain, J. L. (2018). The science of fake news. *Science*, 359 (6380):1094–1096.

Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., VanBriesen, J., e Glance, N. (2007). Cost-effective outbreak detection in networks. In *Proc. of the 13th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, p. 420–429.

Lewandowsky, S., Ecker, U. K. H., Seifert, C. M., Schwarz, N., e Cook, J. (2012). Misinformation and its correction: Continued influence and successful debiasing. *Psychological Science in the Public Interest*, 13(3):106–131.

Raghavan, P. e Tompson, C. D. (1987). Randomized rounding: A technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7(4):365–374.

Vosoughi, S., Roy, D., e Aral, S. (2018). The spread of true and false news online. *Science*, 359 (6380):1146–1151.

Young, N. E. (1995). Randomized rounding without solving the linear program. In *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, p. 170–178. Society for Industrial and Applied Mathematics.