Turkish Journal of Electrical Engineering and Computer Sciences

Volume 26 | Number 6 Article 46

1-1-2018

Influence maximization in social networks: an integer programming approach

MUHAMMED EMRE KESKİN

MEHMET GÜRAY GÜLER

Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2018) 26: 3383 – 3396

doi:10.3906/elk-1802-212

Influence maximization in social networks: an integer programming approach

M. Emre KESKİN[1*], **Mehmet Güray GÜLER**[2]

[1]Department of Industrial Engineering, Faculty of Engineering, Atatürk University, Erzurum, Turkey
[2]Department of Industrial Engineering, Faculty of Mechanical Engineering, Yıldız Technical University, İstanbul, Turkey

Received: 27.02.2018 • **Accepted/Published Online:** 15.10.2018 • **Final Version:** 29.11.2018

Abstract: The use of social networks has been spreading rapidly in recent years. There is a growing interest in influence maximization in social networks, especially after observing that the effects of social events of the Arab Spring, Gezi events of Turkey, uprising in Ukraine, etc. have been built by the help of social networks. Consequently, many institutions like political parties or commercial firms are willing to spread their messages throughout social networks. There are many studies that concentrate on finding the most influential initial nodes, called seeds, which maximize the spread of an intended message over the social network. However, most of these works provide numeric algorithmic methods without including an integer program that seeks for a theoretical optimal point. Integer programs, on the other hand, are provided in very few studies, and they mostly assume an independent cascade model, which is a diffusion model depending on probabilistic affection rates, to formulate the diffusion in the network. In this study, we first provide a basic integer program that works under a linear threshold model, which is a diffusion model assuming threshold affection levels, and extend it for the situation in which there is a competing opinion (like black propaganda for a product, an event, or an opinion). Finally, we provide heuristic solution procedures and efficiency analysis with extensive numerical instances.

Key words: Social networks, influence maximization, integer programming

1. Introduction

A social network is a platform on which people pursue social interactions depending on their personal relationships. Use of social networks has been spreading rapidly in recent years. The prevalence of social networks is still increasing thanks to the spread of mobile phones. Moreover, the tremendous popularity of social networks especially among young generations indicates that the effect of social networks on society will be even larger in the future. Social events of the Arab Spring, Gezi events of Turkey, uprising in Ukraine, etc. have been built by the help of social networks: ideas propagate rapidly through the networks. Consequently, political parties and commercial firms are willing to spread their messages, i.e. their marketing activities, throughout social networks.

There are several studies that showed the significance of selecting the initial seed. For example, [1] shows that a carefully selected group of people can increase the influence spreading in a cascade up to eight times.

Moreover, it was empirically shown that a financially viable cascade initiation requires at most 0.2% of the nodes in a network [2]. This raises the following question: which individuals in a network should be targeted at the beginning in order to maximize the adoption of a product or an opinion? The objective of the influence

*Correspondence: emre.keskin@atauni.edu.tr

maximization (IM) problem is to identify a subset of initial nodes to be targeted in a network to maximize the influence propagation. Although viral marketing campaigns are well-known examples of IM problems [3], social networks are used in many other fields like health care [4], computer networks [5], sociology [6], or politics [7].

Choosing the influential nodes at the beginning yields a high expected number of activated nodes at the end of the process. The first attempts to find the best influential nodes depended on centrality measures [8, 9]. Centrality measures identify key nodes in a network, such as a hub or a bridge. A hub is a node that has a large number of direct connections (distance centrality). A bridge, on the other hand, has short paths to other nodes (degree centrality). Centrality heuristics assume that hubs or bridges are the most influential nodes in a network. However, studies showed that this assumption fails [10].

The work in [11] revisits the IM problem using a different approach. The motivation is that, in marketing decisions, companies decide to target a customer by evaluating the customer's individual profit. However, people are influenced by their peers, and hence the network value, which is the profit from sales to other customers that she may influence to buy, may have a significant effect on the profitability of the customer, which gives rise to viral marketing. The authors modeled the problem using a random Markov field. The probability that a customer is influenced depends on whether her connections are influenced or not. They provided algorithms to mine the network in order to find the probabilities and to find the optimal set of seeds to maximize the spread in the network.

In the seminal paper [12], IM was formulated as a discrete optimization problem. The authors provided two fundamental diffusion models: the linear threshold (LT) model and the independent cascade (IC) model. In the IC model, time is divided into rounds and influence of the nodes on neighboring nodes is represented by influence probabilities. At each round, the affected nodes propagate the message to their neighbors and the neighbors are affected by the message with the mentioned influence probability. The process continues until the end of the planning horizon. In the LT model, on the other hand, each node has its own threshold level and a node can be affected only if the total propaganda coming from its previously affected neighbors exceeds the influence threshold of the node. The authors showed that in both models the problem is NP-hard. Using the submodularity property of the objective function, they also proved that the greedy algorithm is 63% away from the optimal solution. Several algorithms followed that of [12]; for example, [13] provided another greedy algorithm that is 700 times faster than that of [12].

The greedy algorithm does not guarantee an optimal solution. Therefore, the solution quality and the properties of the optimal solution cannot be characterized unless the optimal solution is identified. Hence, the mixed integer programming (MIP) models are significant in terms of assessing the quality of the proposed algorithms with respect to the optimal solution. It turns out that the associated MIP models in the literature are limited. In this study, we focus on modeling the problem using MIP and give the associated literature.

In general, the IM problem has three dimensions [14]. The first is the number of the nodes to be selected at the beginning. These nodes, the seeds, are active at the beginning. Being active means that they have the product or adopt an innovation and hence influence their neighbors throughout the network. The second dimension is the number of activated nodes at the end of the propagation. In a marketing context, this can be the number of people who buy the product at the end of the marketing campaign. The last dimension is the duration of the propagation. One of the earliest studies that dealt with the IM problem with MIP was [15]. The authors' aim was to choose the initial seed such that the entire graph will be activated eventually. In terms of the dimensions given above, in [15], the time is set free, and the coverage should be the whole network while minimizing the cost of the initial seed. In [12], on the other hand, the aim is to maximize the number of

activated people at the end by limiting the nodes at the beginning (using a limited budget) and an unlimited time horizon. The work in [16] provided a two-stage stochastic programming approach for both LT and IC models, which outperforms the greedy heuristic of [12]. Meanwhile, [17] proposed a biobjective probabilistic integer programming model to minimize the initial seeds while maximizing the coverage with unlimited time. The authors modeled the diffusion of SMS texting in a social network of students at a university. The work in [18] formulated the IC model using a MIP model and solved an approximation of it using a sample average approximation method.

In the IM problems cited above, selected consumers at the beginning are given the product gratis. However, an individual can be partially influenced by the use of monetary inducements. For example, [19] and [20] dealt with the problem where the goal is to find the minimum total amount of inducements required to influence a given proportion of the population, and [21] provided a generalized model for the problems in [19] and [20].

There are also several extensions using MIP models in social networks. For example, [22] focused on finding the most critical paths in a social network for the influence propagation process and [23] studied the adding of nodes or arcs to maximize the influence propagation. The authors of [24] developed a MIP model to minimize influence in a network.

Two basic diffusion models representing the propagation of an idea throughout a network are the IC model and the LT model. Furthermore, most IC and LT model-based seed selection problems assumed no competition. However, in real-world settings, there can be collective influence [25] or competing opinions in a network [26]. In some applications, an enemy party is also introduced, which tries to spread its own message. Nodes are supposed to be neutral, positively affected, or negatively affected, which implies that a node cannot be affected by the initial message-spreading party and the enemy party simultaneously. Hence, the initial party and the enemy party are in a competition to reach as many nodes as possible in such situations. There are many studies that concentrated on finding the most influential seed nodes that maximize the spread of an intended message over the social network. However, most of them provided numeric algorithmic methods without providing an integer program that seeks for a theoretical optimal point. Integer programs are provided in a very few studies, and they mostly assume the IC diffusion model. Recently, [27] and [28] studied the IM problem where there is a competing influence in the network. They modeled the problem using a new diffusion model called a parallel cascade. Both studies used a MIP model to formulate the problem. A comprehensive review of information diffusion in networks can be found in [29].

In this study, we provide a basic integer program that works under the LT diffusion model and extend it for the situation in which there is also an enemy party spreading black propaganda in the social network. Therefore, this study is the first attempt to provide an integer programming formulation that models the influence maximization in social networks under a linear threshold model. Integer programming formulation provides a premise in which theoretical upper bounds are also produced. Hence, integer programming formulations make it possible to assess the quality of the feasible solutions (produced by any heuristic method) in terms of the percent deviations from the theoretical upper bounds. Finally, we provide a heuristic solution procedure in order to compensate the efficiency loss of the commercial solvers for the large networks and prove the efficiency of the heuristic with extensive numerical instances. We also report the deviations of the heuristic solutions from the theoretical upper bounds provided by the solver at the end.

2. Mathematical formulations

In this section, we first provide a brief subsection in which we go through integer programming formulations and general solution strategies for the interested readers. Later on, we give the formulations of the integer programs.

2.1. Integer programming

A linear program can be written as $\max/\min\{cx : Ax \leq b, x \geq 0\}$ and can be optimally solved by methods such as the simplex method ([30]) or Karmarkar algorithm ([31]). If we add the restriction that the variables must take integer values, the linear program becomes an integer program. General integer programming is known to be an NP-complete problem, implying that we must resort to heuristic procedures for the solution of the moderate and large-sized problem instances.

There are two basic exact solution procedures that are often used for the solution of the integer programs: branch-and-bound ([32]) and cutting plane algorithms ([33]). Branch-and-bound methods depend on the idea of solving the linear relaxation (neglecting the integrality restriction on the values of the variables giving the linear relaxation) of the integer program in a repetitive manner by branching on variables having nonintegral values and using the bounds (obtained from the objective value of the linear relaxation) for trimming the branches of the branch-and-bound tree. Namely, after solving the linear relaxation of the integer program at a node of the branch-and bound tree, a variable having nonintegral value, say x_i^*, is chosen and two child nodes are added to the tree, for the first of which additional constraint $x_i \leq \lfloor x_i^* \rfloor$ is employed and for the second of which additional constraint $x_i \geq \lceil x_i^* \rceil$ is employed. If there is no variable having nonintegral value after the linear relaxation solution of the node, then the branch is pruned due to the optimality. If a node is infeasible, then the branch is pruned due to infeasibility, and if the objective function value of linear relaxation at a node is worse than the objective function value of the best feasible solution at hand, then the branch is pruned by the bound. The procedure is continued until all the branches are pruned, and the best feasible solution at the end of the procedure is guaranteed to be the optimal solution of the integer program. On the other hand, cutting plane algorithms depend on the idea of approximating the convex hull (which is the smallest convex set including the feasible region of the program) of the integer program by adding valid constraints to the formulation of the program. Theoretically, it is possible to add valid constraints to the formulation of any integer program so that the corner points of the linear relaxation of the program consist of only integral values, implying that the solution of the linear relaxation program will be optimal for the integer program as well. In practice, linear relaxation programs are solved repetitively until an integral solution is obtained. If the optimal solution of the linear relaxation program contains nonintegral values, a separation problem, in which a valid constraint is sought that cuts the nonintegral solution, is solved, and the new formulation is obtained after addition of the new cut. A recent trend in exact solution procedures for integer programs depends on hybridizing the branch-and-bound method and cutting plane mechanisms, which are mostly known as branch-and-cut algorithms ([34]). If valid constraints are added as variables to the dual of the linear relaxation program, the subproblems formed in the body of the procedure are called pricing subproblems, which produce branch-and-price algorithms after integration with the branch-and-bound algorithm ([35]).

The main difference between a heuristic method and an exact solution procedure is that the exact solution procedure guarantees the theoretical optimal solution. On the other hand, a heuristic procedure does not prove or guarantee the optimality of the solution it reports, even if it is able to find the optimal solution. Today's modern integer programming solvers employ branch-and-bound methods and cutting plane algorithms using their huge set of valid cut libraries in an integrated manner and they are able to find the optimal solution in

tolerable computation times for most of the small-sized integer programs. On the other hand, it is possible to use the solvers as heuristic solution procedures by letting them run for an allowed computation time and report the best feasible solution they find. Finally, dual bounds obtained from integer programming make it possible to analyze the feasible solutions in terms of maximum percent deviations from the optimal solutions. We direct interested readers to the seminal work [36] about integer programming.

2.2. Formulations

The first integer program, which is referred to as the basic model, includes a single party trying to find the initial seed nodes to maximize the spread of a particular message. The second integer program, which is referred to as the competition model, extends the first one by introducing an enemy trying to spread its own message. Before providing the mathematical formulations, we describe the sets, parameters, and decision variables used in the model in Table 1.

Table 1. Sets, parameters, and variables used in the model.

Sets	
I	Set of the nodes (people) of the network
T	Set of time periods
I_i	Set of neighbors of node i
Parameters	
B	Budget allocated for selection of initial seeds
b_i	Cost of selecting node i in the initial seed
s_i	Influence threshold of node i
r_i	Negative influence threshold of node i
f_{ji}	Influence of node j on node i
g_{ji}	Negative influence of node j on node i
Decision variables	
x_{it}	Indicates whether or not node i is affected at period t
y_{it}	Indicates whether or not node i is negatively affected at period t

The basic model is given in the following:

$$\max \sum_{i \in I} x_{i|T|} \tag{1}$$

$$\sum_{i \in I} b_i x_{i0} \leq B \tag{2}$$

$$\sum_{j:i \in I_j} x_{j(t-1)} f_{ji} \geq s_i x_{it} \qquad i \in I, t \in T/\{0\} \tag{3}$$

$$\sum_{j:i \in I_j} x_{j(t-1)} f_{ji} \leq s_i + x_{it} \qquad i \in I, t \in T/\{0\} \tag{4}$$

$$x_{i(t-1)} \leq x_{it} \qquad i \in I, t \in T/\{0\} \tag{5}$$

$$x_{it} \in \{0,1\} \qquad i \in I, t \in T \tag{6}$$

In the objective function of Eq. (1), we maximize the number of affected people in the last period. Hence, the aim is to reach as many people as possible at the end of the planning horizon. The constraint of Eq. (2) puts a budget limit on the money spent for the people selected as the initial seeds. The constraint of Eq. (3) makes it sure that a person i cannot be affected at period t if the total positive flow from the neighbors that are already affected in the previous period is below his or her threshold level. On the other hand, if the positive flow from the neighbors exceeds the threshold, then the constraint of Eq. (4) forces person i to be affected at period t. The constraint of Eq. (5) ensures that a convinced person stays convinced as time passes. Finally, the constraint of (6) stands for the usual binary restrictions.

We give the formulation of the competition model in the following:

$$\max \sum_{i \in I} x_{i|T|} \tag{7}$$

$$\sum_{i \in I} b_i x_{i0} \leq B \tag{8}$$

$$\sum_{j:i \in I_j} x_{j(t-1)} f_{ji} \geq s_i x_{it} \qquad i \in I, t \in T/\{0\} \tag{9}$$

$$\sum_{j:i \in I_j} x_{j(t-1)} f_{ji} \leq (s_i + (1.1 - s_i) y_{it}) + x_{it} \qquad i \in I, t \in T/\{0\} \tag{10}$$

$$x_{i(t-1)} \leq x_{it} \qquad i \in I, t \in T/\{0\} \tag{11}$$

$$\sum_{j:i \in I_j} y_{j(t-1)} g_{ji} \geq r_i y_{it} \qquad i \in I, t \in T/\{0\} \tag{12}$$

$$\sum_{j:i \in I_j} y_{j(t-1)} g_{ji} \leq (r_i + (1.1 - r_i) x_{it}) + y_{it} \qquad i \in I, t \in T/\{0\} \tag{13}$$

$$y_{i(t-1)} \leq y_{it} \qquad i \in I, t \in T/\{0\} \tag{14}$$

$$x_{it} + y_{it} \leq 1 \qquad i \in I, t \in T/\{0\} \tag{15}$$

$$x_{it}, y_{it} \in \{0, 1\} \qquad i \in I, t \in T \tag{16}$$

Objective function Eq. (7) and constraints of Eqs. (8)–(9) of the competition model are similar to objective function Eq. (1) and constraints of Eqs. (2)–(3) of the basic model. The constraint of Eq. (10) plays the same role as the constraint of Eq. (4) of the basic model. Suppose node i is not negatively affected at period t, implying that $y_{it} = 0$, and the constraint of Eq. (10) reduces to the constraint of Eq. (4) of the basic model. On the other hand, if $y_{it} = 1$, node i is negatively affected at period t; the constraint of Eq. (4) becomes redundant since the right hand value $1.1 + x_{it}$ will always be larger than the total positive inflow from the neighbors. The constraint of Eq. (11) is the same as the constraint of Eq. (5) of the basic model. On the other hand, constraints of Eqs. (12)–(14) are similar to the constraints of Eqs. (9)–(11) but are written for negative affection. The constraint of Eq. (15) indicates that a node cannot be positively and negatively affected at the same time. Finally, the constraint of Eq. (16) puts the binary restrictions on the variables.

3. Solution methods

It is a well-known fact that general mixed integer linear programming problems belong to the NP-hard problem class and our formulations are no exception. If there is only one single period, i.e. affected nodes will solely constitute the initial seed, the basic model reduces to a binary knapsack problem, which is known to be NP-complete. Therefore, it can be said that even the simplest version of our problem is an extremely difficult problem since a very simplified version of the model is still NP-complete. The competition model can also be reduced to the basic model and hence can be shown to be an NP-hard problem. Consequently, there is almost no hope for finding a solution method that solves influence maximization instances in polynomial time. This is why we must resort to heuristic solution procedures for solving moderate and large-sized instances. We propose two practical and easy to implement solution procedures below.

3.1. Greedy approach

The first thing that comes to mind is to construct the initial seed from the nodes having the highest number of connections. However, this kind of node may have relatively weak influence on its neighbors. Indeed, it is shown that nodes with high centrality measures fail to maximize the influence in the network in some cases [10]. Hence, not only the number of connections but also the capability of the node to affect its neighbors should be taken into consideration while constructing the initial seed. One way of doing this is to sum up the influence parameter values of nodes on their neighbors and select the most influential ones until reaching the budget limitation. Another approach, which is implemented in this work, suggests sorting the nodes with respect to their influence per unit cost. Namely, the rate of total influence of the node over the cost of selecting the node in the initial seed can be calculated for each node and the initial seed can be formed by choosing the nodes having the highest rate without exceeding the budget limit. We summarize the details of the approach below.

Algorithm 1 Greedy approach.

Initialization: Let Cost = 0, Calculate $A_i = \frac{\sum_{j \in I_i} f_{ij}}{b_i}$ for $i \in I$
while (set of nodes is not empty) **do**

- Select node i having the highest A_i value from the set of nodes
- Remove node i from the set of nodes
- **if** $(Cost + b_i \leq B)$ **then**
  add node i to the initial seed and let $Cost = Cost + b_i$
- **end if**

end while

3.2. Period iteration approach

The number of variables of the models can be decreased to tolerable levels by setting a set of selected variables a priori before solving the model from scratch. Note that all the variables of the models have t indices, indicating that a possible decrease in the number of periods has great potential to reduce the number of variables. Number of periods can be set to a small integer like 1 or 2 to produce easy to solve small-sized models with low-quality optimal solutions. On the other hand, setting the number of periods to a large number makes the solution of the formulations very difficult. A compromise between setting the number of periods to very small numbers and setting it to large numbers can be obtained by first setting the number of periods to a low number and

then increasing it one by one. An improvement in the objective function value of the models is expected by a unit increase in the number periods, but this marginal improvement of unit period increments also tends to decrease by the number of periods, as well. Hence, the objective function value is expected to converge to its optimal value as we keep increasing the number of periods. We call this solution approach the period iteration approach (PIA) in the sequel. If the number of periods is limited to F, only the variables belonging to the period F or the earlier periods (x_{it}, y_{it} for $t \leq F$) are set free in the model while the variables belonging to later periods (x_{it}, y_{it} for $t > F$) are set to the values of x_{iF}, y_{iF}. After setting the values of the variables in that manner, they will become parameters in the models leading to smaller model sizes in terms of the number of the decision variables, which we call the restricted models in the sequel. Restricted models are run with the MIP solver Gurobi for a limited amount of time. Gurobi is expected to find optimal or near optimal solutions as the sizes of the models are limited. Moreover, it is possible to speed up the running process of Gurobi for a given value of F, by letting it start the running process from the solution found for the previous restricted model having $F-1$ number of periods, knowing that the solution of the model having $F-1$ number of periods is also feasible for the model having F number of periods. We summarize the details of the approach below.

Algorithm 2 Period iteration approach.

Initialization: Let $DIFF = 100$, $F = 1$, $OBJ_OLD = 0$
while ($DIFF > 0$) **do**

- Solve the restricted model for F number of periods and if $F > 1$ start the running process from the solution of the restricted model with $F-1$ number of periods to obtain OBJ
- Set $DIFF = OBJ - OBJ_{OLD}$
- Set $OBJ_{OLD} = OBJ$ and $F = F + 1$

end while

4. Numerical results

In this section, we first explain the generation of the model parameters and then we indicate the success of the offered heuristic procedures by comparing the number of positively affected nodes by the offered heuristic procedures and the commercial solver Gurobi on several sets of numerical instances.

4.1. Generation of model parameters

We study two different problem sets. We randomly generate the first one, and we take the second one from the literature. In the first set, we study 9 different problem instances with different sizes. Namely, we generate 9 instances having 20, 50, 100, 200, 300, 500, 750, 1000, and 1500 total nodes. Two nodes are accepted as neighbors; that is, they are connected in the network if the randomly generated number from the (0, 1) interval is larger than 0.5. This set is called set 1 in the sequel. The second set consists of the available data taken from arXiv, in which each node is an author and the arcs show the co-authorship relation. These data are from the same source used in the experimental studies of [12] and [3]. There are 4 problems in the set called 1k, 2k, 5k, and 10k including 378, 378, 1157, and 3107 nodes and 1022, 2000, 5000, and 10000 arcs, respectively. This set is called set 2 in the sequel.

On the other hand, positive and negative threshold levels of the nodes are randomly determined within the interval (0.3, 1). Next, positive and negative effects of the nodes on their neighboring nodes are randomly generated within the (0, 0.3) interval and if the summation of the total positive/negative effects on a node

exceeds 1, then these positive/negative effects are normalized so that they sum up to 1. Finally, the costs of selecting nodes in the initial seed are randomly generated from the (30, 100) interval and the total initial seed selection budget is determined as 30% of the total cost that would be incurred if all the nodes were selected as the initial seed.

4.2. Performance of the heuristics

In this subsection, we compare the number of positively affected nodes obtained at the end of the planning horizon by the greedy approach and by PIA with the ones found by the state-of-the-art commercial MIP solver Gurobi (http://www.gurobi.com) for the above-mentioned numerical instances. All methods are coded and run in C # language using Visual Studio using a single Intel Xeon 5460 core with 28 gigabytes of RAM. One hour of computation time is given for each of the instances for all the methods. If Gurobi finds the optimal solution before 1 h ends or the PIA or greedy approach converges to a solution before 1 h, they immediately report the best solution they find and start to work on the next instance. For set 1, we respectively tabulate the number of the affected nodes found and the total computation times used by Gurobi, PIA, and the greedy approach for the basic model in Table 2 and for the competition model in Table 3. Note that ZGUROBI, ZPIA, and ZGREEDY respectively stand for the number of affected nodes found by Gurobi, PIA, and the greedy approach while TGUROBI, TPIA, and TGREEDY represent the computation times they use.

Table 2. Number of affected nodes found and computation times used by Gurobi, PIA, and greedy approach for the basic model run for set 1.

	ZGUROBI	ZPIA	ZGREEDY	TGUROBI	TPIA	TGREEDY
20	**16**	16	15	0.34	0.37	0.08
50	**39**	38	25	479.29	552.03	0.08
100	66	71	49	3600.04	1283.89	0.22
200	142	168	129	3600.14	3000.98	0.72
300	136	169	148	3600.09	2102.05	2.56
500	187	282	250	3600.70	1509.35	12.62
750	266	428	365	3611.51	1531.96	46.44
1000	364	530	486	3627.32	2779.87	123.15
1500	0	761	739	3688.89	2119.31	519.18

Table 3. Number of affected nodes found and computation times used by Gurobi, PIA, and greedy approach for the competition model run for set 1.

	ZGUROBI	ZPIA	ZGREEDY	TGUROBI	TPIA	TGREEDY
20	**9**	9	6	0.27	0.14	0.08
50	**25**	25	21	0.12	0.15	0.07
100	**44**	43	37	4.60	0.60	0.31
200	88	89	77	3600.11	47.23	1.07
300	125	127	111	3600.21	603.24	4.05
500	214	216	181	3601.09	614.19	22.77
750	310	323	267	3603.26	649.92	80.98
1000	421	435	362	3608.29	722.82	201.43
1500	0	649	549	11711.01	2544.87	2247.11

Gurobi is able to find and prove the optimality for two instances of the basic model and for three instances of the competition model, which are written in bold characters in the ZGUROBI column in Table 2 and 3, respectively. It can be seen from Table 2 that Gurobi is more successful than PIA for only one instance, which is one of the smallest instances in which there are 50 nodes, while PIA outperforms Gurobi especially for large instances. Similarly, the greedy approach tends to perform relatively poorly for small instances but its success becomes clearer for large instances. When the number of nodes is increased to 1500, Gurobi is unable to find a feasible solution with more than 0 number of positively affected nodes for both basic and competition models, while both PIA and the greedy approach are able to find relatively good feasible solutions having a large number of positively affected nodes. This implies that using the commercial solver Gurobi as a solution strategy would not be practical for real-life network instances in which there are thousands of nodes. On the other hand, PIA is more successful than the greedy approach for all instances of both basic and competition models. However, performance of the greedy approach is very good for large instances, especially for the basic model, while PIA is clearly better for the competition model. In terms of computation times, the greedy approach is the most successful one. It uses less than 1 min for all the instances but two for the basic model and for all the instances but three for the competition model, while Gurobi uses the allowed 1 h entirely for most of the instances for both basic and competition models. Therefore, if a user wants to obtain good feasible solutions within small computation times, he/she may refer to the greedy approach, but if he/she wants the best, he/she has to choose the PIA at the expense of longer computation times. Although 1-h maximum computation times are allowed for each solution method, we have to wait until the method reports its solution. For large instances, it is possible for Gurobi to exceed the maximum allotted computation time since it checks the time after the transactions of the inner procedures and the transactions may last for long times. For instance, Gurobi uses more than 3 h for the largest instance of the competition model. Therefore, Gurobi is not only the least successful alternative for the large instances, but also it uses very large computation times. On the other hand, the performance of PIA with respect to computation time is in between the competition times of Gurobi and the greedy approach. Hence, PIA is able to find very good solutions in a tolerable amount of computation time.

We also tabulate the results of the Gurobi and the heuristics for set 2, as well, in Table 4 for the basic model and in Table 5 for the competition model.

Table 4. Number of affected nodes found and computation times used by Gurobi, PIA, and greedy approach for the basic model run for set 2.

	ZGUROBI	ZPIA	ZGREEDY	TGUROBI	TPIA	TGREEDY
1k	317	321	207	3600.07	772.69	0.80
2k	287	299	185	3600.12	1842.13	0.93
5k	909	917	535	3600.39	1812.52	9.32
10k	1827	2465	1494	3602.50	2248.43	149.49

As can be seen from Table 4 and Table 5, the greedy approach is the fastest but the least successful method. Results of PIA are at least as good as results of Gurobi for all instances and PIA is able to produce the results in smaller computation times on average (especially for the largest instance). Gurobi and PIA are able to find optimal solutions for all instances for the competition model.

Modeling the influence maximization in social networks as an integer program also provides the opportunity to assess the quality of the feasible solutions by comparing them with the theoretical upper bounds reported by the solver. Hence, the percent deviation values, which are respectively given by %GUROBI, %PIA,

Table 5. Number of affected nodes found and computation times used by Gurobi, PIA, and greedy approach for the competition model run for set 2.

	ZGUROBI	ZPIA	ZGREEDY	TGUROBI	TPIA	TGREEDY
1k	**248**	248	166	1.62	4.01	1.33
2k	**255**	255	186	1.79	4.18	1.48
5k	**827**	827	545	24.52	48.04	18.08
10k	**2165**	2165	1453	3052.37	500.99	345.32

and %GREEDY, are tabulated in Table 6 and Table 7, respectively. Note that percent deviations are calculated as $100 \times \frac{(\text{UB-Z})}{\text{UB}}$, where UB stands for the upper bound provided by the solver and Z represents the feasible objective function value given by the implied method. Namely, %GUROBI $= 100 \times \frac{(\text{UB-ZGUROBI})}{\text{UB}}$, %PIA $= 100 \times \frac{(\text{UB-ZPIA})}{\text{UB}}$, and %GREEDY $= 100 \times \frac{(\text{UB-ZGREEDY})}{\text{UB}}$.

Table 6. Percent deviations from the theoretical upper bounds for the basic model run for set 1.

	%GUROBI	%PIA	%GREEDY
20	0.00	0.00	6.25
50	0.00	2.56	35.90
100	24.14	18.39	43.68
200	19.77	5.08	27.12
300	48.68	36.23	44.15
500	56.10	33.80	41.31
750	58.89	33.85	43.59
1000	58.40	39.43	44.46
1500	100.00	49.27	50.73

Table 7. Percent deviations from the theoretical upper bounds for the competition model run for test 1.

	%GUROBI	%PIA	%GREEDY
20	0.00	0.00	33.33
50	0.00	0.00	16.00
100	0.00	2.27	15.91
200	8.33	7.29	19.79
300	19.35	18.06	28.39
500	15.75	14.96	28.74
750	21.72	18.43	32.58
1000	19.96	17.30	31.18
1500	100.00	56.73	63.40

It can be understood from Table 6 that the solver Gurobi starts with two consecutive successful results as the percent deviation is 0 for the two smallest instances. However, the percent deviations increase rapidly and

even reach 100% for the largest instance as Gurobi is unable to find a feasible solution for that instance. On the other hand, both PIA and GREEDY heuristics start from relatively bad results for the small instances compared to Gurobi, but their results become more and more successful as the problem size increases compared to the results of Gurobi. Both methods are able to provide good feasible solutions, which are at most around 50% away from the theoretical upper bound for the largest instance with 1500 points. Average percent deviations for the Gurobi, PIA, and GREEDY methods are 40.66%, 24.29%, and 37.47%, respectively.

A similar analysis can be conducted by looking at Table 7; Gurobi is very successful for the smallest instances but its results rapidly deteriorate as the problem size gets larger while the PIA heuristic is able to sustain successful results with increasing problem size. The Greedy heuristic is less successful for the competition model than the basic model. Average percent deviations for the Gurobi, PIA, and GREEDY methods are 20.57%, 15.01%, and 29.92%, respectively.

Finally, we present percent deviations of the feasible solutions from the theoretical upper bounds for test 2 respectively for the basic and competition models in Table 8 and Table 9.

Table 8. Percent deviations from the theoretical upper bounds for the basic model run for set 2.

	%GUROBI	%PIA	%GREEDY
1k	4.52	3.31	37.65
2k	10.31	6.56	42.19
5k	10.36	9.57	47.24
10k	40.74	20.05	51.54

Table 9. Percent deviations from the theoretical upper bounds for the competition model run for set 2.

	%GUROBI	%PIA	%GREEDY
1k	0	0	33.06
2k	0	0	27.06
5k	0	0	34.10
10k	0	0	32.89

As can be understood from Table 8 and Table 9, the most successful method is PIA for the basic model, and both PIA and Gurobi are able to find the optimal solutions for all instances for the competition model since the percent deviations from the theoretical upper bounds are all 0. On the other hand, average percent deviations of the greedy approach are respectively 45% and 32% for the basic and competition models.

5. Conclusions

In this study, two integer programs are generated to find an initial seed among the nodes of a network that maximizes the number of affected nodes at the end of the planning horizon. Both methods employ a linear threshold influence emission method. The threshold model requires the total of the effects coming from the previously affected neighbors to be at least as much as the predetermined threshold level of the node in order for a node to be affected in a period. In the first model, a single party that tries to maximize its influence in a network is considered, while in the other model there is an enemy party that tries to emit its own message. Two heuristics named PIA and the greedy approach are proposed as solution strategies. The greedy approach

constructs the initial seed from the nodes having the ratio of largest total affect over cost while PIA limits the number of periods and increases it one by one until the objective values of the restricted models converge. This work can be extended in several ways. First of all, a model in which nodes' threshold levels dynamically change depending on the number of positively affected neighbors can be formed. Finally, a more general model in which it is possible to change one's mind after being positively or negatively affected would also be interesting. In such a framework, threshold levels may have two levels, one for the neutral case and the other for the affected case.

References

[1] Hiasdasdnz O, Skiera B, Barrot C, Becker JU. Seeding strategies for viral marketing: an empirical comparison. J Mark 2011; 75: 55-71.

[2] Aral S, Muchnik L, Sundararajan A. Engineering social contagions: optimal network seeding in the presence of homophily. Netw Sci 2013; 1: 125-153.

[3] Chen W, Wang C, Wang Y. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 2010. pp. 1029-1038.

[4] Sangachin M, Samadi M, Cavuoto L. Modeling the spread of an obesity intervention through a social network. J Healthc Eng 2014; 5: 293-312.

[5] Kimura M, Saito K, Motoda H. Blocking links to minimize contamination spread in a social network. ACM T Knowl Discov D 2009; 3: 9.

[6] Macy MW. Chains of cooperation: threshold effects in collective action. Am Sociol Rev 1991; 56: 730-747.

[7] Deroian F. Formation of social networks and diffusion of innovations. Res Policy 2002; 31: 835-846.

[8] Borgatti S. Identifying sets of key players in a social network. Comput Math Organ Theory 2006; 12: 21-34.

[9] Wasserman S, Faust K. Social Network Analysis: Methods and Applications. Cambridge, UK: Cambridge University Press, 1994.

[10] Campbell A. Word-of-mouth communication and percolation in social networks. Am Econ Rev 2013; 103: 2466-2498.

[11] Domingos P, Richardson M. Mining the network value of customers. In: Proceedings of the seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 26–29 August 2001; San Francisco, CA, USA. New York, NY, USA: ACM. pp. 57-66.

[12] Kempe D, Kleinberg J, Tardos É. Maximizing the spread of influence through a social network. In: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 24–27 August 2003; Washington, DC, USA. New York, NY, USA: ACM. pp. 137-146.

[13] Leskovec J, Krause A, Guestrin C, Faloutsos C, Van Briesen J, Glance N. Cost-effective outbreak detection in networks. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 12–15 August 2007; San Jose, CA, USA. New York, NY, USA: ACM. pp. 420-429.

[14] Goyal A, Bonchi F, Lakshmanan LV, Venkatasubramanian S. On minimizing budget and time in influence propagation over social networks. Soc Netw Anal Min 2013; 3: 179-192.

[15] Ackerman E, Ben-Zwi O, Wolfovitz G. Combinatorial model and bounds for target set selection. Theor Comput Sci 2010; 411: 4017-4022.

[16] Wu HH, Küçükyavuz S. A two-stage stochastic programming approach for influence maximization in social networks. Comput Optim Appl 2018; 69: 563-595.

[17] Kermani MA, Aliahmadi A, Hanneman R. Optimizing the choice of influential nodes for diffusion on a social network. Int J Commun Syst, 2016; 29: 1235-1250.

[18] Güney E. On the optimal solution of budgeted influence maximization problem in social networks. https://doi.org/10.1007/s12351-017-0305-x, 2017.

[19] Günneç D, Raghavan S, Zhang R. Tailored Incentives and Least Cost Influence Maximization on Social Networks. Technical Report. College Park, MD, USA: University of Maryland, 2016.

[20] Günneç D, Raghavan S. Integrating social network effects in the share-of-choice problem. Decis Sci 2017; 48: 1098-1131.

[21] Fischetti M, Kahr M, Leitner M, Monaci M, Ruthmair M. Least cost influence propagation in (social) networks. Math Program 2018; 170: 293-325.

[22] Gillen CP, Veremyev A, Prokopyev OA, Pasiliao EL Critical arcs detection in influence networks. Netw Int J 2018; 71: 412-431.

[23] Sheldon D, Dilkina B, Elmachtoub AN, Finseth R, Sabharwal A, Conrad J, Gomes CP, Shmoys D, Allen W, Amundsen O et. al. Maximizing the spread of cascades using network design. arXiv preprint, arXiv:1203.3514, 2012.

[24] Yang L, Giua A, Li Z. Minimizing the influence propagation in social networks for linear threshold models. IFAC-PapersOnLine 2017; 50: 14465-14470.

[25] Szolnoki A, Perc M. Collective influence in evolutionary social dilemmas. EPL-Europhys Lett 2016; 113: 58004.

[26] Berger J, Sorensen AT, Rasmussen SJ. Positive effects of negative publicity: when negative reviews increase sales. Market Sci 2010; 29: 815-827.

[27] Samadi M, Nikolaev A, Nagi R. A subjective evidence model for influence maximization in social networks. Omega-Int J Manage S 2016; 59: 263-278.

[28] Samadi M, Nikolaev A, Nagi R. The temporal aspects of the evidence-based influence maximization on social networks. Optim Methods Softw 2017; 32: 290-311.

[29] Jalili M, Perc M. Information cascades in complex networks. J Complex Netw 2017; 5: 665-693.

[30] Dantzig GB. Origins of the simplex method. In: Nash SG, editor. A History of Scientific Computing. New York, NY, USA: ACM Press History Series, 1990. pp. 141-151.

[31] Karmarkar N. A new polynomial-time algorithm for linear programming. In: Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing; 30 April–2 May 1984; Washington, DC, USA. New York, NY, USA: ACM. pp. 302-311.

[32] Land AH, Doig AG. An automatic method of solving discrete programming problems. Econometrica 1960; 28: 497-520.

[33] Marchand H, Martin A, Weismantel R, Wolsey L. Cutting planes in integer and mixed integer programming. Discrete Appl Math 2002; 123: 397-446.

[34] Mitchell JE. Branch-and-cut algorithms for combinatorial optimization problems. In: Pardoslos PM, Resende MGC, editors. Handbook of Applied Optimization. New York, NY, USA: Oxford University Press Inc, 2002. pp. 65-77.

[35] Barnhart C, Johnson EL, Nemhauser GL, Savelsbergh MW, Vance PH. Branch-and-price: column generation for solving huge integer programs. Oper Res 1998; 46: 316-329.

[36] Wolsey LA. Integer Programming. Hoboken, NJ, USA: Wiley-Interscience, 1998.