An exact method for influence maximization based on deterministic linear threshold model

Eszter Julianna Csókás Tamás Vinkó

University of Szeged, Institute of Informatics, Hungary

June 10, 2020 — submitted to journal

Abstract

Influence maximization (IM) is a challenging combinatorial optimization problem on (social) networks given a diffusion model and limited choice for initial seed nodes. In a recent paper an integer programming formalization of IM using the so-called deterministic linear threshold diffusion model was proposed [7]. In fact, it is a special 0-1 linear program in which the objective is to maximize influence while minimizing the diffusion time. In this paper, by rigorous analysis, we show that the proposed algorithm can get stuck in locally optimal solution or cannot even start on certain input graphs. The identified problems are resolved by introducing further constraints which then leads to a correct algorithmic solution. Benchmarking results are shown to demonstrate the efficiency of the proposed method.

Keywords: Influence maximization; deterministic linear threshold; integer linear programming

1 Definitions

Influence maximization. Let $G = (V, E, W)$ be a directed weighted graph, where V is the set of nodes, E is the set of edges and $W : E \to \mathbf{R}_+$ is a non-negative weight function. Influence maximization (IM) is a combinatorial optimization problem in which, given a weighted directed graph G, a diffusion (or spreading) model, and an integer $k \geq 1$, it is required to identify the so-called seed nodes $v_1, \ldots, v_k \in V$ which can make the largest influence in the network [6].

In the following it is assumed that $n = |V|$. For a given node $j \in V$ the set of in-neighbors is denoted by $N(j)$. For an integer $0 < k \leq n$, the function $\sigma(S)$ determines that choosing a node set $S \subset V$ of size k as seed set, how many nodes will be influenced by executing the spreading model. Formally, the optimization problem can be described as

$$\max_{S \subset V, |S|=k} \sigma(S).$$

Kempe et al. [6] investigated the influence maximization problem using spreading models with stochastic parameters. Hence, $\sigma(S)$ stands for the expected value of the number of influenced nodes.

The influenced and uninfluenced nodes will also be called as active and inactive nodes, respectively.

Linear threshold model. Let $b_{i,j} \in (0, 1)$ be the edge weight between node i and j, θ_i be the threshold of node i, and set $\hat{N}(j)$ be the already influenced in-neighbors of node j.

The linear threshold (LT) model starts from $t = 1$ (where $t \in \mathbb{N}$), and iteratively does the following steps by increasing the value of t:

Step 1 Let $0 < k \leq n$ be fixed, S_0 a seed set containing k nodes, $t = 1$, $\mathscr{I}_1 = \emptyset$ and $\sigma(S_0) = k$.

Step 2 For all $i \in V$ inactive nodes, if

$$\sum_{j \in \hat{N}(i)} b_{i,j} \geq \theta_i$$

holds, then put node i into the set $\mathscr{I}_t$, in which at the end of this step all the nodes are labeled as influenced.

Step 3 Let $\sigma(S_t) := \sigma(S_{t-1}) + |\mathscr{I}_t|$, so calculate the influence value by adding the newly influenced nodes.

Step 4 If $\mathscr{I}_t = \emptyset$ holds, meaning that it is not possible to make more nodes influenced, then STOP. Otherwise, let $t := t+1$ and go back to Step 2.

The threshold value θ_i determines the influenceability of node i. According to the original paper of Kempe et al. [6], it is arguably difficult to measure (e.g. in social networks) the value of this thresholds. Hence, the evaluation of the LT model is done by executing it R times and then the average influence value is taken; this is how the expected value of σ is obtained. In this case it can be shown that the function $\sigma(\cdot)$ has *submodularity* property, which has the important consequence that a greedy algorithm guarantees that

$$\sigma(S) \geq (1 - 1/e) \cdot \sigma(S^*)$$

holds for any seed set S, where S^* is the optimal seed set [12].

Deterministic linear threshold model. In the deterministic LT (DLT) model all the θ_i threshold values are fixed. In the recent years, this model has been studied [5, 10]. In fact, the original LT model by Granovetter is also deterministic [3]. One of the most interesting fact about DLT that submodularity does not hold [1], thus the greedy algorithm cannot be expected to be efficient. Our paper focuses on the DLT model.

2 Related works

The most relevant works to our paper are the ones using ILP models. For influence *minimization* [16] gives an ILP model, which is another problem. Using the LT model [14] gives such ILP formalism in which the aim is to determine the set of nodes which will never get influenced. This information might be used for solving the original problem. The already mentioned paper [1] considers the problem as constraint satisfaction and investigates the efficiency of belief propagation algorithm.

In [4] a binary integer program that approximates the IM using independent cascade diffusion model (which is different than the one used in this paper) by Monte Carlo sampling is developed together with a linear programming relaxation based method with a provable worst case bound.

The paper [8] focuses on competitive IM based on probabilistic independent cascade model in which the seed individuals of one entity is already known, while another entity wants to choose its seed set of individuals that triggers an influence cascade of maximum impact. An algorithmic framework based on a Benders decomposition is developed which enables to handle graphs with thousands of nodes and edges. Note that the full ILP model in [7] also considers competition explicitly.

Finally, [11] investigates a robust optimization problem using the DLT model. It is assumed that the nodes' thresholds and the edge weights can change within a certain domain. The problem of our paper is a special version of this general one. They construct such an ILP model in which the time parameter t does not play a role (in contrast to our work), moreover, the number of variables grow exponentially.

3 A 0-1 linear programming model

An integer linear programming model of influence maximization based on the DLT model was recently proposed and studied in [7]. More precisely, it is a special 0-1 LP in which $\mathbf{x} \in \{0,1\}^{n\times\mathscr{T}}$ is the decision variable, $n = |V|$, and the index $\mathscr{T}$ is also part of the optimization problem. Hence, $\mathbf{x}$ is a binary matrix in which choosing the rows in the first column to be equal to 1 represents the selection of the seed nodes. This should be done in such a way that, given certain constraints dictated by DLT model, the sum of the last column is to be maximized.

Assuming that $\mathscr{T} > 0$ is a given integer constant, let $T = \{2,\dots,\mathscr{T}\}$ be the set of time periods describing the diffusion process. Let integer $k > 0$ be the number of seed nodes at $t = 1$. The set of in-neighbors of node i is denoted by $N(i)$. In the following, a slightly simplified model from [7] is given, in which the cost of selecting a seed node is equal to 1.

$$\max \sum_{i=1}^{n} x_{i,\mathscr{T}} \tag{1}$$

$$\sum_{i=1}^{n} x_{i,1} \le k \tag{2}$$

$$\sum_{j:i\in N(j)} x_{j,t-1} b_{j,i} \ge \theta_i x_{i,t} \quad \forall (i \in I, t \in T) \tag{3}$$

$$\sum_{j:i\in N(j)} x_{j,t-1} b_{j,i} \le \theta_i + x_{i,t} \quad \forall (i \in I, t \in T) \tag{4}$$

$$x_{i,t-1} \le x_{i,t} \quad \forall (i \in I, t \in T) \tag{5}$$

The objective function in fact has the form

$$\min_{\mathscr{T}} \max \sum_{i=1}^{n} x_{i,\mathscr{T}}$$

and together with constraints (2) - (5) we have a bilevel optimization problem. The AMPL modeling language [2], which we used for implementation and numerical experiments (see Section 7), is not suitable for directly describing bilevel optimization models. That would require to have declarations as `var T; var x{n,T};` which is not supported. Hence, we need to consider and treat $\mathscr{T}$ as constant.

Remark. *The globally optimal solution for the bilevel problem is when we have the maximal influence within the shortest diffusion time. This will be referred as ($\sigma^*, \mathscr{T}^*$) in the followings.*

4 An iterative algorithm

The solution method for the bilevel optimization problem proposed in [7] is shown in Algorithm 1.

This iterative methods makes it possible to find the minimal $\mathscr{T}$ since it stops when further spreading of influence is not possible. Thus, the value of $\mathscr{T}^*$ is given by the loop variable $\mathscr{T}$.

5 Analysis

In this section a thorough analysis of Algorithm 1 proposed in [7] and shown in Section 4 is made.

For a start, it turns out that the optimization problem (1)-(5) needs to be extended by an additional constraint.

Algorithm 1

```
Step 1 Start the iteration from $\mathscr{T} := 1$.
Step 2 Solve the optimization problem (1) - (5) with fixed $\mathscr{T}$.
Step 3 If $\mathbf{x}_{i,\mathscr{T}} = \mathbf{x}_{i,\mathscr{T}-1} \quad \forall(i \in I)$, i.e., the last two columns of $\mathbf{x}$ are the same then STOP, the optimum is
    found. Otherwise, let $\mathscr{T} := \mathscr{T}+1$ and go back to Step 2.
```

Proposition 1. *The input graph needs to be extended by weighted loop edges, i.e.,*

$$b_{i,i} = \theta_i \quad \forall(i \in I), \tag{6}$$

Proof. The seed nodes are selected at time $\mathscr{T} = 1$. At this point there are no further influenced nodes in the graphs, thus constraints (3)-(4) are fulfilled only if the selected seed nodes form a connected subgraph. This cannot be held in general, thus it is not possible to select seed nodes, and the influence spreading cannot be started.

On the other hand, if we have loop edges with weights equal to the node's threshold value, then all the selected seed nodes fulfill constraints (3)-(4). These loop edges do not play any further role in the spreading model. □

As an example, see the graph on Fig. 1. Without the loop edges, the influence spreading cannot be started. The matrix $\mathbf{x}$ corresponding to the correct global optimum for this graph is

$$\mathbf{x} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}.$$

As it can be seen, the nodes represented by row 2 and row 4 are selected as seed nodes, which both have threshold value 0.01. These two nodes are not connected to each other, hence the model (1)-(5) is infeasible at time $\mathscr{T} = 1$.

Remark. *On the illustrative example graphs shown later, the loop edges will not be indicated for simplicity.*

Proposition 2. *From the ILP model* (1) - (6)*, neither* (4) *nor* (5) *can be left out.*

Proof. First, we show that constraint (5) is needed. Constraint (5) prescribes that those nodes becoming influenced they remained to be so. According to (1), the number of these nodes are to be maximized. Hence, by increasing the loop variable $\mathscr{T}$, the number of influenced nodes will not be decreasing by excluding constraint (5). In this case, however, influenced nodes could become uninfluenced corresponding to a solution at a given time t.

For an illustrative example, see Fig. 2. The solution reported by Algorithm 1 from Section 4 is

$$\mathbf{x} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

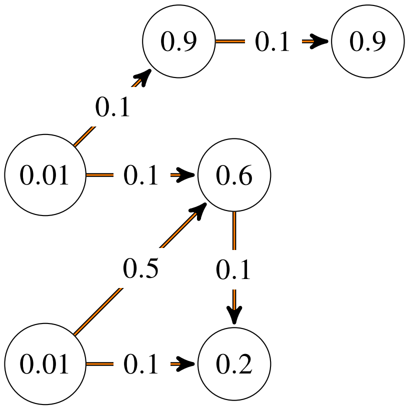

Figure 1: Example graph to show the need of the loop edges, see constraint (6)

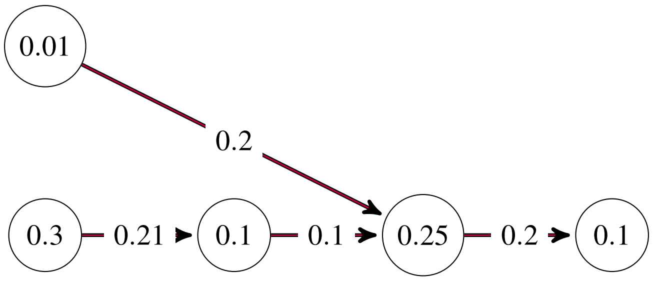

Figure 2: Example graph for Proposition 2

where the diffusion time is $\mathscr{T}^* = 4$. We can see that the node represented by row 3 got influenced at $\mathscr{T} = 2$, then it was inactive at $\mathscr{T} = 3$ and finally it got influenced again at $\mathscr{T} = 4$. Thus, without constraint (5) we do not obtain correct solution.

Now, we discuss the role of constraint (4). Constraints (3) and (4) can get satisfied only together, which is based on the fact that $\theta_i x_{i,t} < \theta_i + x_{i,t}$, because

- if $x_{i,t}$=0, then the left hand side is 0, while the right hand side is θ_i; or
- if $x_{i,t}$=1, then the right hand side is θ_i, while the left hand side is θ_i+1,

where $\theta_i \in [0,1]$. □

Proposition 3. *For the optimization problem* (1) - (6) *there is a graph for which*

$$(\sigma, \mathscr{T}) = (\sigma, \mathscr{T}+1) \quad \textit{and} \quad (\sigma, \mathscr{T}+1) < (\sigma, \mathscr{T}+2).$$

Proof. Such a graph is shown on Fig. 3. By choosing certain seed nodes, the algorithm cannot increase the number of influenced nodes from $\mathscr{T} = 4$ to $\mathscr{T} = 5$. On the other hand, by changing the seed nodes at $\mathscr{T} = 6$ it can increase the number of active nodes. □

Remark. *Algorithm 1 gets stuck in a suboptimal solution on the input graph shown on Fig. 3 even without constraint* (4).

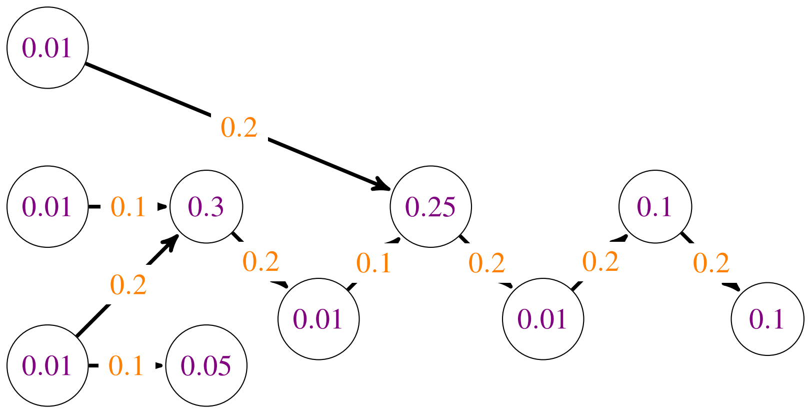

Figure 3: Example graph for Proposition 3

We conclude that an extension of the optimization model (1) - (6) is needed in order to have a strategy about when to stop the iterative algorithm to be sure that it indeed reached the globally optimal solution. At that end, the following constraint is added:

$$\sum_{i=1}^{n} x_{i,\mathscr{T}-1} + 1 \leq \sum_{i=1}^{n} x_{i,\mathscr{T}}. \tag{7}$$

The purpose of constraint (7) is to force that for a given $\mathscr{T}$, the last step of the diffusion must have at least one more influenced node than in the previous step. We can thus guarantee no repetition in the last two column of matrix $\mathbf{x}$.

Proposition 4. *For increasing $\mathscr{T}$ values the solutions of* (1) - (7) *do not necessary form a monotonically increasing sequence. Moreover, it can also happen that repetition occurs for consecutive columns in matrix* $\mathbf{x}$.

Proof. As an example we refer to the graph shown in Fig. 4. The global optimum needs $\mathscr{T} = 4$ diffusion step. By allowing further iteration steps to be taken by the algorithm, we expect to obtain an infeasible

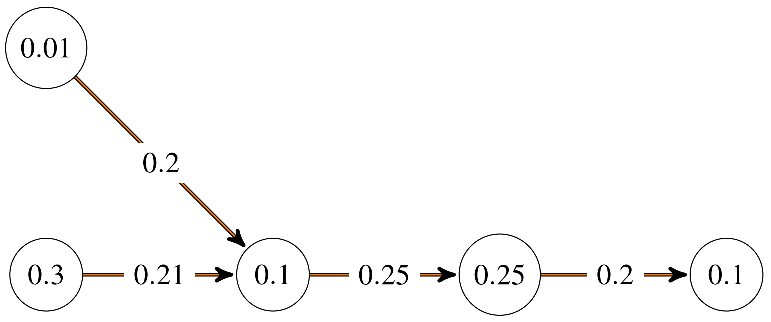

Figure 4: Example graph for Proposition 4

solution. However, the solution matrix for $\mathscr{T} = 5$ is

$$\mathbf{x} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

which contains repetition in its second and third column, hence lengthening the spreading up to $\mathscr{T} = 5$. This column repetition could go on forever. Note that this solution matrix is not representing the global optimum simply because the minimum time is $\mathscr{T}$=4. This phenomenon is caused by constraints (3)-(4), which is explained in the graph shown on Fig. 4. According to matrix **x** reaching the node with the threshold value 0.25 could be delayed, thus we examine that case. Before reaching that node, constraints (3) always gets trivially satisfied, given the fact that its right hand side equals to 0. When the neighbor of the node in question is already activated, then on the left hand side of the constraint (4) the weight of the incoming edge appears, while we have either 0 or the threshold of the node on its right hand side. Since both values satisfy constraint (4), the algorithm allows to have the activation of the node after getting the global optimum. □

Proposition 5. *Column repetition in the solution of the problem* (1) - (7) *can only occur after the globally optimal solution has been found.*

Proof. Recall that by global optimum we mean the maximal σ and minimal $\mathscr{T}$ values, together. The objective function (1) is maximization, thus the algorithm cannot allow to have column repetition before reaching the global optimum, because it would increase the value of $\mathscr{T}$ with keeping the σ value. □

Proposition 6. *The globally optimal solution of* (1) - (6) *satisfies constraint* (7) *as well.*

Proof. Note that constraint (7) takes into account only the last two columns of matrix **x**. There are two cases:

- The number of active nodes in the last column is greater than the number of them in the previous (penultimate) column. Since constraint (7) was introduced for enforcing exactly this situation, this case makes this new constraint satisfied.
- The last two columns are the same. Since the global optimum is considered, the column corresponding to the values $(\sigma^*, \mathscr{T}^*)$ is repeated. This means that we have already passed the global optimum since the $\mathscr{T}$ value of the last column is greater than $\mathscr{T}^*$.

□

Proposition 7. *The diffusion value $\mathscr{T}^*$ and influence value σ^* corresponding to the globally optimal solution of* (1) - (6) *are respectively the same as the values $\mathscr{T}^{**}$ and σ^{**} corresponding to the global optimum of* (1) - (7).

Proof. By introducing the constraint (7) such an optimization problem is obtained in which the value of $\mathscr{T}$ cannot be increased forever: either it gets an infeasible solution or it starts the repetition of certain column or columns.

Firstly, let us see if $\sigma^{**} = \sigma^*$ holds. We have two check two cases.

- Assume that $\sigma^* < \sigma^{**}$. By dropping the constraint (7), we have a better solution for the problem (1) - (6), which is not possible, since $(\sigma^*, \mathscr{T}^*)$ is the globally optimal solution.
- Assume that $\sigma^* > \sigma^{**}$. By Proposition 6 we know that a solution of (1) - (6) also satisfies constraint (7) as well. Thus σ^* would be a better solution for the problem (1) - (7), which is not possible as σ^{**} is maximal.

We have contradictions for both cases, thus $\sigma^* = \sigma^{**}$.

Secondly, we check whether $\mathscr{T}^{**} = \mathscr{T}^*$ holds. We have to check again two cases.

- Assume that $\mathscr{T}^* < \mathscr{T}^{**}$. By Proposition 6 this is not possible as constraint (7) would not be satisfied.
- Assume $\mathscr{T}^* > \mathscr{T}^{**}$. We know that σ^* is global optimum for the problem (1) - (7) as well. This solution cannot be found with smaller amount of iteration steps under the constraints (7).

We have contradictions again, thus $\mathscr{T}^* = \mathscr{T}^{**}$. □

Proposition 8. *If the problem* (1) - (7) *becomes infeasible for a given* $\mathscr{T}$ *value, then it remains to be infeasible for the further iteration steps as well.*

Proof. We prove by induction that if a solution is feasible then it was so in earlier iteration steps.

- $\mathscr{T} = 2$: The algorithm was able to do the first iteration, thus it could select the seed nodes. Hence, it has a feasible solution at $\mathscr{T}$ = 1.
- $\mathscr{T} = 3$: By constraint (7) the last two columns cannot be the same. The first two columns of matrix **x** are certainly feasible, the corresponding nodes can be reach within this time frame.
- $\mathscr{T} = m$: In case we remove the mth column, a feasible solution is obtained since the nodes in the $(m-1)$th column could be reached and activated in $\mathscr{T} - 1$ steps. It is important to see that this solution is not necessarily globally optimal for all $t \in \mathscr{T}$.

□

Finally, implicated by Proposition 7 and 8 we have the following

Corollary 1. *The problem* (1) - (7) *is feasible in the iteration steps* $2, \dots, \mathscr{T}^*$*, i.e., before finding the global optimum.*

6 A correct iterative algorithm

Based on the analysis in Section 5 the globally optimal solution of the influence maximization problem under deterministic linear threshold diffusion model can be found by Algorithm 2.

Algorithm 2

```
Step 1 Start the iteration with $\mathscr{T} := 1$.
Step 2 Solve the problem (1) - (7) on the extended graph with loop edges for the diffusion time value $\mathscr{T}$.
Step 3 If the solution becomes infeasible or two consecutive columns are the same inside the matrix $\mathbf{x}$, then
    STOP, the global optimum is found. Otherwise, let $\mathscr{T} = \mathscr{T} + 1$ and go back to Step 2.
```

7 Numerical experiments

7.1 Computational environment

The implementation of all the investigated ILP models were done in AMPL [2]. For the numerical experiments the solver Gurobi 9.0.0 was used with the non-default options: `threads=1 lpmethod=0 cuts=0 mipgapabs=1e-2`, which, compared to the default options, turned out to be much more efficient for these particular models. The computer used had Intel Xeon CPU E5-2660 at 2.00GHz with 64G memory running Ubuntu Linux 18.04.4.

7.2 Test graphs

For benchmarking the proposed algorithm some random graphs were generated. Two types of random graphs were used: Watts-Strogatz (WS) small-world graphs [15] and so-called LFR graphs with prescribed community structures [9].

WS graphs. These graphs were generated by using the package `R/igraph`. The parameters were:

- number of nodes is 60,
- number of neighbors in the starting graphs are $s = 4, 8$ and 12,
- and the rewiring probabilities are $\beta = 0.1$ and 0.3.

The `mutual` parameter was used which makes the graphs directed by doubling the undirected edges. Then 45% of randomly selected edges got removed. The edge weights were assigned as follows.

- First, for each edge a uniform at random number were generated in the interval $[0, 1]$.
- Nodes with larger than 1 in-weights were normalized to 1.
- Moreover, we applied a multiplication with a factor r_w which was a uniform at random number in the interval $[0.6, 1]$.

The threshold values of the nodes were generated uniform at random in the interval $[0.15, 0.4]$.

Using this particular procedure we were able to find such WS graphs on which the greedy algorithm found suboptimal solutions.

LFR graphs. These graphs were generated by the code from [9], obtaining weighted directed graphs with community structure (thus, resembling social networks). The weights were assigned to the edge using the followings.

- Nodes with in-weights larger than 1 (generated by the LFR method) were normalized to 1.
- Moreover, we applied a multiplication with a factor r_w which was a uniform at random number in the interval $[0.6, 1]$.

The threshold values of the nodes were generated uniform at random in the interval $[0.05, 0.4]$. Two configurations were made:

- number of nodes is fixed to $n = 120$,
- average degree $k = 6, 7$,
- maximum degree $maxk = 13, 10$,
- mixing parameter $\mu_w = 0.1$,
- minimal community size $minc = 7, 5$,
- maximal community size $maxc = 21, 42$.

For both types $5 - 5$ graphs were generated.

7.3 Benchmarking results

The testing of Algorithm 1 and 2 is shown by not only comparing the execution times of these two versions but also the results obtained by two other methods. Namely, the greedy algorithm [6] was implemented, in particular to investigate the lack of submodulatity. The other simple method was a random choice of seed nodes set in 20 times and then the best solution was reported. In all experiments the number of seed nodes were fixed to $k = 2$.

WS graphs. The results obtained for the Watts-Strogatz test graphs are reported in Table 1 and 2.

The random algorithm were able to find the optimal σ^* value in 6.6% of the cases. However, it always missed the minimal diffusion time $\mathscr{T}^*$. The greedy algorithm found the globally optimal (σ^*, $\mathscr{T}^*$) pairs in 40% of the cases . Note the effect of the fact that the submodularity (see Section 1) does not hold for the greedy algorithm due to the DLT diffusion model. Algorithm 1 from [7] missed the globally optimal solution in 2 cases (meaning 93% success rate).

Regarding the running time, see Table 2, obviously the random and the greedy algorithm were really fast. Comparing Algorithm 1 and 2 it can be seen that most of the cases they had roughly equal running times, however, our Algorithm 2 can be up to 21 times slower. Closer inspection into the results reveal that, for example, for the case $s = 4, \beta = 0.1, i = 3$ our proposed algorithm needed 1,496 seconds to prove that there is no better solution than $(59, 15)$. For $t > 15$ values the σ value got decreasing. Note that there are cases where the optimal seed set can make the entire graph influenced, i.e., where $\sigma^* = 60$, yet, our proposed algorithm is much slower. For example, in the case $s = 8, \beta = 0.1, i = 2$ it turns out that our algorithm was struggling in the last two iterations - this is certainly caused by the constraint (7). On the other hand, for the difficult problem $s = 8, \beta = 0.1, i = 3$, Algorithm 2 was about 1.5 times faster, which was gained in the last two iterations.

Table 1: Benchmarking results for the small-world Watts-Strogatz graphs; optimum values

			random		greedy		Algorithm 1		Algorithm 2	
s	β	$i.$	σ	$\mathscr{T}$	σ	$\mathscr{T}$	σ	$\mathscr{T}$	σ	$\mathscr{T}$
4	0.1	1	58	15	60	14	59	10	60	14
		2	60	14	60	9	60	9	60	9
		3	2	1	59	15	59	15	59	15
		4	59	15	60	11	60	11	60	11
		5	4	2	60	10	60	10	60	10
4	0.3	1	60	13	60	7	60	7	60	7
		2	3	2	59	12	58	9	59	12
		3	3	2	58	8	58	8	58	8
		4	6	3	58	9	58	9	58	9
		5	58	11	58	9	59	11	59	11
8	0.1	1	2	1	58	10	60	10	60	10
		2	2	1	11	4	60	11	60	11
		3	3	2	60	10	60	10	60	10
		4	3	2	60	9	60	9	60	9
		5	3	2	33	6	60	10	60	10
8	0.3	1	5	4	48	7	60	8	60	8
		2	2	1	60	7	60	7	60	7
		3	2	1	12	5	60	11	60	11
		4	2	1	7	3	60	12	60	12
		5	2	1	6	2	60	9	60	9
12	0.1	1	2	1	4	2	60	9	60	9
		2	2	1	3	2	60	10	60	10
		3	3	2	4	2	60	11	60	11
		4	2	1	2	1	60	12	60	12
		5	2	1	5	2	7	3	7	3
12	0.3	1	3	2	4	2	7	4	7	4
		2	2	1	4	2	8	5	8	5
		3	2	1	4	2	60	10	60	10
		4	2	1	8	3	60	9	60	9
		5	2	1	8	4	60	10	60	10

Table 2: Benchmarking results for the small-world Watts-Strogatz graphs; running times in seconds.

s	β	$i.$	random	greedy	Algorithm 1	Algorithm 2
4	0.1	1	0.28	6.28	11.58	15.53
		2	0.23	4.84	18.91	21.63
		3	0.01	91.83	113.5	1665
		4	0.22	13.45	18.35	17.79
		5	0.01	6.86	18.13	24.77
4	0.3	1	0.19	8.93	7.636	8.562
		2	0.01	12.02	17.75	555.1
		3	0.02	7.40	7.411	60.31
		4	0.02	9.44	36.88	759.1
		5	0.18	8.33	15.56	78.28
8	0.1	1	0.01	7.40	259.5	507.8
		2	0.01	0.50	441.8	1794
		3	0.02	9.93	2117	1365
		4	0.02	6.73	114.1	156.6
		5	0.02	1.51	1646	1131
8	0.3	1	0.04	2.29	96.02	82.32
		2	0.01	2.22	30.68	32.29
		3	0.01	0.83	1383	2644
		4	0.01	0.26	4596	5590
		5	0.01	0.11	285.4	460.1
12	0.1	1	0.01	0.12	351.1	626.6
		2	0.01	0.11	2088	5352
		3	0.02	0.11	2769	10124
		4	0.01	0.00	35556	44955
		5	0.01	0.11	33.18	21.45
12	0.3	1	0.02	0.11	65.41	48.91
		2	0.01	0.11	131.1	40.41
		3	0.01	0.11	4851	5304
		4	0.01	0.27	447.1	1208
		5	0.01	0.52	2794	7044

Table 3: Benchmarking results for the LFR graphs; optimal values

						random		greedy		Algorithm 1		Algorithm 2	
k	*maxk*	μ_w	*minc*	*maxc*	*i.*	σ	$\mathscr{T}$	σ	$\mathscr{T}$	σ	$\mathscr{T}$	σ	$\mathscr{T}$
6	10	0.1	5	42	1	110	13	120	13	110	9	120	13
					2	103	12	103	10	103	10	103	10
					3	120	12	120	9	120	8	120	8
					4	19	6	91	12	91	12	91	12
					5	49	15	101	12	101	11	101	11
7	13	0.1	7	21	1	75	10	94	11	96	10	96	10
					2	62	13	120	13	120	13	120	13
					3	23	6	120	15	120	15	120	15
					4	2	1	83	8	83	8	83	8
					5	51	9	111	10	111	10	120	18

Table 4: Benchmarking results for the LFR graphs; runnig times in seconds

k	*maxk*	μ_w	*minc*	*maxc*	*i.*	random	greedy	Algorithm 1	Algorithm 2
6	10	0.1	5	42	1	0.62	107.89	371.2	419.1
					2	0.49	121.65	981.6	4,996
					3	0.42	36.99	203.5	187.8
					4	0.19	30.97	169.1	389.5
					5	0.68	128.83	3,289	25,508
7	13	0.1	7	21	1	0.36	24.2558	206.6	1,979
					2	0.59	59.03	435.7	332.7
					3	0.21	71.37	519.7	534.9
					4	0.02	49.51	140.1	1,168
					5	0.33	79.829615	223.4	396.1

LFR graphs. The results obtained for the LFR graphs are shown in Table 3 and 4.

The random and greedy algorithms were able to find the optimal σ^* value in 20% and 80% of the cases, respectively. However, the random selection of seed nodes always missed the corresponding minimal diffusion time $\mathscr{T}^*$. The greedy algorithm found the globally optimal (σ^*, $\mathscr{T}^*$) pairs in 60% of the cases. Note that greedy reported larger diffusion time than the optimal in two cases. We can see that Algorithm 1 missed the globally optimal solution for two graphs.

Regarding the running times, see Table 4, random and greedy were again really fast. Comparing Algorithm 1 and 2 we can see that our proposed model was slightly faster in two cases. On the other hand it can be up to 8 times slower. This is due to the same fact mentioned for the WS graphs: it takes considerable time to prove the optimality of the solution.

7.4 Possibility for improvement?

The exact ILP model can provide us with rather pessimistic running times even for the relatively small graphs used in our benchmarking experiments. It is tempting to suggest that the exact model used in our Algorithm 2 could be combined with either the random selection of the seed nodes or with the greedy approach as finding initial values and hence potentially reduce the overall running time. It is easy to see that this could lead to suboptimal results and the argument is as follows. Recall that the IM problem considered in this paper aims at finding both the optimal value of the maximum influence together with the minimum diffusion time. According to our experiments both random and greedy approach can sometimes overshoot for the diffusion time, i.e., can find a solution for which the diffusion time is larger than the optimal.

8 Conclusions

We proposed an exact 0-1 linear programming model for the influence maximization problem based on deterministic linear threshold model. By rigorous analysis the correctness was shown. The work was inspired by a recent paper [7]. In fact, our proposed model is an improved version in a sense that the model in [7] does not always find the global optimum. We demonstrated this fact in our analysis and by numerical testings.

According to our benchmarking results, even for relatively small graphs, finding the exact solution can only be done in very pessimistic running times. In one hand this is not surprising as the problem is NP-hard. On the other hand, our exact model is the first one to computationally demonstrate how difficult is to find the global solution.

Acknowledgements

The project has been supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.3-VEKOP-16-2017-00002) and by grant TUDFO/47138-1/2019-ITM of the Ministry for Innovation and Technology, Hungary.

References

[1] Altarelli F, Braunstein A, Dall'Asta L, Zecchina, R (2013) Optimizing spread dynamics on graphs by message passing. Journal of Statistical Mechanics: Theory and Experiment, P09011.

[2] Fourer R, Gay David M, Kernighan Brian W (1993) AMPL. A modeling language for mathematical programming. Thomson

[3] Granovetter M (1978) Threshold models of collective behavior. American Journal of Sociology, 83(6), 1420-1443

[4] Güney E (2019) An efficient linear programming based method for the influence maximization problem in social networks, Information Sciences, 503:589–605

[5] Karampourniotis PD, Szymanski BK, Korniss G (2019) Influence Maximization for Fixed Heterogeneous Thresholds Scientific Reports, 9, 5573

[6] Kempe D, Kleinberg J, Tardos É (2003) Maximizing the spread of influence through a social network. In: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, 137–146

[7] Keskin ME, Güler MG (2018) Influence maximization in social networks: an integer programming approach. Turkish Journal of Electrical Engineering & Computer Sciences, 26(6): 3383–3396

[8] Kahr M, Leitner M, Ruthmair M, Sinnl M (2020) Benders decomposition for competitive influence maximization in (social) networks. Omega, accepted for publication

[9] Lancichinetti A, Fortunato S, Radicchi F (2008) Benchmark graphs for testing community detection algorithms. Phys. Rev. E 78, 046110

[10] Lu Z, Zhang W, Wu W, Kim J, Fu B (2012) The complexity of influence maximization problem in the deterministic linear threshold model. Journal of combinatorial optimization, 24(3): 374–378

[11] Nannicini G, Sartor G, Traversi E, Calvo RW (2019) An exact algorithm for robust influence maximization, Optimization Online

[12] Nemhauser G, Wolsey L, Fisher M (1978) An analysis of the approximations for maximizing submodular set functions. Mathematical Programming, 14:265–29

[13] K Rahimkhani, Aleahmad A, Rahgozar M, Moeini A (2015) A fast algorithm for finding most influential people based on the linear threshold model. Expert Syst. Appl. 42(3):1353–1361

[14] Rosa D, Giua A (2013) On the spread of innovation in social networks. IFAC Proceedings Volumes, 46(27):322–327

[15] Watts DJ, Strogatz SH (1998) Collective dynamics of ’small-world’-networks. Nature, 393(6684):440–442

[16] Yang L, Giua A, Li Z (2017) Minimizing the influence propagation in social networks for linear threshold models. IFACPapersOnLine, 50:14465-14470.