LII Simpósio Brasileiro de Pesquisa Operacional
João Pessoa - PB, 3 a 5 de novembro de 2020

Tighter Dual Bounds on the Least Cost Influence Problem

Renato Silva de Melo
Federal University of Paraná
Informatics Department, Polithecnique Center
rsmelo@inf.ufpr.br

Andre Luís Vignatti
Federal University of Paraná
Informatics Department, Polithecnique Center
vignatti@inf.ufpr.br

Flávio Keidi Miyazawa
State University of Campinas
Institute of Computing
fkm@ic.unicamp.br

Matheus Jun Ota
State University of Campinas
Institute of Computing
matheus.ota@students.ic.unicamp.br

ABSTRACT

The Least Cost Influence Problem is a combinatorial problem that is usually described in the context of social networks. The objective is to give incentives to a set of individuals in the network, such that some information is spread at minimum cost. We provide an efficient algorithm to get lower bounds in a branch-and-bound scheme, and use these in a Branch-and-Cut method. Computational results show the benefit of using our proposed bounds.

KEYWORDS. Integer Programming. Social Networks. Diffusion of information.

PAPER TOPICS. Combinatorial Optimization. Mathematical Programming.

1. Introduction

In the context of diffusion of information in social networks, early studies on sociology [Rogers, 2010] observes that an information spread through a social system like a contagious process, starting with a small group and spreading to other individuals in the system through the interrelation between them. One relevant problem that emerges from this background consists in identify a good set of individuals to target, the *early adopters*, and hope that the chosen individuals are able to persuade their friends into adopt a new behavior, who in turn also influence friends of friends by generating a chain of adoption. In applications like viral marketing, for example, the early adopters can be influenced by receiving products for free or by getting discounts for buying a new product.

Kempe et al. [2003] were among the first ones to study this problem from the discrete optimization perspective. They considered a maximization version, called INFLUENCE MAXIMIZATION PROBLEM, where the goal is to find a set of fixed size, such that the expected influence of such set is the largest possible. They show that this problem is NP-hard in stochastic diffusion models and propose an $(1 - \frac{1}{e})$-approximate greedy algorithm. To get this approximation guarantee they have shown that the influence function in randomized settings is submodular and monotone. Such approximation ratio was originally proved by Nemhauser et al. [1978] and holds for every maximization function which is submodular and monotone. Furthermore, computing the exact expected influence of a given set of individuals is a #P-hard problem [Chen et al., 2010a,b].

Chen [2009] studied the minimization related version which seeks for a target set of minimum size ensuring that a fixed fraction of the network will be activated. In the literature, this problem is referred to as the TARGET SET SELECTION (TSS) problem. Besides being NP-hard to solve, this problem is also hard to approximate. Chen [2009] proved that TSS cannot be approximated within a poly-logarithmic ratio. Even if explicitly deterministic thresholds are given, the problem is NP-hard to approximate within a ratio of $n^{1-\epsilon}$, for every $\epsilon > 0$ [Kempe et al., 2003; Chen, 2009]. This problem becomes tractable in some restricted class of graphs, for example, if the underlying graph is a tree [Chen, 2009], a block-cactus graph [Chiang et al., 2013], or a bounded treewidth graph [Ben-Zwi et al., 2011], then the problem can be solved in linear time.

In this work we investigate an extension of the TSS problem called LEAST COST INFLUENCE PROBLEM (LCIP). Rather than searching for a set where the propagation starts, this problem consists in offering incentives to the individuals in such a way that it will trigger a cascade that spreads to a given fraction of the network. Demaine et al. [2014] observe that the LCIP is a generalization of the TSS problem, so from a theoretical point of view, LCIP has at least the same complexity as the TSS problem. Günneç et al. [2016] focused on mathematical programming models for LCIP and describes algorithms for the problem on trees. Fischetti et al. [2018] introduce a generalization of the LCIP and introduce the concept of *activation functions* which is an extension of the commonly used *threshold functions*. Also, they propose a mathematical heuristic using column generation.

1.1. Contributions

While previous works provided relevant exact solutions [Ackerman et al., 2010; Günneç et al., 2016; Fischetti et al., 2018] and heuristic algorithms that can be used as upper bounds for this problem [Kempe et al., 2003; Chen et al., 2009; Cordasco et al., 2015; Demaine et al., 2014], nothing beyond the standard LP-relaxation was proposed to compute lower bounds on the LCIP.

We derive a problem-dependent relaxation algorithm based on the observation that the influence propagation network is a directed acyclic graph (DAG). The proposed algorithm exploits connectivity properties of graphs to obtain a lower bound for the problem. Furthermore, we prove

that the algorithm is correct, and show experimentally that our lower bounds are tighter than the linear programming relaxation, providing smaller optimality gaps. To the best of our knowledge, there are no works on combinatorial lower bounds for this problem. The main objective of our relaxation is for fathoming in a branch-and-bound algorithm and helping reduce the computational effort to obtain exact solutions.

The rest of this paper is organized as follows. Section 2 contains a brief overview about the diffusion process in social networks and the problem definition. Section 3 is devoted to describe the mathematical programming formulation of the problem and the exact method to solve it. In Section 4, we propose an algorithm to find lower bounds on the problem. Computational experiments are presented in Section 5. We conclude the paper in Section 6.

2. Problem definition

Let G be a directed graph that models a social network, where the vertex set $V(G)$ represents individuals and the arc set $E(G)$ corresponds to the relationships between these individuals. When the context is explicit, we denote the vertex set and arc set by V and E, respectively. Each arc $(i, j) \in E$ has an associated weight $d_{ij} > 0$ that models the strength of the influence of i over j.

To model the diffusion of influence between the individuals in a social network, we consider a well-known model called *threshold model*, presented by Granovetter [1978]. In this model, we say that a vertex i is *active* when it was persuaded to adopt a new behavior and *inactive*, otherwise. Every $i \in V$ has a threshold $t_i > 0$ which indicates the amount of influence needed to activate i, coming from i's neighbors. The activation process is *progressive*, i.e., each vertex can be active or inactive and a vertex can change from inactive to active, but not the other way around. Initially, a subset $A_0 \subseteq V$ is chosen to be active. Then, the vertices in A_0 send influence to their inactive neighbors. These neighbors might become active in the next iteration and give rise to a new set $A_1 \subseteq V$ of active vertices. This process is repeated until no vertex can be activated. Let $\{A_\tau\}_{\tau=0}^{k}$ be the sequence of obtained active vertices during the diffusion process, where A_k is the first set in the sequence that cannot activate any other vertex in the graph. We say that any vertex $i \in A_k \setminus A_0$ has been *influenced* by A_0.

Also, we consider the offer of external influences. These influences, which we call incentives, aim to break the resistance of an individual in becoming influenced in the activation process. The incentives are represented by a vector $\mathbf{y} \in \mathbb{Z}^{|V|}$, where each coordinate $y_i \in \mathbb{N}_0$ denotes the amount of incentive given to a vertex $i \in V$. Applying the incentive y_i on a vertex i decrease its threshold t_i and make it more susceptible to be activated. This incentive is added with the influence coming from the other vertices. More formally, the initial set of active vertices is given by $A_0 = \{i \in V : y_i \geq t_i\}$. The vertices in A_0 begin the process as active and all the others as inactive. Time progresses in discrete steps $\tau = 0, 1, \ldots, k$, an inactive vertex i becomes active at time τ if the total influence of its active in-neighbors plus its incentive exceeds the threshold t_i, i.e., if

$$\sum_{j \in N_i \cap A_{\tau-1}} d_{ji} \geq t_i - y_i,$$

where N_i denotes the set of in-neighbors of i.

The problem consists in offering incentives to the vertices in a way that it will trigger a cascade of influence that spreads to a given fraction α of the network. The goal is to minimize the total of incentives given to the individuals of the social network. The definition is given below.

Problem 1 (LEAST COST INFLUENCE PROBLEM (LCIP))**.** *We are given a real number $\alpha \in [0, 1]$, a directed graph G with weights d_{ij} on each arc $(i, j) \in E$, and a threshold t_i, for each vertex*

$i \in V$. The goal is to find a vector $\mathbf{y}$ of incentives which minimizes the sum $\sum_{i \in V} y_i$, ensuring that at least $\lceil \alpha |V| \rceil$ vertices are activated by the end of the activation process.

In the remainder of this text we will refer explicitly to the graph associated with a solution $\mathbf{y}$, thus we introduce the following notation. For time steps $\tau = 0, 1, \ldots, k-1$, let $E_\tau = \{(i,j) \in E : i \in A_\tau, j \in A_{\tau+1} \setminus A_\tau\}$, be the set of arcs in which the influence was exerted at time τ. Construct then the set $E^* = \bigcup_{\tau=0}^{k-1} E_\tau$. We say that the *propagation graph* $G^* = (A_k, E^*)$ is the graph induced by the solution $\mathbf{y}$. Note that the propagation graph must be acyclic, the intuition behind this fact is that a vertex should not be able to transmit influence to itself.

3. Integer linear programming formulation

There are different integer linear programming (ILP) formulations that model the propagation using variables on the arcs [Ackerman et al., 2010; Günneç et al., 2016; Raghavan e Zhang, 2015]. The following formulation is a special case of the model proposed by Fischetti et al. [2018]. For each vertex $i \in V$, let x_i be a binary variable that indicates whether i is active at the end of the diffusion process. Similarly, for each arc $(i,j) \in E$, let z_{ij} be a binary variable that indicates whether i exerts influence over j. As mentioned previously, the integer variable y_i is the amount of incentive to be paid to a vertex $i \in V$.

$$\min \sum_{i \in V} y_i$$

$$\text{s.t.} \sum_{i \in N_j} z_{ij} d_{ij} \geq t_j x_j - y_j \qquad \forall j \in V, \tag{1}$$

$$\sum_{(i,j) \in C} z_{ij} \leq \sum_{i \in V(C) \setminus \{k\}} x_i \qquad \forall k \in V(C), \text{cycle } C \subseteq E, \tag{2}$$

$$z_{ij} \leq x_i \qquad \forall (i,j) \in E, (j,i) \notin E, \tag{3}$$

$$\sum_{i \in V} x_i \geq \lceil \alpha |V| \rceil, \tag{4}$$

$$x_i \in \{0,1\} \qquad \forall i \in V, \tag{5}$$

$$y_i \in \mathbb{N}_0 \qquad \forall i \in V, \tag{6}$$

$$z_{ij} \in \{0,1\} \qquad \forall (i,j) \in E. \tag{7}$$

The objective function minimizes the total amount of incentives offered. Constraints (1) models the condition that a vertex $i \in V$ gets active only when the total of influence received from its active neighbors plus its incentive is greater than or equal to its threshold. The cycle elimination constraints in (2) generalizes the classic cycle elimination from Grötschel et al. [1985], and impose that the propagation graph of a solution must be acyclic. Meaning that the number of chosen arcs in a cycle C cannot be greater than the number of active vertices in $V(C) \setminus \{k\}$, where $V(C)$ is the set of vertices in the cycle. Constraints (3) ensures that an arc (i,j) can be chosen only if vertex i is activated. Note that if there is an arc (j,i) the cycle of size two is eliminated by constraints of type (2). Therefore, these constraints are needed only when (i,j) is not in a 2-cycle. Finally, constraints (4) impose that, by the end of the diffusion process, the number of active vertices is at least $\lceil \alpha |V| \rceil$.

Due to the number of possible cycles in the graph G, the number of constraints in (2) grows exponentially. The standard exact procedure for solving integer linear programs with an exponential number of constraints is the branch-and-cut method, which is a combination of LP-based branch-and-bound and constraint generation techniques. To generate the cycle elimination

constraints, we need to solve the separation problem for the inequalities in (2). In our approach, we implement the separation procedure proposed by Grötschel et al. [1985]. In short, the procedure adapts a shortest path algorithm to find a cycle that violates the cycle elimination constraints.

Since we are looking for the optimality conditions that will provide stopping criteria, an important method is to find lower (dual) bound $\underline{z} \leq z$ and an upper (primal) bound $\bar{z} \geq z$ such that $\underline{z} = z = \bar{z}$, where z is the optimum value for the objective function of our problem. Every feasible solution provides an upper bound, while for dual bounds the most common approach is by relaxing the integrality constraints of the original problem. Our combinatorial relaxation can be used at each node of the branch-and-bound tree to obtain lower bounds.

4. Lower bound algorithm

Consider the following aspects of the LCIP.

(i) For any solution, at least one vertex needs to be paid the whole threshold value. This follows from the fact that for any solution $\mathbf{y}$, the associated propagation graph is a DAG, which means that it has at least one vertex with no incoming arcs (source), i.e. $N_v = \emptyset$.

(ii) In the best possible case, the vertices chosen to receive the total incentive have the minimum threshold, meaning that they need less incentive.

We introduce a combinatorial algorithm to obtain a dual (lower) bound for this problem. The idea consists in using connectivity properties of a sub-graph of the input graph at each node of the branch-and-bound tree. The recursive decomposition of a problem into sub-problems generate a decision tree where its root corresponds to the original problem and each node corresponds to a smaller sub-problem. A natural branching rule in a branch-and-bound algorithm is by variable fixing. In the case of the formulation (1)-(7), we can fix the values of the binary variables $\mathbf{x}$ and $\mathbf{z}$. Suppose a binary variable, say z_{ij}, is selected to be the branching variable. Then two sub-problems are generated by fixing $z_{ij} = 0$ in one branch and $z_{ij} = 1$ in the other. We observe that fixing some arc variables z_{ij} (or vertex variables x_i) at zero means that theses arcs (or vertices) were not chosen, and this can disconnect the sub-graph related to that node of the decision tree. We are interested in using this information to increase the lower bound of each sub-problem, during the branch-and-bound algorithm.

Our algorithm uses the concept of a *Condensed Component Graph*.

Definition 1 (Condensed Component Graph)**.** *A condensed component graph H of a directed graph G is obtained by contracting the strongly connected components (s.c.c.) of G. More formally, each $v \in V(H)$ is associated to a distinct s.c.c. C_v of G and there is an arc $(u, v) \in E(H)$ if and only if there exists an arc from a vertex $i \in V(C_u)$ to a vertex $j \in V(C_v)$, where C_u and C_v are the s.c.c.'s associated with u and v, respectively.*

At each node of the branch-and-bound tree, a different sub-graph G' of the input graph G is considered. The graph G' is obtained from G by removing the arcs and vertices which were fixed at zero by the decision tree. That is, $V(G') = V(G) \setminus \{i \in V(G) : x_i \text{ is fixed in zero}\}$ and $E(G') = E(G) \setminus \{(i, j) \in E(G) : z_{ij} \text{ is fixed in zero}\}$, at the current node of the branch-and-bound tree.

Let H be the condensed component graph of G'. From now on, the graph H we consider has the arc weights and vertex thresholds defined as follows. Consider C_u and C_v be the s.c.c.'s associated to $u, v \in V(H)$ and $E_{uv} = \{(i, j) \in E(G') : i \in V(C_u) \text{ and } j \in V(C_v)\}$. We set the

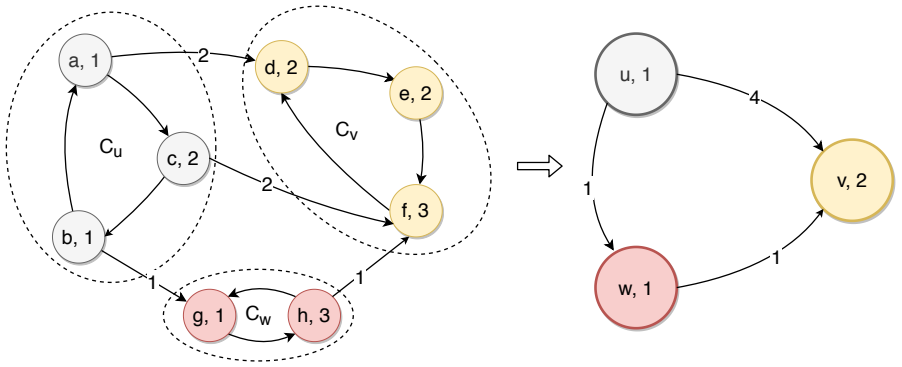

Figure 1: Example of a condensed component graph. The original graph is on the left, and the condensed component graph is on the right.

weight of each arc $(u, v) \in E(H)$ to be the sum of all arc weights that go from C_u to C_v, that is, $d_{uv} = \sum_{(i,j)\in E_{uv}} d_{ij}$. Furthermore, for each vertex $v \in V(H)$ we set $t_v = \min_{i\in V(C_v)} \{t_i\}$.

Figure 1 presents an example for the condensed component graph of a small graph. The labels in each vertex of the figure denote the name and the threshold respectively. For instance, the vertex in upper left corner has name a and threshold $t_a = 1$. In the leftmost graph, there are three strongly connected components C_u, C_w and C_v. For the sake of simplicity, we only show the arcs weights between different components. In the second graph, we have the condensed component graph with the new thresholds and weights on arcs. For instance, the arc (u, v) has weight $d_{uv} = d_{ad} + d_{cf} = 4$ and the vertex v has threshold $t_v = \min\{2, 2, 3\}$.

The algorithm follows:

Algorithm 1 LCIP - Combinatorial Lower Bound Algorithm

1: **procedure** LCIPLowerBound(G', d, t, α)
2: **if** G' is strongly connected **or** $\alpha < 1$ **then**
3: **return** $w = \min_{i\in V(G')} \{t_i\}$.
4: **else**
5: Let H be the condensed component graph of G'.
6: **return** $w = \sum_{v\in V(H)} \max\{0, t_v - \sum_{u\in N_v} d_{uv}\}$ ▷ optimum value of LCIP on H

When $\alpha = 0$ the problem is trivial, then our algorithm does not address this case.

Let w^{LP} be the lower bound obtained at the current node of the branch-and-bound tree by standard LP-relaxation, and let w be the lower bound obtained by the procedure described above. When updating the lower bound $\underline{w}$ at the current node, we simply do $\underline{w} = \max\{w^{LP}, w\}$.

We need to compute the total cost for activating all the vertices in the graph H (step (6) of Algorithm 1), as a sub-problem. As H is a condensed component graph of G', then H is directed and acyclic. Therefore, we generalized the algorithm proposed by Günneç et al. [2016] to get the exact solution of LCIP in DAGs, as Theorem 1 states.

Theorem 1. *If $\alpha = 1$ and the graph H is a DAG, step (6) of Algorithm 1 gives an exact solution for the LCIP on H.*

Proof. As $\alpha = 1$, every $v \in V(H)$ is active in an exact solution, in particular, each $u \in N_v$ is active. Thus, every incoming arc $(u, v) \in E(H)$ of v can be selected (constraints 3), because there are no cycles in H restricting such selection (constraints 2). Therefore, v receives $\sum_{u \in N_v} d_{uv}$ of influence. If $\sum_{u \in N_v} d_{uv} > t_v$ then the influence coming from neighbors is enough to activate v and y_v is kept at the minimum, i.e., $y_v = 0$. Otherwise, the minimum influence needed to activate v is $y_v = t_v - \sum_{u \in N_v} d_{uv}$. □

Due to the construction of the auxiliary structures for the relaxation, we state that the solution of the LCIP in the condensed graph is a lower bound for the problem in G'. As a result, we prove the correctness of our combinatorial relaxation.

Lemma 2. *When $\alpha = 1$, for a not strongly connected sub-graph G' of G and a condensed component graph H of G', the cost of an optimal solution of the LCIP on H is a lower bound for the cost of an optimal solution of the problem in G'.*

Proof. W.l.o.g., we prove the result for each s.c.c of G', together with its associated vertex on H. Let v be the vertex of $V(H)$ associated with the s.c.c. C_v of G'. Let $i^* \in V(C_v)$ be a vertex where $t_{i^*} = \min_{i \in V(C_v)} t_i$. The condensed component graph H is acyclic by Definition 1. So, we need to consider two cases, depending if a vertex of H is a source vertex or not.

If v is a source vertex of H, then it has no incoming neighbors, and we are obligated to pay the total threshold to activate it, i.e., pay $y_v = t_v$. By the construction of H, we have that $t_v = t_{i^*}$, then we are paying the minimum incentive needed to activate the whole s.c.c. C_v in G'. Therefore, y_v is a lower bound on the value of the solution for the s.c.c. C_v.

On the other hand, if v is not a source vertex, then let $d_v = \sum_{(u,v) \in E(H)} d_{uv}$ be the maximum possible influence that can arrive in v, meaning that all v's in-neighbors are active. So, $y_v = t_v - d_v$ is the minimum incentive needed to activate v. We need to prove that y_v is also a lower bound for the associated s.c.c. C_v in G'.

At this point, it is important to remember that the propagation graph of G' must be acyclic. This means that for each C_v at least one vertex needs to receive only external influence. From the point of view of the graph G', the maximum external influence d_v can reach the s.c.c C_v through several edges. However, there are two cases: (a) d_v arrives uniquely in vertex i^* and (b) d_v do not arrives uniquely in i^*. In case (a), i^* receives d_v of influence. Thus, paying $y_{i^*} = t_{i^*} - d_v$ is sufficient to activate i^*. As $t_v = t_{i^*}$, then $y_v = y_{i^*}$. So y_v is a lower bound to activate the cheapest vertex in C_v, and therefore is a lower bound on the value of the solution for the s.c.c. C_v. In case (b), the external influence that arrives in i^* is less than d_v, because the arcs arriving in C_v are distributed between more than one vertex. So, paying $y_{i^*} = t_{i^*} - d_v$ is not enough to activate i^*, i.e. $y_{i^*} = t_{i^*} - d_v$ is a lower bound in the activation cost of i^*. As $y_v = y_{i^*}$, then y_v is a lower bound in the activation cost of C_v.

Finally, let y^*_v be the optimum incentive given to each $v \in V(H)$. We have

$$\sum_{v \in V(H)} y^*_v = \sum_{C_v \in G'} y_{i^*} \leq \sum_{i \in V(G')} y_i,$$

where y_i is the optimum for each $i \in V(G')$. In words, for each $v \in V(H)$, the value y_v, as set by the algorithm, is a lower bound on the solution value for each associated s.c.c. C_v, and therefore they are a lower bound to the solution of G. □

Theorem 3. *The solution obtained by the algorithm is a lower bound for the* LEAST COST INFLUENCE PROBLEM *in the sub-graph* G'.

Proof. When G' is strongly connected, the result is direct by the first observation in this section. The case in which G' is not strongly connected and $\alpha = 1$ (step 2 of the algorithm) holds by Lemma 2. □

5. Computational Experiments

Our computational experiments were obtained with the branch-cut-and-price framework SCIP 6.0 running in an Intel Core i5-3210M 2.50GHz with 4GB of RAM, using Gurobi Optimizer 8.1 as the underlying LP-solver and the algorithms were implemented in C++. The test set is composed by synthetic random directed graphs and by real networks.

5.1. Synthetic graphs

As in [Fischetti et al., 2018; Günneç et al., 2016], we use the generative model proposed by Watts e Strogatz [1998] for small world random graphs. The rewiring probability parameter for the small world graph is fixed in $\beta = 0.3$. The influence weights on the arcs are chosen uniformly at random from $\{1, ..., 10\}$. Let $N(\mu, \sigma)$ be a normally-distributed random variable. For every $i \in V$, we set $t_i = \max\{1, \min\{N(\mu, \sigma), d_i\}\}$, where $d_i = \sum_{(j,i)\in E} d_{ji}$, $\mu = 0.7d_i$ and $\sigma = \frac{d_i}{|N_i|}$. We restrict the experiments with synthetic graphs for $\alpha = 1$.

Table 1: Experiments comparing the branch-and-cut with and without the proposed lower bound.

Graph	Algorithm	Time(s)	Nodes	Dual bound	Primal bound	Gap
50-4	BC	377.30	81,960.4	51.70	51.70	0.00
	BC+	**344.37**	**55,987.8**	51.70	51.70	0.00
50-8	BC	-	**37,535.2**	2.91	**846.25**	∞
	BC+	-	39,053.4	**162.60**	983.55	**6.07**
75-4	BC	1,094.52	116,853.4	61.93	**81.75**	0.54
	BC+	**875.57**	**77,507.6**	**76.66**	90.20	**0.38**
75-8	BC	-	**12,479.8**	9.48	2,529.55	∞
	BC+	-	13,792.0	**129.80**	**2,465.55**	**20.74**
75-12	BC	-	2,895.6	0.00	**3,622.80**	∞
	BC+	-	**2,768.6**	**376.25**	3,631.10	**9.70**
100-4	BC	-	93,452.5	48.14	231.56	4.17
	BC+	-	**89,074.0**	**68.89**	**223.00**	**3.13**
100-8	BC	-	6,295.2	3.29	3,279.40	∞
	BC+	-	**5,410.0**	**114.95**	**3,234.40**	**34.78**
100-12	BC	-	**1,713.2**	0.00	5,890.90	∞
	BC+	-	1,941.8	**396.30**	**5,830.75**	**14.31**
100-16	BC	-	280.4	0.00	**8,301.75**	∞
	BC+	-	**234.0**	**727.35**	8,393.90	**10.58**

Table 1 summarizes the difference in performance when we apply the lower bound in the branch-and-cut. Each value on the table is the average of 5 executions, each execution generates a new graph of a given size and average degree. In the first column are the name of instances in format n-deg where n is the number of vertices and deg is the average degree. In the other columns, BC means the branch-and-cut algorithm using only the LP-relaxation, and BC+ means we are using our combinatorial relaxation to get the lower bound. Next, we present the time in seconds for those that finished before the time limit. The time limit for these instances is set to 1800 seconds. Then we have the number of explored nodes in the branch-and-bound tree. In the last column we have the relative gap between the dual and primal bounds. We marked in bold face the results that were better. E.g., in the instance "50-4" our method (BC+) required less running time and nodes to find the optimum. See that the dual bound in BC+ was better than BC in all cases, implying smaller gaps for the cases that reached the time limit. Dashed cells in the column "Time" means that the running time reached the time limit. The symbol ∞ in the column "Gap" means the gap is infinity or very large. The gap is computed as follows: let l be the dual bound and u be the primal bound. We set the gap to ∞ if $l \leq 0$. Otherwise, the gap is $(u - l)/l$. Recall that the values in Table 1 are averages, so the gaps in the table are the average gaps.

5.2. Real world networks

To demonstrate the effects of apply the lower bound algorithm on real data, we also performed experiments with real world social networks. The datasets we use are part of the Koblenz Network Collection [Kunegis, 2013], human social network category. A short description of each social network used here is shown below.

- Wiki-vote: Wikipedia who-votes-on-whom network. Contains voting data from the inception of Wikipedia until January 2008. Vertices in the network represent users and a directed edge from vertex i to vertex j represents that user i voted on user j. This network was obtained from Network Repository [Rossi e Ahmed, 2015].

- Adolescent health: in this network each student lists his/her five best female and five best male friends. A vertex i represents a student and a directed edge (i, j) between two students shows that the student i chose the student j as a friend. Higher edge weights indicate more interactions.

- High school: contains friendships between boys in a small high school in Illinois. A vertex i represents a boy and an directed edge between two boys shows that boy i chose the boy j as a friend. The edge weights show how often that happened.

- Physicians: this network captures innovation spread among physicians in Illinois, Peoria, Bloomington, Quincy and Galesburg. A vertex i represents a physician and an edge (i, j) between two physicians means that physician i told physician j is his friend or that i comes to j if he needs advice. Edges are weightless.

- Residence hall: contains friendship ratings between residents living at a residence hall located on the Australian National University campus. The friendships (edges) are weighted from strongest to weakest tie from 5 to 1. As higher the weight the closer the individuals are.

The weights on the arcs are the original weights of the networks. On graphs with no arcs weights, we set the weights to 1. Lastly, the threshold t_i, for each vertex i, are defined in the same way as in the synthetic graphs (see Section 5.1).

Table 2 summarizes the results for the real world social networks. For each network, n is the number of vertices and m is the number of arcs. Dashed cells means that the running time reached the time limit. Here, the time limit is set to 3600 seconds. Note that for this type of graph there is small benefit in applying the dual bound algorithm, i.e., the performance of the branch and cut is almost the same whether using the lower bound or not. We believe this happens because real world social networks are usually sparse, thus few arc variables (z_{ij} in formulation (1)-(7)) are fixed in 0 in the branch and bound tree. Also, when $\alpha = 1$, all the vertex variables (x_i in formulation (1)-(7)) are fixed in 1. In this way, there are few changes in the structures of the subgraphs obtained from the branches. However, for the same instances there is a notable difference when we vary the value of α. For $\alpha = .5$ and $\alpha = .1$, our algorithm achieves better gaps in all the networks tested, see Tables 3 and 4. Observe that, in some cases, the number of nodes generated in the branch and bound tree (column "Nodes") is smaller when we do not use the lower bound, but even so the gap is improved because of the dual bound.

Table 2: Experiments on real world based social networks for $\alpha = 1$.

Network	n	m	Alg.	Time	Nodes	Dual bound	Primal bound	Gap
High School	70	366	BC	3.48	143	18	18	0
			BC+	**3.45**	143	18	18	0
Residence	217	2,672	BC	-	**13,582**	18	24	0.33
			BC+	-	13,927	18	24	0.33
Physicians	241	1,098	BC	-	189,931	137	153	0.12
			BC+	-	**186,594**	137	153	0.12
Wiki-vote	889	2,914	BC	0.26	1	3,834	3,834	0
			BC+	0.26	1	3,834	3,834	0
Adolescent	2,539	12,969	BC	-	38	656	∞	∞
			BC+	-	38	656	∞	∞

Table 3: Experiments on real world based social networks for $\alpha = 0.5$.

Network	n	m	Alg.	Time	Nodes	Dual bound	Primal bound	Gap
High School	70	366	BC	-	**1,723**	0.40	15	36.01
			BC+	-	2,115	**3**	15	**4**
Residence	217	2,672	BC	-	15,020	0	234	∞
			BC+	-	**13,298**	**6**	234	**38**
Physicians	241	1,098	BC	-	**26,495**	10.68	103.5	8.68
			BC+	-	34,815	**26.59**	**99**	**2.72**
Wiki-vote	889	2,914	BC	**934.72**	10,764	273	273	0
			BC+	970.05	10,764	273	273	0
Adolescent	2,539	12,969	BC	-	9	0	2,929.5	∞
			BC+	-	**8**	**5.25**	2,929.5	**557**

6. Conclusion

We proposed and analyzed an algorithm to compute a lower bound for LCIP based on particular properties of the problem. The experiments presented achieved better results on solving the problem. Our results show that the subject should be approached carefully, and we envision some space for improvements. For example, in dense graphs, we predict that bounds behave better

Table 4: Experiments on real world based social networks for $\alpha = 0.1$.

Network	n	m	Alg.	Time	Nodes	Dual bound	Primal bound	Gap
High School	70	366	BC	**32.17**	916	3	3	0
			BC+	32.43	**816**	3	3	0
Residence	217	2,672	BC	-	**5,122**	0	66	∞
			BC+	-	5,327	**6**	**60**	**9**
Physicians	241	1,098	BC	-	**7,953**	2	18	7.76
			BC+	**582.89**	11,196	**14**	**14**	**0**
Wiki-vote	889	2,914	BC	2195.9	23,062	52	52	0
			BC+	**1,340**	**12,573**	52	52	0
Adolescent	2,539	12,969	BC	-	**32**	0	809	∞
			BC+	-	35	**5**	809	**153**

when $\alpha = 1$ than the case when $\alpha < 1$. Thus, the lower bound could be improved by an algorithm that deals with subgraphs that are not strongly connected, even for $\alpha < 1$. In addition, it is possible to improve the experimental results if we use the lower bound algorithm to add new cutting planes on the model, selecting branching variables and choosing the next branch to explore. Therefore, more involved experiments need to be carried out, and we expect to address such issues in a continuation of this work.

7. Acknowledgements

We gratefully acknowledge the support given by CNPq (Proc. 314366/2018-0, 425340/2016-3) and FAPESP (Proc. 2015/11937-9).

References

Ackerman, E., Ben-Zwi, O., e Wolfovitz, G. (2010). Combinatorial model and bounds for target set selection. *Theoretical Computer Science*, 411(44-46):4017–4022.

Ben-Zwi, O., Hermelin, D., Lokshtanov, D., e Newman, I. (2011). Treewidth governs the complexity of target set selection. *Discrete Optimization*, 8(1):87–96.

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

Chen, W., Wang, C., e Wang, Y. (2010a). 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*, p. 1029–1038. ACM.

Chen, W., Wang, Y., e Yang, S. (2009). Efficient influence maximization in social networks. In *Proceedings of the 15th ACM SIGKDD*, p. 199–208. ACM.

Chen, W., Yuan, Y., e Zhang, L. (2010b). Scalable influence maximization in social networks under the linear threshold model. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, p. 88–97. IEEE.

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

Cordasco, G., Gargano, L., Rescigno, A. A., e Vaccaro, U. (2015). Optimizing spread of influence in social networks via partial incentives. In *International Colloquium on Structural Information and Communication Complexity*, p. 119–134. Springer.

Demaine, E. D., Hajiaghayi, M., Mahini, H., Malec, D. L., Raghavan, S., Sawant, A., e Zadimoghadam, M. (2014). How to influence people with partial incentives. In *Proceedings of the 23rd international conference on World wide web*, p. 937–948. ACM.

Fischetti, M., Kahr, M., Leitner, M., Monaci, M., e Ruthmair, M. (2018). Least cost influence propagation in (social) networks. *Mathematical Programming*, p. 1–33.

Granovetter, M. (1978). Threshold models of collective behavior. *American journal of sociology*, p. 1420–1443.

Grötschel, M., Jünger, M., e Reinelt, G. (1985). On the acyclic subgraph polytope. *Mathematical Programming*, 33(1):28–42.

Günneç, D., Raghavan, S., e Zhang, R. (2016). Tailored incentives and least cost influence maximization on social networks. Technical report, Technical report.

Kempe, D., Kleinberg, J., e Tardos, É. (2003). Maximizing the spread of influence through a social network. In *Proceedings of the 9th ACM SIGKDD*, p. 137–146. ACM.

Kunegis, J. (2013). KONECT – The Koblenz Network Collection. In *Proc. Int. Conf. on World Wide Web Companion*, p. 1343–1350. URL `http://userpages.uni-koblenz.de/~kunegis/paper/kunegis-koblenz-network-collection.pdf`.

Nemhauser, G. L., Wolsey, L. A., e Fisher, M. L. (1978). An analysis of approximations for maximizing submodular set functions-i. *Mathematical Programming*, 14(1):265–294.

Raghavan, S. e Zhang, R. (2015). Weighted target set selection on social networks. Technical report, Working paper, University of Maryland.

Rogers, E. M. (2010). *Diffusion of innovations*. Simon and Schuster.

Rossi, R. A. e Ahmed, N. K. (2015). The network data repository with interactive graph analytics and visualization. In *AAAI*. URL `http://networkrepository.com`.

Watts, D. J. e Strogatz, S. H. (1998). Collective dynamics of 'small-world'networks. *Nature*, 393 (6684):440.