http://pubsonline.informs.org/journal/ijoc

INFORMS JOURNAL ON COMPUTING
Articles in Advance, pp. 1–14
ISSN 1091-9856 (print), ISSN 1526-5528 (online)

Least-Cost Influence Maximization on Social Networks

Dilek Günneç,[a] S. Raghavan,[b] Rui Zhang[c]

[a] Department of Industrial Engineering, Ozyegin University, Istanbul 34794, Turkey; [b] Robert H. Smith School of Business and Institute for Systems Research, University of Maryland, College Park, Maryland 20742; [c] Leeds School of Business, University of Colorado, Boulder, Colorado 80309

Contact: dilek.gunnec@ozyegin.edu.tr, http://orcid.org/0000-0002-0749-2584 (DG); raghavan@umd.edu, http://orcid.org/0000-0002-9656-5596 (SR); rui.zhang@colorado.edu, http://orcid.org/0000-0002-4029-6585 (RZ)

Received: March 16, 2017
Revised: April 23, 2018; December 17, 2018
Accepted: December 25, 2018
Published Online in Articles in Advance: November 26, 2019

https://doi.org/10.1287/ijoc.2019.0886

Abstract. Viral-marketing strategies are of significant interest in the online economy. Roughly, in these problems, one seeks to identify which individuals to strategically target in a social network so that a given proportion of the network is influenced at minimum cost. Earlier literature has focused primarily on problems where a fixed inducement is provided to those targeted. In contrast, resembling the practical viral-marketing setting, we consider this problem where one is allowed to "partially influence" (by the use of monetary inducements) those selected for targeting. We thus focus on the "least-cost influence problem (LCIP)": an influence-maximization problem where the goal is to find the minimum total amount of inducements (individuals to target and associated tailored incentive) required to influence a given proportion of the population. Motivated by the desire to develop a better understanding of fundamental problems in social-network analytics, we seek to develop (exact) optimization approaches for the LCIP. Our paper makes several contributions, including (i) showing that the problem is NP-complete in general as well as under a wide variety of special conditions; (ii) providing an influence greedy algorithm to solve the problem polynomially on trees, where we require 100% adoption and all neighbors exert equal influence on a node; and (iii) a totally unimodular formulation for this tree case.

History: Accepted by David Woodruff, Area Editor for Software Tools.

Keywords: social networks • influence maximization • complexity • integer programming • strong formulation • greedy algorithm

1. Introduction

The increasing number of online communication channels has led to fast and easy exchange of experiences (ideas, information, emotions, etc.), allowing influence to play a significant role in purchasing (or product adoption) decisions. Although social influence has been recognized as a factor in decision making for a long time (see, for example, Bourne 1957, Brown and Reingen 1987, and Granovetter 1978), the interaction is now tracked easily due to readily available online customer footprints. Social networks play a fundamental role in the spread of information, ideas, and influence amongst their members. Consequently, there is significant interest in understanding the dynamics of adoption within a social network that may yield clues to better marketing strategies.

In this paper, we focus on a viral-marketing problem that has garnered significant interest amongst algorithmic researchers. Suppose we want to promote a new product over a given social network and wish for this product to be adopted by everyone (or by a given fraction of individuals) in this network. We can initialize the diffusion process by "targeting" some influential people. Then, ideally, a cascade will be caused by these initial adopters, and other people will start to adopt this product due to the influence they receive from earlier adopters. But how should we select these influential people who are targeted initially? Kempe et al. (2003) were the first ones to consider this problem in an operational framework using models from mathematical sociology (Granovetter 1978) that explicitly represent the step-by-step dynamics of adoption. They considered a budgeted version of the problem (i.e., given a budget of k seed products, identify the k individuals to target so as to maximize the adoption of the product in the social network) in a randomized setting and showed that it is NP-hard to find the optimal initial set. Based on the submodularity property of the objective function (which is due to the particular randomized assumption in the problem data they make), they developed a $(1 - 1/e)$-approximation algorithm for the problem. Their work led to a flurry of follow-up work on the problem that mostly focused on speeding up the algorithm (because their algorithm has a costly simulation in each step). Initiated by Chen (2009), another stream of follow-up work has focused on the problem in a deterministic setting with a cost-minimization aspect (i.e., instead of the marketer

being given a budget k, the desire is to find the minimum number of nodes to target in the network so that the entire network is influenced). The recent monograph by Chen et al. (2013) nicely summarizes most of the relevant work in the area.

The mathematical model studied by earlier researchers suffers from a significant practical shortcoming. It restricts the marketer to interventions where those selected for targeting receive the product gratis. Motivated by practical considerations, we consider a version of this viral-marketing problem where an individual can be partially influenced by the use of monetary inducements (e.g., coupons that reduce the price of a product instead of receiving the product for free). We believe this is a crucial aspect, and the use of tailored (i.e., partial) incentives that allows for differentiated targeting is more natural in a marketing setting. The problem we study is in a deterministic setting and seeks to minimize the cost of tailored incentives provided to individuals in a social network while ensuring that a given fraction of the network is "influenced." We refer to this problem as the *least-cost influence problem* (LCIP). Günneç (2012) and Günneç and Raghavan (2017) first describe the LCIP, where it arose in a product-design setting that took into account social-network effects.

1.1. Problem Definition

Consider a social network represented as an undirected graph $G = (V, E)$, where node set $V = \{1, 2, \ldots, n\}$ denotes the set of people in the network and edge set E shows the connections between people on the social network. Following a well-studied linear threshold model on the diffusion of innovations (Granovetter 1978), we will use the term *active* if a node has adopted the product and the term *inactive* if it has not adopted the product. In the threshold model, each inactive node $i \in V$ is influenced by an amount d_{ij} (referred to as the *influence factor*) by its neighbor node j (i.e., there is an edge in the graph between nodes i and j) if node j is active (i.e., has already adopted the product). For each node in the network, $i \in V$, there is a threshold, denoted by b_i. This threshold represents how easily a node can be influenced. We permit a payment p_i, which is the tailored incentives for a node $i \in V$. Also, α is given as the desired penetration rate, taking values between 0 and 1 ($0 \le \alpha \le 1$).

All nodes are inactive initially. Then, we decide the tailored incentives p_i for each node $i \in V$. Now, a node i becomes active immediately if $p_i \ge b_i$. (That is, if the payment is greater than or equal to the threshold. Under linear scaling, it is without loss of generality the units of the payments (typically monetary units) and threshold (typically utility) are equivalent. For example, if p_i payment units at node i were equivalent to $\gamma_i p_i$ threshold units at the node, then we would simply scale the threshold b_i and incoming influence d_{ij} by dividing them by γ_i.) After that, in each step, we update the states of nodes by the following rule: An inactive node i becomes active if the sum of the tailored incentive p_i and the total influence coming from its active neighbors is at least b_i. The process continues until there is no change in the state of the network (i.e., no additional nodes are becoming active). The goal is to find the minimum total payment (i.e., $\sum_{i \in V} p_i$), while ensuring that at least $\alpha|V|$ nodes are active by the end of this activation process.

Note that the assumption that all nodes are inactive is without loss of generality. If some nodes were active at the outset, we can propagate their influence and reduce the problem to a smaller one where all nodes are inactive initially. We also note that in a deterministic setting, there is no benefit to delaying the payment of the tailored incentive. Hence, all incentives paid to a node i may be viewed as being paid at the outset of the process.

A simple integer-programming model for the LCIP introduces the notion of time periods $t = 1, 2, \ldots, T$ when nodes can become active. This is to capture the order in which nodes become active in the social network. Binary variable y_{it} denotes whether node i is active in time period t (note that in the diffusion model, once a node becomes active, it remains active), and so $y_{iT} = 1$ indicates that node i is active at the end of the diffusion process. Let $a(i)$ denote the set of node i's neighbors. The formulation is as follows:

MIP1 :

$$\min \sum_{i \in V} p_i, \tag{1}$$

$$\text{s.t. } y_{i0} = 0 \qquad \forall i \in V, \tag{2}$$

$$p_i + \sum_{j \in a(i)} d_{ij} y_{j(t-1)} \ge b_i y_{it} \quad \forall i \in V,\ t = 1, 2, \ldots, T, \tag{3}$$

$$\sum_{i \in V} y_{iT} \ge \alpha |V|, \tag{4}$$

$$y_{it} \in \{0, 1\}, p_i \ge 0 \quad \forall i \in V,\ t = 1, 2, \ldots, T. \tag{5}$$

Here, the objective (1) is to minimize the sum of the incentives given over the network. Constraint set (2) models the initial condition that all nodes are inactive initially. Constraint set (3) models the diffusion process—an inactive node i becomes active if the sum of the tailored incentive p_i and the total influence coming from its active neighbors is at least b_i. Constraint set (4) ensures that the desired penetration rate is achieved at the end of the diffusion process. Note that because there are $|V|$ nodes, the number of time indices required for the diffusion process to complete should be less than or equal to $|V|$. However, because

we do not know a priori how quickly the diffusion process terminates, we set $T = |V|$.

1.2. Related Literature

Given a social network, finding a target set of customers of the smallest possible size that could lead the whole network to adopt the product through influencing others is referred to as the Target Set Selection (TSS) Problem (Chen 2009) in the literature. In the TSS problem, given a connected undirected graph, each node has associated with it a critical value g_i, which takes values between 1 and the degree of the node, denoted by $\deg(i)$. All nodes are inactive initially. A selected subset of nodes, the target set, is activated (i.e., switched to an active state). Next, the states of nodes are updated step by step with respect to the following rule: An inactive node i becomes active if at least g_i of its neighbors are active in the previous step. The goal is to find the minimum-size target set, while ensuring that all nodes are active by the end of this diffusion process. Chen (2009) showed that the TSS problem is hard to approximate within a polylogarithmic factor. He also provided a polynomial algorithm for the TSS on trees.

Raghavan and Zhang (2018c, 2019) discuss the *weighted* TSS (WTSS) problem. In the WTSS problem, for each node $i \in V$, there is a weight, denoted by b_i, that models the fact that different nodes require differing levels of effort to become active (in practice, it is reasonable to assume that different individuals require different amounts of effort to be convinced to adopt the product). Note that the WTSS problem is closely connected to the LCIP. For the WTSS problem, all neighbors have equal influence—that is, $d_{ij} = d_i$ for all neighbors j of node i. The assumption that all neighbors of a node exert equal influence is relevant, especially when privacy concerns are present in social networks. In this way, the influence does not depend on the identity of the neighbor and the information on the strength of the relationship. Although an individual is affected equally by each neighbor, this influence factor may be different for each individual. The critical value g_i (for the WTSS problem) is equal to $\lceil \frac{b_i}{d_i} \rceil$. Unlike the LCIP, which allows for the payment of a tailored incentive, in the WTSS problem, a node is either paid the full amount b_i or nothing at all. This is where the two problems differ. Finally, the WTSS problem is concerned with 100% adoption. Raghavan and Zhang (2019) describe the polytope of the WTSS problem on trees and cycles. Raghavan and Zhang (2018c) describe a branch-and-cut approach for arbitrary graphs and apply it to 180 real-world graph instances (with up to approximately 155,000 nodes and 327,000 edges). Their branch-and-cut approach finds solutions that are on average 0.90% from optimality and solves 60 out of the 180 instances to optimality.

As mentioned earlier, Kempe et al. (2003) considered a budgeted version of the problem in a randomized setting that we will refer to as the *influence maximization problem* (IMP). They make a particular assumption (for technical convenience and on which the submodularity property their results critically depend) on the distribution of the threshold values b_i. Roughly, this can be interpreted as the thresholds b_i being distributed uniformly in the range $[0, L]$, where $L = \max_{i \in V}\{\max\{b_i, \sum_{j \in a(i)} d_{ij}\}\}$. (In their notation, the threshold values lie between 0 and 1. The data in our setting can be converted to theirs and vice versa by dividing or multiplying all data values by L, respectively.) Furthermore, their objective is to maximize the number of people influenced given that only k individuals are allowed to be fully influenced (i.e., provided the product for free). Wu and Küçükyavuz (2018) studied the IMP in a two-stage stochastic-programming framework. They proposed a delayed constraint-generation algorithm by taking advantage of the submodularity property in the objective function and conducted computational experiments on the stochastic version of the IMP. Although promising, their approach is limited in that it is only computationally viable for seed sets of size five, whereas heuristics in the literature easily deal with problems where the size of the seed set is considerably larger (e.g., Kempe et al. (2003) considered seed sets with 100 nodes).

Subsequent to Günneç (2012), Demaine et al. (2014) presented a fractional version of the IMP considered by Kempe et al. (2003). In their model, identically to the LCIP, nodes can be partially influenced via a payment, and the goal is to maximize the number of nodes influenced for a given budget. They retain the same technical assumption (as Kempe et al. 2003) on the uniform distribution of thresholds and consider the budgeted version of the problem with a budget of kL. In theory, they showed that the fractional version of the IMP has the same computational complexity as the IMP. That is, the submodularity property of the objective function holds (which is again due to the particular randomized assumption on the uniform distribution of thresholds), yielding the same $(1 - 1/e)$-greedy approximation algorithm for the problem. In practice, they showed that the solutions of the two versions could be significantly different: The fractional allocation can improve the influence greatly (i.e., with a budget of k, where $L = 1$ in their setting, the fractional allocation model can influence a larger number of nodes).

Cordasco et al. (2015) studied a problem that corresponds to a specialized version of the LCIP where the influence factor is the same over the whole network

(i.e., $d_i = d_j \ \forall i, j \in V$) and provided a polynomial time algorithm for complete graphs and trees. As we will see, our results will provide a trivial algorithm for the problem on trees (they provide a nontrivial algorithm for the problem on trees).

Although the focus of our paper and discussion of related literature has been on mathematical models for influence maximization, we should point out that there is a large body of (complementary) experimental and behavioral research focused on diffusion on real-world networks. These observational studies focus on understanding influence dynamics such as identifying influencer characteristics, the effect of network structure and influencer location, and peer influence and types of sharing on the final spread (Centola 2010, Tucker and Zhang 2010, Aral and Walker 2012, Bapna and Umyarov 2015).

1.3. Our Contributions

Much of the previous work involving influence maximization has focused on heuristics and approximation. In the case of partial incentives, the only prior papers (Demaine et al. 2014, Cordasco et al. 2015) in this area that we are aware of focus on heuristics and approximation. Our research is motivated by the desire to develop optimization-based approaches for the LCIP. In Section 2, we first study the complexity of the LCIP and show that the LCIP is NP-complete. We then consider several special conditions, including (1) equal influence from neighbors, (2) 100% adoption, and (3) restricting the problem to trees. Specifically, we show that the LCIP is NP-complete, even on bipartite graphs when all neighbors exert equal influence, and we do not require 100% adoption. When we require 100% adoption, the problem remains NP-complete (and is in fact APX-hard). For trees, when neighbors exert *unequal* influence and we require 100% adoption, the problem remains NP-complete.

Next, in Section 3, we focus on the case when neighbors exert equal influence and 100% adoption is required. We study the LCIP on trees. Our contributions in this regard are threefold. First, we propose two polynomial algorithms for the LCIP on trees. We describe a greedy algorithm that has $O(|V|_{\log}|V|)$ running time. Second, we show a dynamic programming (DP) algorithm that has a better $O(|V|)$ running time. The DP algorithm also works in the unequal influence case, although the running time is no longer polynomial (it is dependent on that of the mixed 0-1 knapsack problem). Third, we present a totally unimodular (TU) formulation for the LCIP on trees. This TU formulation is built on the influence propagation network and makes use of special structures about the amount of influence passing along an arc.

2. Problem Complexity

In this section, we show that the LCIP is NP-complete. We then consider several special conditions and establish their complexity. Table 1 summarizes our results.

Consider the decision version of the LCIP. Given an instance of the LCIP with $(H, \mathbf{b}, \mathbf{d}, \alpha, k)$, where $H = (V_H, E_H)$ is the graph, $\mathbf{b}$ is the threshold vector, $\mathbf{d}$ is the influence factor vector, k the budget, and α is the desired penetration rate; does there exist a payment/inducement vector $\mathbf{p}$ satisfying $\sum_{i \in V_H} p_i \leq k$ that achieves the desired market penetration (i.e., the number of nodes that are influenced is at least $\alpha|V_H|$)?

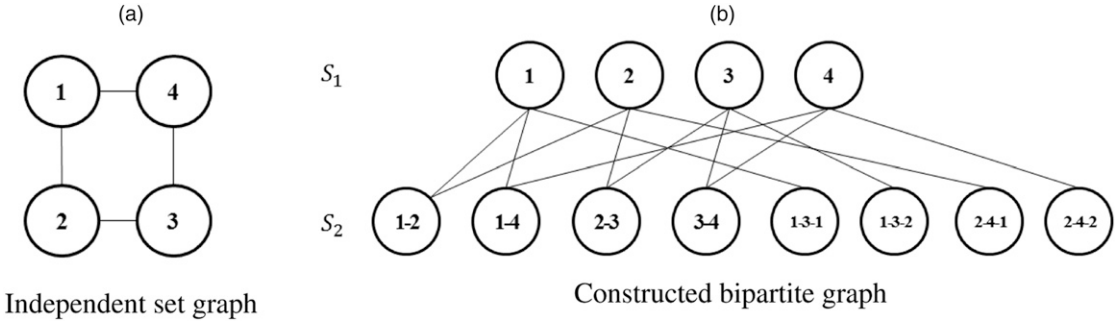
Theorem 1. *The LCIP is NP-complete.*

Proof. Given a graph $G = (V, E)$ and a number t, the decision version of the independent set problem asks whether there is an independent set in G of cardinality t. We will transform an instance of the independent set problem to an instance of the LCIP. To do so, we construct a bipartite graph $H = (S_1 \cup S_2, E_H)$ as follows. For each node $i \in V$, we create a node in S_1. Then, we create nodes in S_2. For each edge $\{i, j\}$ in E, we create a node denoted by *i-j* in S_2 and connect nodes i and j in S_1 to it. For example, as shown in Figure 1, for edge $(1, 2)$, we create node 1-2 in S_2 and connect it with nodes 1 and 2 in S_1. For any edge $\{i, j\}$ not in E (i.e., a possible edge that does not exist in the graph), we create two nodes in S_2, denoted by *i-j*-1 and *i-j*-2. We connect node i in S_1 to node *i-j*-1 in S_2 and node j in S_1 to node *i-j*-2. In Figure 1(a), edge $(1, 3)$ is not present in G. We add nodes (1-3-1) and (1-3-2) to S_2 and connect them to nodes 1 and 3, respectively. The number of nodes in H is $|V| + |E| + 2(\frac{|V|^2 - |V|}{2} - |E|) = |V|^2 - |E|$, and the number of

Table 1. Summary of Complexity Results

$0 < \alpha < 1$			$\alpha = 1$		
	Equal influence	Unequal influence		Equal influence	Unequal influence
Arbitrary graphs	NP-hard	NP-hard	Arbitrary graphs	APX-hard	APX-hard
Bipartite graphs	NP-hard	NP-hard	Trees	P	NP-hard

Figure 1. Illustration of the Reduction from the Independent Set Problem



edges in E is $|V|(|V|-1)$. Next, for each node $i \in S_1 \cup S_2$, we set b_i as 1. For the influence factors, if a node $i \in S_1$, we set $d_{ij} = 0\ \forall j \in a(i)$, and if a node $i \in S_2$, we set $d_{ij} = 1\ \forall j \in a(i)$. We set $\alpha = \frac{k|V|}{|V|^2-|E|}$ and $k = t$.

We claim that there exists a payment vector $\mathbf{p}$ that satisfies the budget (i.e., $\sum_{i \in V_H} p_i \le k$) and meets the desired penetration rate (i.e., the number of nodes influenced is at least $\lceil \alpha(|V|^2 - |E|) \rceil = k|V|$) if and only if G has an independent set of size t. Notice, in the constructed LCIP instance, it suffices to only make payments to nodes in S_1. Because all b_i values are 1 and d_{ij} values are 0 for nodes $i \in S_1$ and 1 for nodes $i \in S_2$, if we ever make a payment to a node j in S_2 (i.e., $p_j = 1$), we can get (at least) the same market-penetration rate by reallocating this payment to its neighbor in S_1. Hence, we can focus on solutions that only make payments to nodes in S_1. Each node in S_1 has exactly $|V| - 1$ neighbors, and two nodes in S_1 share one neighbor if and only if they are neighbors in G. So, when S ($\subseteq S_1$) is the set of nodes receiving unit payment, the number of influenced nodes in V_H can be calculated as $|V||S| - |S_c|$, where $S_c = \{\{i,j\} \in E : i,j \in S\}$. Thus, we can see that for a given budget k, we can influence at least $k|V|$ nodes in V_H if and only if $|S| = k$ and $S_c = \emptyset$. Looking at the definition of S_c, it implies that S is an independent set of size $t(= k)$ in G. □

Corollary 1. *The LCIP is NP-complete even when all neighbors of a node exert equal influence (i.e., $d_{ij} = d_i$ for all $i \in V$).*

Proof. Notice that in the transformation used in the proof of Theorem 1 all neighbors of a node exert equal influence. □

Corollary 2. *The LCIP is NP-complete on bipartite graphs.*

Proof. The proof follows from the bipartite graph instance constructed in Theorem 1. □

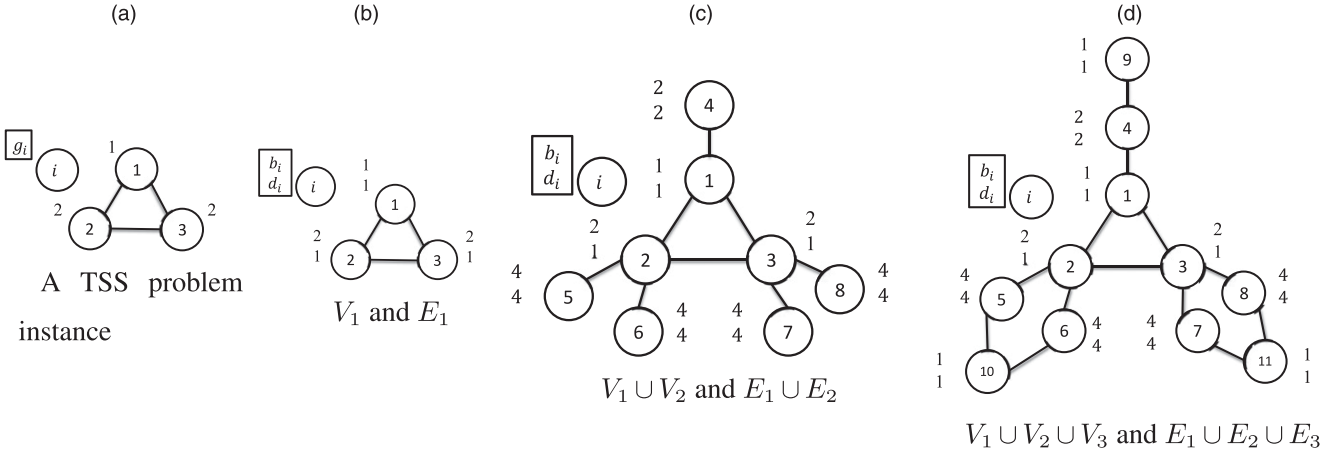
Consider the penetration rate $\alpha = \frac{k|V|}{|V|^2-|E|}$ obtained from the transformation. This can be written as $\alpha = \frac{k}{|V|-\frac{\Delta}{2}}$, where Δ represents the average degree of nodes in a graph. It is easy to see that, by varying the size of the graph, the average degree Δ, and k, a continuum of values strictly between 0 and 1 may be obtained for α. If the instance for the independent set problem has at least one edge, we will never select all nodes in S_1, and thus 100% adoption is not feasible. It is easy to see, then, if $\alpha \ge 1$, it means the specific t value is infeasible for the independent set problem. Or $k \le |V| - \frac{\Delta}{2}$. As an aside, this implies that an upper bound on the size of an independent set in a graph G is given by $\min\{\lfloor |V| - \frac{\Delta}{2} \rfloor, \lceil |V| - \frac{\Delta}{2} - 1 \rceil\}$. This naturally raises questions about the complexity of the problem when we require 100% penetration. We answer this question below.

Theorem 2. *Unless $NP \subseteq DTIME(|V|^{\text{polylog}(|V|)})$, the LCIP with $\alpha = 1$ cannot be approximated within $O(2^{\log^{1-\epsilon}|V|})$ for any fixed constant $\epsilon > 0$.*

Proof. We will prove the theorem by a reduction from the TSS problem. Consider a TSS problem instance on an undirected graph $G_t = (V_t, E_t)$, where each node i in V_t has critical value g_i (recall for the TSS problem that these are the number of nodes of a neighbor that must adopt before node i is influenced). We construct an LCIP instance based on the given TSS problem instance and show that the two problem instances have optimal solutions with identical costs. Because Chen (2009) proves that the TSS problem cannot be approximated within a ratio of $O(2^{\log^{1-\epsilon}|V_t|})$ for any fixed constant $\epsilon > 0$, unless $NP \subseteq DTIME(|V_t|^{\text{polylog}(|V_t|)})$, the same result for the LCIP follows.

From G_t for the TSS problem instance, we construct a new graph $G = (V, E)$ to create an LCIP instance as follows. First, we copy the entire graph G_t into G. We denote these nodes and edges as V_1 and E_1, respectively (they are identical to V_t and E_t), and for each node i in V_1, set its $b_i = g_i$ and $d_i = 1$ (for notational convenience, we use d_i for node i's influence factor when its neighbors exert equal influence). This is illustrated in Figure 2(b). Next, for each node i in V_1, we add g_i nodes and connect all of them to node i. These new nodes and edges are denoted by V_{i2} and E_{i2}, respectively. For each of these nodes $j \in V_{i2}$ set $b_j =$

Figure 2. An Illustration of the Reduction from the Target Set Selection Problem



$d_j = 2b_i$. Let $V_2 = \cup_{i \in V_1} V_{i2}$ and $E_2 = \cup_{i \in V_1} E_{i2}$. This is illustrated in Figure 2(c). Finally, for each i in V_1, one more node is added and connected to all nodes in V_{i2}. v_{i3} and E_{i3} are used to denote this new node and its associated edges. Also, let $V_3 = \cup_{i \in V_1} v_{i3}$ and $E_3 = \cup_{i \in V_1} E_{i3}$. For each node i in V_3, set $b_i = 1$ and $d_i = 1$. This is illustrated in Figure 2(d). We consider the LCIP instance (with $\alpha = 1$) on $G = (V_1 \cup V_2 \cup V_3, E_1 \cup E_2 \cup E_3)$. Notice in the LCIP instance created if a node in $i \in V_1$ is activated, it will activate all nodes in V_{i2}, which will activate v_{i3}. Similarly, if node v_{i3} is activated, it will activate all nodes in V_{i2}, which will activate node i.

If the TSS problem instance (on G_t) has an optimal target set S with size k, then we can find a payment vector with total amount k for the LCIP instance to activate the entire graph (G) in the following way. For each node i in S, we find its corresponding node in V_1 (continue denoting this node as i) and pay node v_{i3} an amount 1. By the preceding arguments, this will activate all nodes corresponding to S in V_1. But because G_1 and G_t are identical, all nodes in V_1 will become active as S is a feasible target set, which again by the preceding arguments will activate the rest of the graph G.

If the LCIP instance has an optimal payment vector with total amount k, we can find a feasible solution for the TSS problem instance with size k. Observe, in the LCIP instance created, any nonzero payment to a node (in an optimal solution) must be at least 1. Hence, it suffices to focus on solutions to the LCIP that only make nonzero payments to nodes in V_3 (because any payment of 1 to a node v_{i3} will activate all nodes in V_{i2}, which will activate node i). Consider such an optimal solution. For each node v_{i3} in V_3 that receives a payment (of 1), we add the corresponding node i (in V) to the target set S. The cardinality of S is k and by the previous arguments S is a feasible target set for G_t (because G_1 and G_t are identical). □

2.1. Unequal Influence Factors

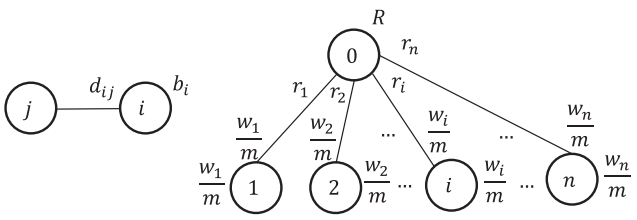
We prove that when a node i receives unequal influence from its neighbors and 100% penetration is required, the LCIP is NP-complete on trees. Recall that each node i has a threshold value b_i. Also, we use d_{ij} to denote the influence factor, which captures how much node j influences node i if node j has become active before node i.

Theorem 3. *The LCIP with unequal influence and 100% adoption is NP-complete on trees.*

Proof. Without loss of generality, we assume all input data are positive integers. The decision version of the 0-1 knapsack problem is defined as follows: Given a set N of n items numbered from 1 up to n, each item i with a weight w_i and a value r_i, along with a maximum weight capacity W, can we select a subset of these items such that a value of at least R will be achieved without exceeding the weight W?

We construct a star network (which is a tree) from this 0-1 knapsack problem. For each item i, we put a node i in the graph as a leaf node. After that, we add one extra node and label it as node 0, which is the central node. All leaf nodes connect to the central node, but do not connect to each other. Thus, there are $n + 1$ nodes numbered from 0 up to n in the graph. Next, we find a value $m = \max\{w_i + 1 : i \in N\}$. Then, for each leaf node i, we have $b_i = d_{i0} = \frac{w_i}{m}$. For the central node 0, we have $d_{0i} = r_i$ for all $i \in a(0)$ and $b_0 = R$. The constructed star is shown in Figure 3. The decision question is: Can we find a payment vector without its total cost exceeding $\frac{W}{m}$ to activate the whole network? In the constructed LCIP instance, each leaf has weight strictly less than 1 and provides an integer amount of influence to the central node 0. Given that R is an integer, it is easy to see that we should never pay the central node 0 any incentives. So, it is equivalent to ask: Can we select a subset of leaf nodes such that the incoming influence of the central node is at least R and the total cost of those

Figure 3. Transforming a 0-1 Knapsack Problem to the LCIP with Unequal Influence on Stars



selected leaf nodes does not exceed $\frac{W}{m}$? Therefore, if the answer is "yes," those selected leaf nodes also solve the 0-1 knapsack problem. □

3. LCIP on Trees

The study of the LCIP on trees is important for several reasons. First, Adcock et al. (2013) empirically demonstrated that realistic social and information networks have meaningful large-scale tree-like structure in terms of decomposability. Equally important, the proposed solution techniques for trees play a crucial part in developing a branch-and-cut approach to solving the LCIP on arbitrary graphs (Günneç et al. 2018).

From here on, for the LCIP, we assume that all neighbors of a node exert equal influence (recall that this is appropriate in settings with privacy concerns in social networks), and we require 100% adoption ($\alpha = 1$). Consequently, each node i in the network has associated with it two parameters, b_i and d_i, that represent the threshold and the influence factor for that node. Without loss of generality, we assume $d_i \le b_i \le \deg(i)d_i$. If $b_i \le d_i$, we would either pay node i the full amount b_i to activate it, or it will become active from the influence of a single neighbor (i.e., either $p_i = 0$ or $p_i = b_i$). Thus, we can simply update $d_i = b_i$ in cases where $b_i < d_i$. If $b_i > \deg(i)d_i$ node i must be paid a minimum of $b_i - \deg(i)d_i$. This cost can be taken care of in preprocessing and b_i is updated to equal $\deg(i)d_i$.

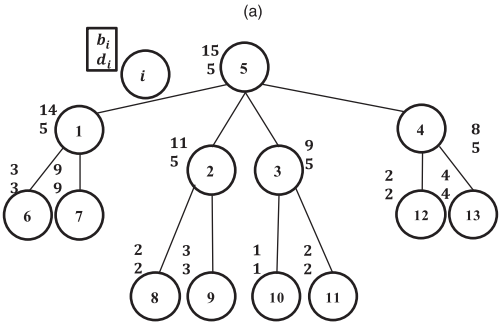
3.1. Greedy Algorithm

We now describe an $O(|V|_{\log}|V|)$ greedy algorithm. In each step, from among the inactive nodes, we select the node with the smallest value of d_i (ties can be broken arbitrarily) and pay it the amount of its threshold b_i. Next, we carry out the propagation process from this newly activated node. We update the thresholds by lowering the value of b_i by the amount of (incoming) influence. After that, all active nodes are removed from the graph.

Figure 4 illustrates the greedy algorithm. Figure 4(a) shows the instance of the LCIP on a tree. There are 13 nodes, and the numbers next to each node show the threshold and the influence factor for that node. Node 10 has the smallest influence factor ($d_{10} = 1$), so it is selected and paid $b_{10} = 1$. Newly activated node 10 sends influence to node 3, causing its threshold to be lowered to $b_3 = 4$. Because $d_3 > b_3$, now we also update $d_3 = b_3 = 4$. Next, node 8 is selected because it has the lowest influence factor value ($d_8 = 2$, recall ties are broken arbitrarily) and paid $b_8 = 2$. Its influence causes the threshold on node 2 to be lowered to 6. Then, node 11 is selected and paid $b_{11} = 2$, which causes node 3 to become active. This propagates to node 5 and reduces its threshold to 10. After that, node 12 is selected and paid $b_{12} = 2$, which causes the threshold and influence factor of node 4 to be updated to 3. Then, node 6 is selected and paid $b_6 = 3$, which causes the threshold of node 1 to be updated to 9. Next, node 9 is selected and paid $b_9 = 3$, which causes node 2's threshold and influence factor to be updated to 1. Then, node 2 is selected and paid $b_2 = 1$, which causes node 5's threshold to be updated to 5. Next, node 4 is selected and paid $b_4 = 3$, which causes nodes 5 and 13 to become active. This influence propagates from node 5 to 1, causing node 1's threshold and influence factor to be updated to 4. Finally, node 1 is selected and paid $b_1 = 4$, which causes node 7 to become active resulting in all nodes being active.

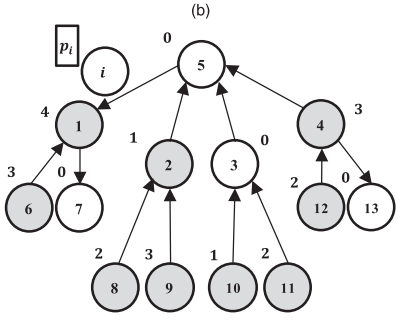
We show the correctness of the greedy algorithm by proving that there exists an optimal solution to the

Figure 4. Greedy Algorithm for the LCIP on a Tree



An LCIP instance

Step	Node Selected	Payment
1	10	1
2	8	2
3	11	2
4	12	2
5	6	3
6	9	3
7	2	1
8	4	3
9	1	4



The final solution

LCIP on a tree where node $k = \arg\min\{d_i : i \in V\}$ is paid its full threshold value b_k. The greedy algorithm recursively applies this property to obtain an optimal solution.

Claim 1. *Given an instance of the LCIP on a tree, there exists an optimal solution where node $k = \arg\min\{d_i : i \in V\}$ is paid its full threshold value b_k.*

Proof. In the LCIP, to initialize the influence-propagation process, at least one node must be paid its full threshold. Such a node will propagate influence out (i.e., it will not receive influence). Consider an optimal solution P^*, where node k is not paid its full threshold value b_k. That means it must receive influence from at least one of its neighbors (say, node l). This node l must either be paid its full threshold value (b_l) in P^* or it must receive influence from at least one of its neighbors (different from node k). Repeating this process, we can identify a node j that is paid its full threshold value and from whom influence propagates to node k via a directed path. Furthermore, along this path from node j to node k, all nodes other than node j are receiving payments strictly less than their full threshold values. We can now change the solution as follows. We will reverse the propagation of influence on this path from node k to node j. This means the payment to node j decreases by an amount of d_j its influence factor (and it is no longer paid its full threshold value), and the payment to node k increases by at most d_k its influence factor. The payments for all other nodes remain the same. Because $d_k \leq d_j$, the cost of this solution does not increase. Repeating this argument until node k no longer receives any influence from one of its neighbors proves the claim. □

Theorem 4. *The greedy algorithm solves the LCIP on a tree optimally in $O(|V|\log|V|)$ time.*

Proof. The greedy algorithm is based on repeatedly applying Claim 1. In the first step, the node with the smallest influence factor is paid its full threshold value. Once influence is propagated from the node that has just received its full payment and been removed from the graph and the solution is updated, we are left with smaller LCIPs on separate trees. Claim 1 holds separately to each one of these trees, and so in the next step, the node with the smallest influence factor can be selected and paid its full threshold value (in this updated graph). It takes $O(|V|\log|V|)$ time to initially sort the nodes based on their d_i values. After that, in each step, updating the sorted list (when the d_i value of a node changes) takes $O(\log|V|)$ time, and there are at most $|E| = |V| - 1$ updates. □

3.1.1. Special Case of Equal Influence Factors. Cordasco et al. (2015) studied a problem that corresponds to a specialized version of the LCIP where the influence factor is equal to 1 over the entire network and provided a polynomial time algorithm for complete graphs and trees. This can also be viewed as a setting where all the influence factors are equal (i.e., $d_i = d_j \ \forall i, j \in V$) and the threshold values are integral multiples of the influence factors. They provide a fairly complex algorithm for this specialized version of the LCIP on trees. On the other hand, when we apply the greedy algorithm to this problem, the result is a trivial algorithm! Pick any node and pay it the remaining value of its threshold.

3.2. Dynamic Programming Algorithm

We now describe a dynamic programming algorithm with a better $O(|V|)$ running time. Another advantage of the DP algorithm over the greedy algorithm is the fact that the DP algorithm can be applied when neighbors of a node have unequal influence (the running time is dependent on that of the mixed 0-1 knapsack problem and is no longer polynomial), whereas the greedy algorithm becomes a heuristic.

The DP algorithm decomposes the tree into subproblems. Each subproblem is used to find the most promising solution candidates (at most two), where one of them will be part of the final solution of the tree. A subproblem is defined on a star network, which has a single central node and (possibly) multiple leaf nodes. By solving the subproblem, we have one solution candidate for the case where there is influence coming into the central node along the edge that connects the star to the rest of the tree and one solution candidate for the case where influence goes out of the central node on this link (to its parent). Next, the star is compressed into one single leaf node for the next star network. This process is repeated until we are left with a single (last) star with its central node as the root node of the tree. After we exhaust all subproblems, a backtracking method is used to combine the solutions from star subproblems and obtain the final solution (set of nodes that are paid incentives along with the incentive amounts) for the tree. The pseudocode of the proposed algorithm is shown in Algorithm 1. The global variable TC has the cost of the optimal solution.

Algorithm 1 (DP Algorithm for the LCIP on Trees)

1: Arbitrarily pick a node as the root node of the tree and let $TC = 0$.
2: Define the order of star problems based on the bottom-up traversal of the tree.
3: **for** each star subproblem **do**
4: StarHandling
5: **end for**
6: SolutionBacktrack

We now discuss how to solve the LCIP on a star. Let c denote the central node of a star (all the other nodes are leaf nodes) and refer to this star as star c. To select

which nodes to give incentives to on a star, we focus on the central node c. Any leaf node i with $b_i \geq d_c$ can be neglected (recall for all leaf nodes their thresholds are equal to their influence factors). When $b_i \geq d_c$, giving b_i units of incentives to a leaf node i sends d_c units of influence to node c; thus, the decrease in threshold is less than or equal to what we spend ($d_c \leq b_i$). We are no worse off giving the incentive directly to the central node c and never use such leaf nodes (this can also be seen by invoking the greedy algorithm on the star). We collect the nodes with thresholds less than d_c in set S and sort them in increasing order of their thresholds. The nodes in S are candidates to receive incentives (in addition to the central node). Let $g_c = \lceil \frac{b_c}{d_c} \rceil$ be the number of active neighbors required to activate node c if no incentives are paid to it. The cost of the solution to the LCIP on a star depends on the size of the set S. When $|S| \geq g_c$ (i.e., there are more than enough leaf nodes), the solution is to pay the first $(g_c - 1)$ nodes in S an amount equal to their threshold and then to compare the threshold of the g_c-th leaf node in S (b_{g_c}) against the remaining threshold $(b_c - (g_c - 1)d_c)$ needed to activate the central node c. If $b_{g_c} < (b_c - (g_c - 1)d_c)$, we pay the g_c-th leaf node b_{g_c}; else, we pay the central node $(b_c - (g_c - 1)d_c)$. If $|S| < g_c$, then all nodes in the set S are paid incentives equal to their thresholds, and the remaining amount of the threshold of the central node $(b_c - |S|d_c)$ is paid directly to the central node. Here, we assumed that there is no influence coming into the central node from its parent. For the situation where the central node receives influence from its parent, we simply reduce b_c to $b_c - d_c$ on the star and, accordingly, update $d_c = \min\{b_c, d_c\}$, $g_c = \lceil \frac{b_c}{d_c} \rceil$, and solve the problem on the star.

Algorithm 2 (StarHandling)

Input: star c

$(X^c_{NI}, \mathbf{p}^c_{NI}, C^c_{NI}) \leftarrow$ SolveStar(star c, no-influence).
if star c is the last star **then**
  $TC = TC + C^c_{NI}$.
else
  $(X^c_I, \mathbf{p}^c_I, C^c_I) \leftarrow$ SolveStar(star c, with-influence).
  The compressed node's threshold is $C^c_{NI} - C^c_I$.
  $TC = TC + C^c_I$.
end if
function SolveStar(a star c, flag)
  if flag is with-influence **then**
    $b_c = b_c - d_c$ and $d_c = \min\{b_c, d_c\}$.
  end if
  Let $g_c = \lceil \frac{b_c}{d_c} \rceil$ and $S = \{i \mid b_i < d_c, \quad i \in L(c)\}$.
  if $|S| \geq g_c$ **then**
    Let S_{g_c} and $S_{g_c - 1}$ be the sets of the first g_c and $(g_c - 1)$ nodes in S, respectively.
    if $\sum_{i \in S_{g_c}} b_i \leq \sum_{i \in S_{g_c - 1}} b_i + b_c - (g_c - 1)d_c$ **then**
      $X \leftarrow S_{g_c}$, $p_i = b_i$ for $i \in X$.
    else
      $X \leftarrow S_{g_c - 1}$, $p_i = b_i$ for $i \in X$, and $p_c = b_c - (g_c - 1)d_c$, $X \leftarrow X \cup c$.
    end if
  else
    $X \leftarrow S$, $p_i = b_i$ for $i \in S$, and $p_c = b_c - |S|d_c$, $X \leftarrow X \cup c$.
  end if
  $C = \sum_{i \in X} p_i$.
  return $X, \mathbf{p}, C$.
end function

After we determine the two solution candidates for the current star subproblem, the star is compressed into a single node for the next star subproblem. If in the optimal solution, the central node c receives influence from its parent, the cost of the solution is denoted by C^c_I. If in the optimal solution, the central node c sends influence to its parent, the cost of the solution on the star is denoted by C^c_{NI}. Thus, the amount C^c_I (which is smaller) must be incurred for the star at a minimum in the optimal solution. The incremental amount $C^c_{NI} - C^c_I$ must be paid if the star sends influence to its parent in the optimal solution. Thus, when we compress the star into a single node for the next star subproblem, the threshold for the compressed star is $C^c_{NI} - C^c_I$. Because the compressed star is a leaf node for the next star, its influence factor is also set to $C^c_{NI} - C^c_I$. This DP calculation procedure is repeated until we arrive at the root node of the tree. Here, because there is no possibility of external influence to the root node, there is only one solution candidate.

Algorithm 2 provides the pseudocode associated with this calculation procedure. At its core is the function SolveStar that finds the optimal solution for a given star. The function returns X (set of nodes that are given incentives), $\mathbf{p}$ (vector of partial incentives given, i.e., $\{p_i | i \in V\}$) and C (total cost of the star). In Algorithm 2, the subscripts NI and I and superscript c (for X, $\mathbf{p}$, and C) represent the outputs in the cases of no influence, influence, and star c, respectively. Also, $L(c)$ denotes the set of leaf nodes for star c.

Algorithm 3 (SolutionBacktrack)

Input: the last star r and its solution

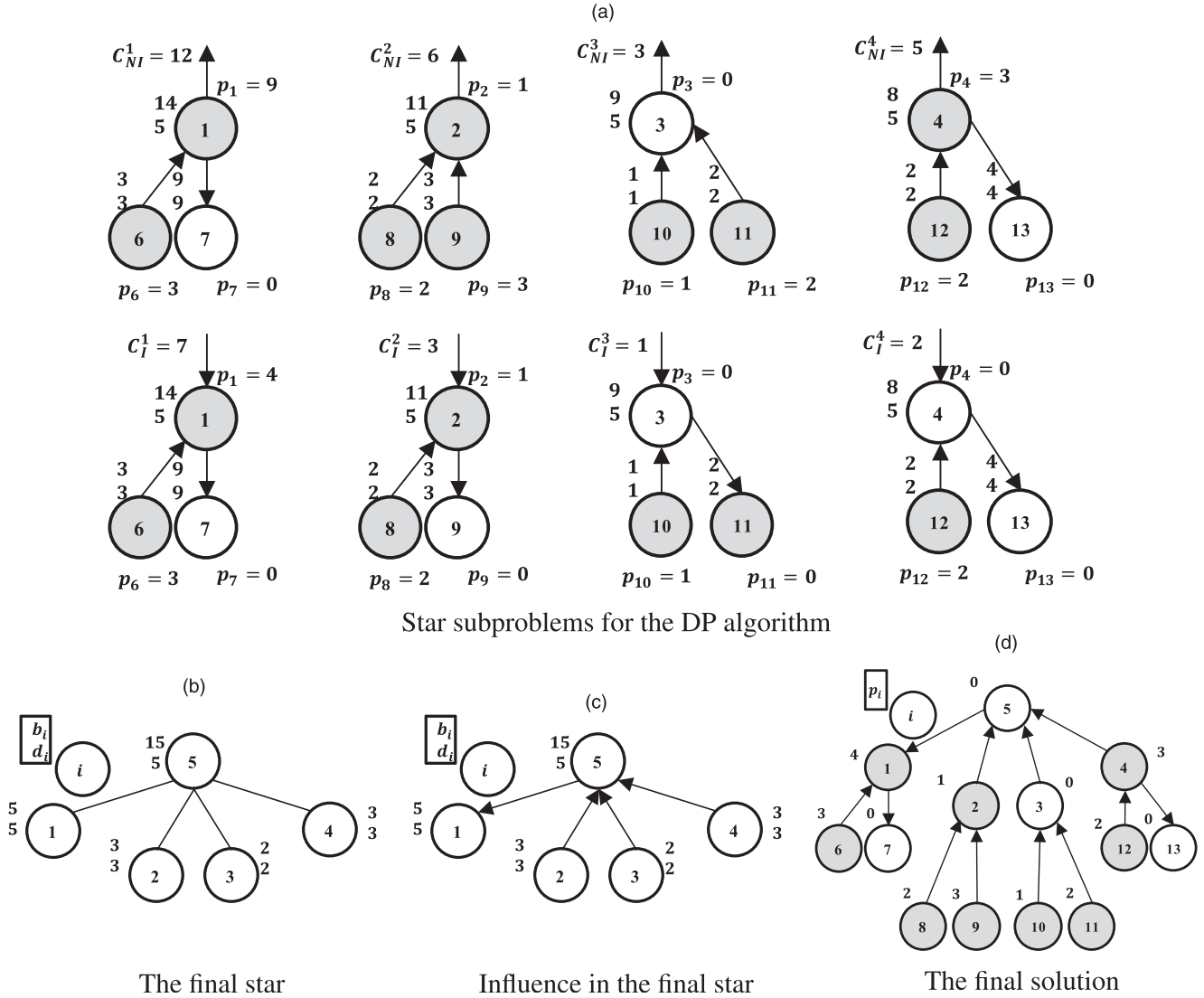
Let $\mathbf{p} \leftarrow \mathbf{p}^r_{NI}$.
$\forall l \in \{L(r) \cap X^r_{NI} \cap NL\}$ call No-Influence(l, $\mathbf{p}$, X).
$\forall l \in \{L(r) \setminus X^r_{NI} \cap NL\}$ call With-Influence(l, $\mathbf{p}$, X).
return $\mathbf{p}$, X.
function With-Influence(c, $\mathbf{p}$, X)
  $X \leftarrow (X \setminus c) \cup X^c_I$, update $p_c = 0$, and $\mathbf{p} = \mathbf{p} + \mathbf{p}^c_I$.
  $\forall l \in \{L(c) \cap X^c_I \cap NL\}$ call No-Influence(l, $\mathbf{p}$, X).
  $\forall l \in \{L(c) \setminus X^c_I \cap NL\}$ call With-Influence(l, $\mathbf{p}$, X).
  return $\mathbf{p}$, X.
end function
function No-Influence(c, $\mathbf{p}$, X)

12: $X \leftarrow (X \setminus c) \cup X^c_{NI}$, update $p_c = 0$, and $\mathbf{p} = \mathbf{p} + \mathbf{p}^c_{NI}$.
13: $\forall l \in \{L(c) \cap X^c_{NI} \cap NL\}$ call No-Influence$(l, \mathbf{p}, X)$.
14: $\forall l \in \{L(c) \setminus X^c_{NI} \cap NL\}$ call With-Influence$(l, \mathbf{p}, X)$.
15: **return** $\mathbf{p}$, X.
16: **end function**

After we obtain the solution of the last star, which has the root node as its central node, we invoke a backtracking procedure to obtain the final solution for this tree (if we are simply interested in the cost of the optimal solution, no backtracking is necessary, as global variable TC contains the cost of the optimal solution). Let r denote the root of the tree (as determined in Algorithm 1) and NL denote the set of nonleaf nodes in the tree. After solving the last star subproblem, we know (from Algorithm 2) that leaf nodes in X^r_{NI} do not receive influence from their parent (the root), but the remaining leaf nodes do. With this information, we can proceed down the tree, incorporating partial solutions at each node based on whether it receives influence from its parent or not. Algorithm 3 describes this backtracking procedure. It contains two recursive functions: with-influence for the case where a central node receives influence from its parent and no-influence for the case where a central node does not receive influence from its parent.

Figure 5 illustrates the DP algorithm for the instance shown in Figure 4(a). Figure 5(a) shows solutions for the star subproblems in the DP algorithm. The first row of Figure 5(a) displays the solutions for the no-influence case (i.e., no influence from parent node), and the second row displays the solutions for the with-influence case (i.e., influence from parent node). Figure 5(b) shows the final star at the root node (after all other stars have been compressed). From Figure 5(a), one can see that star 1 has cost 12 and 7 for the no- and with-influence solutions, respectively.

Figure 5. The DP Algorithm for the LCIP on a Tree



Thus, the threshold (and influence factor) for compressed node 1 is 5 in this final star. Similarly, the thresholds (and influence factors) are 3, 2, and 3 for compressed node 2, 3, and 4, respectively, in this final star. In Figure 5(c), the influence propagation directions are displayed for the optimal solution of the final star. This identifies which stars receive influence from their parents and which ones do not. Thus, star 1 uses the with-influence solution; and stars 2, 3, and 4 use the no-influence solution. Figure 5(d) provides this final solution, which is identical to the one found by the greedy algorithm.

Theorem 5. *The DP algorithm solves the LCIP on trees optimally in $O(|V|)$ time.*

Proof. The bottleneck is the calculation to solve each star subproblem. There are at most $|V|$ stars in a tree. For each star, we need to find the g_i cheapest children, and it takes $O(\deg(i))$ time. (Finding the g_ith order statistics can be done in $O(\deg(i))$ time by the Quickselect method in chapter 9 of Cormen et al. (2009); thus, it takes $O(\deg(i))$ time to go through the list to collect the g_i cheapest children.) For the whole tree, this sum is bounded by $O(|V|)$ (because $\sum_{i\in V}\deg(i) = 2|E| = 2|V|-2$). □

In addition to a better time complexity, another advantage of the DP algorithm is the fact that it also applies for the LCIP on trees with unequal influence factors. The DP recursion remains the same. However, as we now explain, the solution to the no-influence and with-influence cases corresponds to a mixed 0-1 knapsack problem (Marchand and Wolsey 1999). Given an LCIP on stars with unequal influence factors, let the central node c have threshold b_c and influence factor d_{cj} for all j in $L(c)$. Each leaf node j in $L(c)$ has $b_j = d_j$, given that it only has one neighbor. Then, finding the optimal solution for this star is equivalent to solving the following problem:

$$\textbf{MixedKP} \quad \text{Min} \quad p_c + \sum_{j\in L(c)} b_j x_j, \tag{6}$$

$$\text{Subject to } p_c + \sum_{j\in L(c)} d_{cj} x_j \geq b_c, \tag{7}$$

$$p_c \geq 0,\ x_j \in \{0,1\} \quad \forall j \in V, i \in L(c), \tag{8}$$

where p_c represents the incentive we give to node c. For a node j in $L(c)$, binary variable x_j decides whether it receives payment. Note that for a leaf node, it either receives full payment or no payment in an optimal solution. For the with-influence solution, we update the threshold b_c as $b_c - d_{cp}$, where d_{cp} is the influence from node c's parent. Thus, we need to solve two mixed 0-1 knapsack problems. Although the running time is no longer polynomial, a mixed 0-1 knapsack problem can be solved quite efficiently in practice (Lin et al. 2011).

3.3. Totally Unimodular Formulation

The mixed-integer programming (MIP) model in Section 1.1 for the LCIP tracks influence propagation by creating artificial time periods. In this section, we propose a different MIP formulation for the LCIP on trees (recall that we have equal influence and 100% adoption) that uses the directed influence propagation network (i.e., the direction influence travels over edges in the network). First, consider the following formulation (MIP2).

$$\textbf{MIP2} \quad \text{Min} \quad \sum_{i\in V} p_i, \tag{9}$$

$$\text{Subject to} \quad y_{ij} + y_{ji} = 1 \qquad \forall \{i,j\} \in E, \tag{10}$$

$$p_i + \sum_{j\in a(i)} d_i y_{ji} \geq b_i \qquad \forall i \in V, \tag{11}$$

$$p_i \geq 0 \qquad \forall i \in V, \tag{12}$$

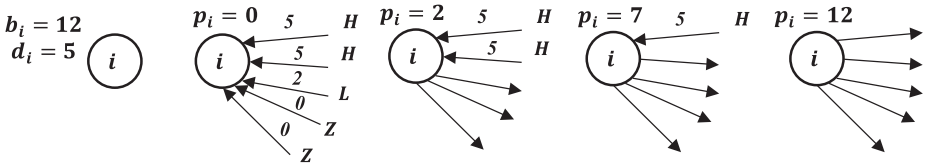
$$y_{ji} \in \{0,1\} \qquad \forall j \in V, i \in a(j). \tag{13}$$

In MIP2, p_i is a continuous variable denoting the amount of incentive paid to a node i, and y_{ij} is a binary variable that tells us whether node i influences node j (it is 1 if node i influences node j). Constraint set (10) says that for each edge $\{i,j\}$ in the network, either node i influences node j or node j influences node i. Constraint set (11) ensures that for each node i in V, the total of the incoming influence and the payment it receives is greater than or equal to its threshold. Although this model is much smaller than MIP1, its linear relaxation does not provide integral solutions on trees (nor does MIP1).

We will build upon MIP2 to derive a TU formulation for trees. Observe that if constraint set (11) always held at equality, we could replace p_i by $b_i - \sum_{j\in a(i)} d_i y_{ji}$ in the objective function and eliminate the payment variables from the model. In that case, we are left with constraint set (10), which is totally unimodular. Unfortunately, constraint set (11) does not necessarily hold at equality because a node may have g_i or greater incoming arcs in a feasible solution (i.e., it may receive influence from g_i or more neighbors). Our model will instead categorize the incoming influence to node i on an arc into three types: H with incoming influence d_i, L with incoming influence $l_i = b_i - (g_i - 1)d_i$, and Z with incoming influence 0, so that the incoming influence is exactly equal to the difference between its threshold b_i and its payment p_i.

We first consider the situation where $g_i \geq 2$ for node i and explain this categorization. Consider the example in Figure 6. Here, $b_i = 12$ and $d_i = 5$. Thus $g_i = 3$ and $l_i = 2$. Observe that there are $g_i + 1 = 4$ possible scenarios. Either the node receives no payment (i.e., $p_i = 0$), which means that it must receive incoming influence of type H on $g_i - 1 = 2$ arcs, and an incoming influence of type L on one arc. Any remaining incoming arcs are of type Z and provide an incoming influence of 0. Or, the node

Figure 6. Categorization of Incoming Influence when $g_i \geq 2$



receives a payment of $l_i + \lambda d_i$ (where $\lambda = 0, \ldots, g_i - 1$) and has exactly $g_i - 1 - \lambda$ incoming arcs of type H. Figure 6 shows these scenarios with $p_i = 0, 2, 7, 12$, respectively.

We now consider the situation where $g_i = 1$ for node i and explain this categorization. In the example in Figure 7, $b_i = d_i = 4$. Thus, $g_i = 1$ (recall without loss of generality when $g_i = 1$, $b_i = d_i$) and $l_i = d_i$. There are $g_i + 1 = 2$ possible scenarios. Either the node receives no payment, which means that it must receive an incoming influence of type L on one arc. Any remaining incoming arcs are of type Z and provide an incoming influence of 0. Or, the node receives a payment of $p_i = l_i$ and has no incoming arcs. Figure 7 shows these scenarios with $p_i = 0, 4$, respectively.

Taken together, we observe that when there is no payment to a node, there are exactly $(g_i - 1)$ arcs with the incoming influence of type H and one arc with the incoming influence of type L. When there is a payment (notice the payment set is discrete), the incoming arcs can only provide influence of type H, and there are at most $(g_i - 1)$ of them. Finally, the payment at a node is easily recovered by subtracting the sum of incoming influences from its threshold b_i.

We use these observations and develop our third-formulation BIP3 (a pure 0-1 integer program). Essentially, the binary variable y_{ji} in MIP2 is decomposed into three binary variables, x^H_{ji}, x^L_{ji}, and x^Z_{ji}, to represent the type of incoming influence. In other words, x^H_{ji}, x^L_{ji}, and x^Z_{ji} are set to 1, when node i receives incoming influence of type H, L, and Z, respectively, from node j, and is 0 otherwise.

$$\textbf{BIP3:} \quad \max \sum_{i \in V} \sum_{j \in a(i)} \sum_{k \in \{H,L,Z\}} c^k_i x^k_{ji}, \tag{14}$$

$$\text{subject to} \quad \sum_{k \in \{H,L,Z\}} (x^k_{ij} + x^k_{ji}) = 1 \qquad \forall \{i,j\} \in E, \tag{15}$$

$$\sum_{j \in a(i)} x^H_{ji} \leq g_i - 1 \qquad \forall i \in V, \tag{16}$$

$$\sum_{j \in a(i)} x^L_{ji} \leq 1 \qquad \forall i \in V, \tag{17}$$

$$x^k_{ji} \in \{0,1\} \qquad \forall i \in V, j \in a(i), k \in \{H,L,Z\}. \tag{18}$$

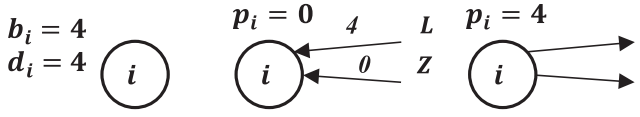
Constraint set (15) is the analog of constraint set (10). It specifies that for each edge $\{i,j\}$ in the network, either node i influences node j or node j influences node i; and the amount of influence may only be one of the three types. Constraint set (16) ensures that a node i has no more than $g_i - 1$ of its incoming arcs having type H influence. Constraint set (17) ensures that a node i has at most one incoming arc with type L influence. The objective is to maximize the influence propagation on the network. The objective coefficient $c^H_i = d_i$ provides the amount of incoming influence into node i when it receives type H influence on an arc; $c^L_i = l_i$ provides the amount of incoming influence into node i when it receives type L influence on an arc; and $c^Z_i = 0$ provides the amount of incoming influence into node i when it receives type Z influence on an arc. Because our model ensures that the total sum of incoming influences never exceeds the threshold, minimizing the sum of the payments $(\sum_{i \in V} p_i)$ is the same as minimizing the sum of the thresholds minus the incoming influences at each node $(\sum_{i \in V}(b_i - \sum_{j \in a(i)} \sum_{k \in \{H,L,Z\}} c^k_i x^k_{ji})$. However, $\sum_{i \in V} b_i$ is a constant, and so minimizing the sum of the payments is equivalent to maximizing $\sum_{j \in a(i)} \sum_{k \in \{H,L,Z\}} c^k_i x^k_{ji}$, the total incoming influence over the network.

Theorem 6. *The constraint matrix of BIP3 is totally unimodular.*

Proof. Let A denote the constraint matrix of BIP3—composed of constraint sets (15), (16), and (17)—and a_{ij} denote its elements. A is a 0–1 matrix that has at most two nonzero elements in each column. Observe that we can partition the rows of A into two subsets Q_1 containing constraint set (15) and Q_2 containing constraint sets (16) and (17). With this, columns that have two nonzero elements have one of the nonzero coefficients in Q_1 and one of the nonzero coefficients in Q_2. Thus, from corollary 2.8 in Nemhauser and Wolsey (1988), A is a TU matrix. □

Because the right-hand sides of the constraint sets are integers, the linear relaxation of BIP3 provides integral solutions to the LCIP on trees.

Figure 7. Categorization of Incoming Influence when $g_i = 1$



4. Conclusions

With the widespread use of online social networks, there is significant interest in solving problems dealing with viral marketing and influencing maximization on social networks. This paper defines and studies the NP-hard LCIP, where we are concerned with finding the least-expensive way of maximizing influencing over a social network. In contrast to previous literature, where an individual selected for targeting is paid a fixed amount, the LCIP allows those selected for targeting to be paid fractions of the fixed amount with a corresponding partial influence. This model is more closely aligned with marketing practice, where it is common to provide tailored incentives (e.g., coupons that reduce the price of a product) instead of receiving a product gratis.

We addressed the complexity of the LCIP, including several special cases. We then considered the LCIP with equal neighbor influence (recall, this is particularly relevant in a setting where privacy concerns are present in social networks) and 100% adoption. This variant of the LCIP is polynomially solvable on trees via both a greedy and a DP algorithm. We presented a TU formulation for the LCIP on trees built on the influence-propagation network that makes use of special structures about the amount of influence passing along an arc.

Using the observation that the influence-propagation network must be a directed acyclic graph (DAG), Günneç et al. (2018) embed this TU formulation for trees into a formulation on arbitrary graphs, where an additional exponentially sized set of constraints is added to ensure that the arcs selected form a DAG. They design and implement a branch-and-cut approach for the LCIP on arbitrary graphs (when neighbors exert equal influence and 100% adoption is required). In their computational study, on real-world graph instances with up to 10,000 nodes and 40,000 edges, they obtain solutions that are on average 1.95% from optimality within a 10-minute time limit.

A recent paper by Fischetti et al. (2018), published after the present paper was first submitted, cites an earlier version of the present paper that also included the branch-and-cut approach described in Günneç et al. (2018). Fischetti et al. (2018) proposed a rather novel set covering formulation that applies even when neighbors have unequal influence; the influence structure is nonlinear (e.g., diminishing influence or increasing influence from each additional neighbor); or when it is not necessary to achieve 100% adoption. Their formulation has an exponential number of variables as well as an exponential number of constraints to address these three issues. Using this formulation, they describe a branch-price-and-cut approach that dynamically generates both columns (variables) and cuts (constraints). In the setting where neighbors have equal (linear) influence and 100% adoption is required, they apply their approach to simulated graph instances with up to 100,000 nodes, with an average degree of 4, obtaining solutions ranging from 0% to 53.2% from optimality. In the more general setting (unequal influence, nonlinear influence structure, and without 100% adoption), they apply their approach to simulated graph instances with up to 100 nodes and average degree of a node up to 16. Their branch-price-and-cut approach finds optimal solutions when the average degree of a node is 4, but the quality of the solutions rapidly deteriorates when the average degree increases, reaching a maximum of 94.8% from optimality when the average degree of a node is 16.

The model we consider allows the influence propagation process to take as many steps/time periods as necessary. A natural direction is to consider the influence maximization (e.g., the LCIP and the WTSS problem) with a constraint on the time allowed for the diffusion of influence (we call these latency constraints). Raghavan and Zhang (2018a, b) begin to address this question. They discuss the LCIP and WTSS problem in the scenario where there is only one time period for the influence-propagation process. We note that a latency constraint of 1 may be relevant for networks like Twitter. Goel et al. (2015) investigated the diffusion of nearly a billion news stories, videos, pictures, and petitions on Twitter and found that the vast majority of influence cascades (over 99%) terminate within a single time period.

References

Adcock AB, Sullivan BD, Mahoney MW (2013) Tree-like structure in large social and information networks. *Proc. 13th Internat. Conf. Data Mining (ICDM)* (IEEE, Piscataway, NJ), 1–10.

Aral S, Walker D (2012) Identifying influential and susceptible members of social networks. *Science* 337(6092):337–341.

Bapna R, Umyarov A (2015) Do your online friends make you pay? A randomized field experiment on peer influence in online social networks. *Management Sci.* 61(8):1902–1920.

Bourne F (1957) Group influence in marketing and public relations. *Some Applications of Behavioral Research* (UNESCO, Basel, Switzerland), 207–225.

Brown J, Reingen P (1987) Social ties and word-of-mouth referral behavior. *J. Consumer Res.* 14(3):350–362.

Centola D (2010) The spread of behavior in an online social network experiment. *Science* 329(5996):1194–1197.

Chen N (2009) On the approximability of influence in social networks. *SIAM J. Discrete Math.* 23(3):1400–1415.

Chen W, Castillo C, Lakshmanan LV (2013) *Information and Influence Propagation in Social Networks* (Morgan & Claypool Publishers, Williston, VT).

Cordasco G, Gargano L, Rescigno AA, Vaccaro U (2015) Optimizing spread of influence in social networks via partial incentives. Scheideler C, ed. *Structural Information and Communication Complexity* (Springer, Cham, Switzerland), 119–134.

Cormen TH, Leiserson CE, Rivest RL, Stein C (2009) *Introduction to Algorithms* (MIT Press, Cambridge, MA).

Demaine ED, Hajiaghayi M, Mahini H, Malec DL, Raghavan S, Sawant A, Zadimoghadam M (2014) How to influence people

with partial incentives. *Proc. 23rd Internat. Conf. World Wide Web* (ACM, New York), 937–948.

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

Goel S, Anderson A, Hofman J, Watts DJ (2015) The structural virality of online diffusion. *Management Sci.* 62(1):180–196.

Granovetter M (1978) Threshold models of collective behavior. *Amer. J. Sociol.* 83(6):1420–1443.

Günneç D (2012) Integrating social network effects in product design. Unpublished doctoral dissertation, University of Maryland, College Park, College Park.

Günneç D, Raghavan S (2017) Integrating social network effects in the share-of-choice problem. *Decision Sci.* 48(6):1098–1131.

Günneç D, Raghavan S, Zhang R (2018) A branch-and-cut approach for the least cost influence problem on social networks. Working paper, University of Maryland, College Park, College Park.

Kempe D, Kleinberg J, Tardos É (2003) Maximizing the spread of influence through a social network. *Proc. 9th ACM SIGKDD Internat. Conf. Knowledge Discovery Data Mining* (ACM, New York), 137–146.

Lin G, Zhu W, Ali MM (2011) An exact algorithm for the 0–1 linear knapsack problem with a single continuous variable. *J. Global Optim.* 50(4):657–673.

Marchand H, Wolsey L (1999) The 0-1 knapsack problem with a single continuous variable. *Math. Programming* 85(1):15–33.

Nemhauser GL, Wolsey LA (1988) *Integer and Combinatorial Optimization* (Wiley, New York).

Raghavan S, Zhang R (2018a) Influence maximization with latency requirements on social networks. Working paper, University of Maryland, College Park, College Park.

Raghavan S, Zhang R (2018b) Rapid influence maximization on social networks: The positive influence dominating set problem. Working paper, University of Maryland, College Park, College Park.

Raghavan S, Zhang R (2018c) Weighted target set selection on trees and cycles. Working paper, University of Maryland, College Park, College Park.

Raghavan S, Zhang R (2019) A branch-and-cut approach for the weighted target set selection problem on social networks. *INFORMS J. Optim.*, ePub ahead of print July 8, https://doi.org/10.1287/ijoo.2019.0012.

Tucker C, Zhang J (2010) Growing two-sided networks by advertising the user base: A field experiment. *Marketing Sci.* 29(5):805–814.

Wu H-H, Küçükyavuz S (2018) A two-stage stochastic programming approach for influence maximization in social networks. *Comput. Optim. Appl.* 69(3):563–595.