Rapid Influence Maximization on Social Networks: The Positive Influence Dominating Set Problem

S. Raghavan
Robert H. Smith School of Business & Institute for Systems Research, University of Maryland, College Park, Maryland 20742, USA, raghavan@umd.edu.

Rui Zhang*
Leeds School of Business, University of Colorado, Boulder, Colorado 80309, USA, rui.zhang@colorado.edu.

Motivated by applications arising on social networks, we study a generalization of the celebrated dominating set problem called the Positive Influence Dominating Set (PIDS). Given a graph \( G = (V, E) \), each node \( i \in V \) has a weight \( b_i \), and a threshold requirement \( g_i \). We seek a minimum weight subset \( T \) of \( V \), so that every node \( i \in V \setminus T \) is adjacent to at least \( g_i \) members of \( T \). When \( g_i = 1 \) for all nodes, we obtain the weighted dominating set problem. First, we propose a strong and compact extended formulation for the PIDS problem. We then project the extended formulation onto the space of the natural node-selection variables to obtain an equivalent formulation with an exponential number of valid inequalities. Restricting our attention to trees, we show that the extended formulation is the strongest possible formulation, and its projection (onto the space of the node variables) gives a complete description of the PIDS polytope on trees. We derive the necessary and sufficient facet-defining conditions for the valid inequalities in the projection and discuss their polynomial time separation. We embed this (exponential size) formulation in a branch-and-cut framework and conduct computational experiments using real-world graph instances, with up to approximately 2.5 million nodes and 8 million edges. On a test-bed of 100 real-world graph instances, our approach finds solutions that are on average 0.2% from optimality and solves 51 out of the 100 instances to optimality.

Key words: Rapid Influence Maximization, Social Networks, Dominating Set, Facets, Integer Programming, Strong Formulation

1. Introduction

A recent report (Shearer and Matsa 2018) shows that in the United States, people recognize online social networks as one of the most effective ways for disseminating information, and two-thirds of the population use their online social networks as one of the channels for receiving information and news. Not surprisingly, people’s decisions are affected by the information they receive through social media. As pointed out in Valente (2012), online social media provide unique opportunities in comparison with other communication channels to monitor, respond to, amplify and intervene in people’s behavior. In a 61-million-person experiment on Facebook, Bond et al. (2012) showed that direct targeting can change people’s behavior; they also report that this influence can be transmitted to their friends. Matz et al. (2017) report on the results of three field experiments that

* This paper is partly based on the third chapter of the author’s doctoral dissertation (Zhang 2016).
reached over 3.5 million individuals via advertising tailored to individuals’ psychological characteristics. They found that tailored advertising that matched with people’ psychological characteristics resulted in up to 40% more clicks and up to 50% more purchases than their counterparts who didn’t receive tailored advertising. Given this large shift of information dissemination to the online forum and the effectiveness of tailored targeting strategies in the online environment, the social media influencer marketing segment has taken off and become a thriving industry. This segment is estimated to be worth somewhere between $5 billion to $16 billion in 2020 (Schmidt 2019), and may be worth up to $15 billion by 2022 (Schomer 2020).

Chen (2009) initiated the study of an influence maximization problem on social networks called the Target Set Selection (TSS) problem. In the TSS problem, given a connected undirected graph, each node has associated with it a threshold value $g_i$, which takes values between 1 and the degree of the node, denoted by $\text{deg}(i)$. All nodes are inactive initially. A selected subset of nodes, the target set, are activated (i.e., switched to an active state). Next, the states of the 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 cardinality target set, while ensuring that all nodes are active by the end of this information diffusion process. Raghavan and Zhang (2019) discuss the weighted TSS (WTSS) problem where each node $i \in V$ has a weight $b_i$ to model the fact that different nodes require differing levels of effort to become active in practice.

An important issue that is not considered in these previous works is the difference between the “direct influence” received from a node that has been selected for targeting and the “indirect influence” received from a node that has not been selected for targeting. In practice, there is a significant difference regarding the magnitude of effects of direct and indirect influence. For instance, Goel et al. (2015) investigated the diffusion of nearly a billion news stories, videos, pictures, and petitions on the microblogging service Twitter and observed that the vast majority of cascades (over 99%) terminate within a single time period (indicating that almost all of the diffusion consists of direct influence). Zhang et al. (2018) considered the diffusion of technology adoption in the context of caller ringback tones (CRBT) on a data set of 200 million calls between 1.4 million users and found that the measured effects of direct and indirect influence are markedly different. Specifically, they found that the adoption of CRBT is consistently predicted by direct peer influence, indicating that the magnitude of direct influence is statistically much larger than that of indirect influence.

When we consider the WTSS problem and restrict our attention to direct influence we obtain the Positive Influence Dominating Set (PIDS) problem. Formally, we define the PIDS problem as follows: Given an undirected graph $G = (V,E)$, each node $i$ in $V$ has a weight $b_i$ and a threshold requirement $g_i$ taking values between 1 and its degree $\text{deg}(i)$; we seek a subset $T$ of $V$ such that
every node $i$ not in $T$ is adjacent to at least $g_i$ members of $T$, while minimizing the total weight of the nodes in $T$ (denoted by $\sum_{i \in T} b_i$). Notice that when $g_i = 1$ for all nodes $i \in V$, we obtain the classic dominating set problem that has applications in many different areas, including social networks (Haynes et al. 1998a,b).

Direct influence is seen in medical technology diffusion (Coleman et al. 1966) and also in settings where behavioral change is desired for medical reasons (Greaves et al. 2010) within a target population. Morse and Schulze (2013) and Walther et al. (2014) discuss the importance of peer networks for mental health counseling on college campuses. Indeed, Wang et al. (2009) described an application of the PIDS problem associated with alleviating binge drinking on college campuses and fraternities. In this setting, peer pressure plays a large role in pressuring students to binge drink, and a student is likely to become a binge drinker if more than 50% of his/her friends are binge drinkers (Walters and Neighbors 2005, Larimer and Cronce 2002). The idea is to introduce an intervention program (which can be costly) to convert binge drinking students to abstainers in such a way that the majority of each student’s friends are no longer binge drinkers. With this example, Wang et al. (2009) introduced the minimum cardinality version of the PIDS, where the goal is to select the smallest PIDS for the intervention so that after the intervention program, the entire network is influenced (i.e., we want to find a minimal subset $T$ of $V$ such that every node not in $T$ has at least half of its neighbors in $T$). A similar scenario occurs in political campaigns (the paper by Stewart et al. 2019, provides an interesting discussion of information gerrymandering in political campaigns and the value of strategically targeting a voter pool after accounting for their social network interactions), where the goal is to strategically identify a set of individuals to influence in a specific group so that a particular issue is adopted by this group.

Our definition of the PIDS problem is slightly more general than that previously studied in the literature in two ways. First, we consider the weighted version. Second, a node can require any positive number of neighbors to be in $T$ (as opposed to a fixed number or a fixed proportion of neighbors that is the same for all nodes in the graph). This reflects the situation where different people require different amounts of peer influence to adopt a behavior in practice. One can also view the PIDS problem as a rapid influence maximization problem. One where all influence propagation must take place in one step (or time period) to completely influence the network, compared to the WTSS problem where long chains of influence propagation are permitted, and no limits are placed on the number of time periods taken to completely influence the network.

1.1. Related Literature

Optimization problems in social network analytics have been a vital part of the operations research community. There have been several papers focused on optimization for classic social network
structures: for example, the clique problem (Verma et al. 2015, Walteros and Buchanan 2019), the $k$-plex problem (Balasundaram et al. 2011), and the 2-club problem (Pajouh et al. 2016). However, such structures do not explicitly model differences in user behavior or the influence propagation process. More recently, there is another stream of mathematical programming research that explicitly accounts for differences in the individual behavior of the nodes and the influence propagation process in the network. Roughly, in the variants of the problems described in these papers (Fischetti et al. 2018, Wu and Kucukyavuz 2018, Li et al. 2019, Gunnecci et al. 2020, Raghavan and Zhang 2019), the goal is to find a seed set of nodes to target in order to either maximize the number of influenced nodes in the network or to influence the entire network.

The PIDS problem has typically been considered in the literature with the assumption that a node $i$ in the graph not selected in the PIDS needs at least half its neighbors in the PIDS. Zou et al. (2009) first considered the problem under the name “Fast Information Propagation Problem”. They proved it to be NP-hard. Wang et al. (2009) coined the name Positive Influence Dominating Set problem. They also presented and tested an iterative greedy selection algorithm for the PIDS with a real-world online social network data set. Zhu et al. (2010) further showed the PIDS problem to be APX-hard and described a greedy approximation algorithm with a performance ratio $O(\ln \delta)$, where $\delta$ is the maximum degree in the given graph. Several greedy constructive methods have been proposed for the PIDS problem by Wang et al. (2011) and Dhawan and Rink (2015). Khomami et al. (2018) proposed a learning automation-based meta-heuristic algorithm for the PIDS problem. Furthermore, they considered a well-known budgeted version of the TSS problem introduced by Kempe et al. (2015) (here, the goal is to choose a seed set of nodes to maximize the number of activated nodes at the conclusion of the influence propagation process) and empirically showed that restricting the choice of seed nodes to be a subset of a PIDS provides better results than six existing well-known algorithms for the problem. Lin et al. (2018) presented an integer linear programming based memetic algorithm.

Dinh et al. (2014) considered a slightly more general version of the PIDS problem, where a node $i$ in the graph not selected in the PIDS needs at least $\lceil \rho \deg(i) \rceil$ of its neighbors in the PIDS, where $0 < \rho < 1$ (note that the value of $\rho$ is identical for all nodes). They showed that the PIDS problem is hard to approximate within $\ln \delta - O(\ln \ln \delta)$. In power-law graphs, they showed that the greedy method targeting the highest degree node has a constant factor approximation ratio. They also presented an algorithm for trees with a time complexity of $O(|V|)$.

The dominating set problem has a long history and a tremendous amount of literature (see Haynes et al. 1998a,b). However, there is limited work from the polyhedral perspective. Saxena (2004) presented the dominating set polytope for trees and cycles. His formulation used only variables associated with nodes in the graph. Baiou and Barahona (2014) showed an extended
formulation via the concept of the facility location problem (it used both node and arc variables) and proved that the projection of this formulation onto the node-selection space describes the polytope for cactus graphs. In a cactus graph, each edge is contained in at most one cycle. Thus, both trees and cycles are examples of cactus graphs.

Overall, all previous work on the PIDS problem has focused on its approximability and heuristics. Further, it generally requires $\rho = 0.5$ or the value of $\rho$ to be identical for all nodes; moreover the weighted version does not seem to have gotten any attention. Our research is motivated by the desire to better understand this generalization of the dominating set problem, which has important and natural applications in social network analytics; furthermore, our research seeks to develop practical computational approaches based upon a better understanding of the underlying polytopes.

1.2. Our Contributions and Organization of the Rest of this Paper

In Section 2, we first discuss formulations that are natural extensions of formulations for the dominating set problem. Then, we propose a stronger and compact extended formulation for the PIDS problem. One component in developing the proposed extended formulation is an edge-splitting idea, that has also been applied to obtain a stronger formulation for the WTSS problem (see Raghavan and Zhang 2019, 2021). We note that although the edge-splitting idea is similar, its manifestation and the resulting formulations are quite different from each other, leading to distinct technical proofs and computational results. We also show (in Section EC.2) that the extended formulation is the strongest possible formulation for the PIDS problem on trees. As a bonus, we present a linear time dynamic programming algorithm for the PIDS problem on trees (in Section EC.1). While the extended formulation is strong, it needs artificial variables defined on the edge space. Thus, in Section 3, we project the extended formulation projected onto the natural node-selection variable space. Although the constraints based on the projection are exponentially many, we present a polynomial time separation algorithm. In Section 4, we focus on deriving the necessary and sufficient facet-defining conditions for the set of projected valid inequalities. Section 5 presents our computational experience on real-world graphs. We are able to obtain high-quality solutions for real-world graph instances, with up to approximately 2.5 million nodes and 8 million edges within a one-hour time limit. In fact, on a test-bed of 100 real-world graph instances, our approach finds solutions that are on average 0.2% from optimality and solves 51 out of the 100 instances to optimality. Finally, Section 6 provides some concluding remarks.

2. Formulations for the PIDS Problem

In this section, we discuss three formulations for the PIDS problem. The first two are straightforward extensions of known formulations for the dominating set problem. However, as we will see later in our computational work, these two formulations are weak. In other words, the gap between
the optimal objective values of their LP relaxations and that of the optimal integer solution is large. Next, using an edge-splitting idea, we develop a stronger compact extended formulation for the PIDS problem.

A combinatorial optimization problem can be formulated as an Integer Programming (IP) in multiple ways. The standard way (see Conforti et al. 2014, Nemhauser and Wolsey 1988) to compare different IP formulations for the same problem is to solve their linear programming (LP) relaxations (as this has implications for the computational tractability of a formulation). Then, these IP formulations are evaluated by the optimal objective values of their LP relaxations. Given two different IP formulations, $A$ and $B$, of a given minimization problem, let $z_{LP}^A$ and $z_{LP}^B$ be the optimal objective values of their LP relaxations, respectively. We say that formulation $A$ is stronger (or better) than formulation $B$ if $z_{LP}^A > z_{LP}^B$. Typically, a stronger formulation is more computationally efficient (Barnhart et al. 1993).

The first formulation (BIP1) (similar to Saxena 2004, for the dominating set problem) has a binary variable $x_i$ for each node $i$ in the graph, where $x_i = 1$ if node $i$ is in the PIDS, and is 0 otherwise. Let $n(i)$ denote the set of node $i$'s neighbors.

$$\text{(BIP1) \quad \text{Minimize} \quad \sum_{i \in V} b_i x_i} \quad (1)$$

Subject to:

$$\sum_{j \in n(i)} x_j + g_i x_i \geq g_i \quad \forall i \in V \quad (2)$$

$$x_i \in \{0, 1\} \quad \forall i \in V \quad (3)$$

BIP1 also encompasses the formulation proposed by Lin et al. (2018) by including weights ($b$) in the objective function. The objective function (1) is to minimize the total cost of the PIDS. Constraint (2) states that either node $i$ is selected in the PIDS or it has at least $g_i$ neighbors in the PIDS.

The second formulation (BIP2) (similar to Baïou and Barahona 2014, for the dominating set problem) creates additional variables $y_{ij}$ and $y_{ji}$ for each edge $\{i, j\}$ in the graph. If node $i$ in the PIDS influences its neighbor $j$, $y_{ij} = 1$. Otherwise, it is 0.

$$\text{(BIP2) \quad \text{Minimize} \quad \sum_{i \in V} b_i x_i} \quad (4)$$

Subject to:

$$\sum_{j \in n(i)} y_{ji} + g_i x_i \geq g_i \quad \forall i \in V \quad (5)$$

$$x_i - y_{ij} \geq 0 \quad \forall i \in V, j \in n(i) \quad (6)$$

$$x_i, y_{ij} \in \{0, 1\} \quad \forall i \in V, j \in n(i) \quad (7)$$

The objective function (4) is to minimize the total cost of the PIDS. Constraint (5) says that for a node $i$ in $V$, either it is selected or it has at least $g_i$ neighbors influencing it. Constraint (6) says that node $i$ can influence its neighbors only if it is selected in the PIDS.
When $g_i$ and $b_i$ are 1 for all $i$ in $V$, both BIP1 and BIP2 are formulations for the dominating set problem. Although Saxena (2004) and Baïou and Barahona (2014) show that BIP1 and BIP2 (respectively) are integral for the dominating set problem on trees, they are no longer integral for the PIDS problem on trees, as we demonstrate by the instance in Figure 1.

In Figure 1(a), we have a social network with five nodes. For each node $i$, its weight and threshold values ($b_i$ and $g_i$) are listed beside it. For node 1, $b_1$ is 4 and $g_1$ is 3. By relaxing the binary variables in BIP1 and BIP2, we have their LP relaxations, which are referred to as LP1 and LP2, respectively. If LP1 and LP2 are used to solve the instance in Figure 1(a), both of them return a fractional optimal solution, $x_1 = \frac{1}{3}$, $x_2 = 0$, $x_3 = 1$, $x_4 = 1$, and $x_5 = 1$, with an objective value of $4\frac{1}{3}$. However, given that this problem can be solved in polynomial time on trees (as shown in Section EC.1), it would be ideal to find a perfect integer programming formulation so that an integral optimal solution could be obtained by solving its LP relaxation. More importantly, such a perfect formulation on trees may yield a stronger formulation for the PIDS problem on arbitrary graphs. Next, we present a stronger and compact extended formulation, which is indeed the strongest possible formulation for the PIDS problem on trees.

To obtain our extended formulation, we first create a transformed graph. From the input graph $G$, we create a new graph $G_t$ by subdividing each edge. For each edge $\{i, j\} \in E$, insert a dummy node $d$. Let $D$ denote the set of dummy nodes. Since the dummy nodes have effectively split each edge into two, we replace each of the original edges $\{i, j\} \in E$ by two edges $\{i, d\}$ and $\{d, j\}$ in the new graph $G_t$. The procedure is shown in Figure 2(a) and Figure 2(b). Notice that for a transformed edge $\{i, d, j\}$, with a slight abuse of notation, we may only specify two out of the three indexes because the third one can be inferred from the two specified indexes. Let $E_t$ denote the set of edges
in $G_t$ ($G_t = (V \cup D, E_t)$). Figure 2(c) shows the transformed graph of the example in Figure 1(a) based on this procedure. Dummy nodes are represented by rectangles.

The idea behind the strengthened formulation is as follows. For any node selected in the PIDS, we require that they influence their neighbor (in this case, a dummy node). Next, we allow (but do not require) this influence to propagate further onto the other neighbor of the dummy node. Finally, we require that the influence propagates only in one direction on an edge in the graph. We will see that the resulting formulation is valid and produces a stronger formulation. As before, for each node $i \in V$ (notice that dummy nodes are not included), we define a binary decision variable $x_i$ that is 1 if node $i$ is selected in the PIDS, and 0 otherwise (these are the node-selection variables). For each edge $\{i, d\} \in E_t$, where $i \in V$ and $d \in D$ (notice that $G_t$ is bipartite, and $E_t$ only contains edges between the nodes in $V$ and $D$), define two binary arc variables $y_{id}$ and $y_{di}$. They represent the direction of influence. If node $d$ sends influence to node $i$, $y_{di}$ is 1, and 0 otherwise. For a node $i \in V \cup D$, let $a(i)$ denote the set of node $i$’s neighbors in the transformed graph $G_t$. Recall that we use $n(i)$ to denote the set of node $i$’s neighbors in the original graph $G$. We now write the stronger and compact extended formulation for the PIDS problem and refer to it as BIP3:

\[\text{Minimize} \quad \sum_{i \in V} b_ix_i \tag{8}\]

Subject to:

\[x_i \geq y_{dj} \quad \forall i \in V, j \in n(i) \tag{9}\]

\[x_i \leq y_{id} \quad \forall i \in V, d \in a(i) \tag{10}\]

\[y_{id} + y_{di} = 1 \quad \forall \{i, d\} \in E_t \tag{11}\]

\[\sum_{d \in a(i)} y_{di} \geq g_i \quad \forall i \in V \tag{12}\]

\[x_i \in \{0, 1\} \quad \forall i \in V \tag{13}\]

\[y_{id}, y_{di} \in \{0, 1\} \quad \forall \{i, d\} \in E_t \tag{14}\]

The objective function (8) is to minimize the total cost of the PIDS. The first constraint (9) says that if node $i$ is selected, then node $d$, which is adjacent to node $i$, can send influence to node $j$ for any node $j$ in $n(i)$ (i.e., node $i$’s neighbors in the original graph). Constraint (10) requires that when a node $i$ is selected, it sends influence to all of its neighbors. Constraint (11) says that an edge $\{i, d\}$ must be directed in one of two directions. Constraint (12) ensures that for a node $i$ in $V$, either it is selected or it has at least $g_i$ incoming arcs.

**Proposition 1.** BIP3 is a valid formulation for the PIDS problem.

**Proof.** First, given any feasible solution to the PIDS problem, for any node $i$ in $V$ that is selected in the PIDS, $x_i = 1$ and 0 otherwise. The cost of this solution is correctly captured by the objective function (8). With this choice of $x$ variables, we will show that the values of the $y$
variables can be selected to obtain a feasible solution to BIP3. Initially, we set all \( y \) variables to 0. Consider a node \( i \) that is selected in the PIDS. We set \( y_{id} = 1 \) for all \( d \) in \( a(i) \). Then, we consider all \( j \in n(i) \) and set \( y_{dj} = 1 \) if node \( j \) has not been selected in the PIDS. This is repeated for every node selected in the PIDS. Thus, constraints (9) and (10) are satisfied. Notice that in the procedure described, at most one of \( y_{id} \) and \( y_{di} \) has a non-zero value for an edge \( \{i, d\} \) in \( E_t \). If an edge \( \{i, d\} \) in \( E_t \) is not directed yet, set \( y_{id} = 1 \) and \( y_{di} = 0 \). Thus, constraint (11) is respected. Furthermore, constraint (12) is trivially satisfied for any node \( i \) in the PIDS and is satisfied for any node \( i \) not in the PIDS because we have a feasible solution; thus a node \( i \) not in the PIDS has at least \( g_i \) nodes from its neighbors, \( n(i) \), in the PIDS that will set their associated \( y_{di} \) to 1 in the procedure described. Lastly, only zero-one values are assigned to all variables. Therefore, constraints (13) and (14) are respected. We obtain a feasible solution to BIP3.

Second, because of constraints (9) and (12), the \( x \) variable part of any feasible solution to BIP3 satisfies the definition of the PIDS problem. □

Next, in Theorem 1, we show that in terms of the strength of LP relaxation, BIP1 and BIP2 are equivalent; at the same time BIP3 is a stronger formulation than BIP1 and BIP2. Let LP3 be the LP relaxations of BIP3. Then, let \( z_{LP1} \), \( z_{LP2} \) and \( z_{LP3} \) denote the optimal objective values of LP1, LP2 and LP3, respectively.

**Theorem 1.** \( z_{LP1} = z_{LP2} \leq z_{LP3} \).

**Proof.** First, we show that LP1 and LP2 are equivalent. Given any solution of LP1, denoted by \( x^* \), we set \( y_{ij}^* = x_{i}^* \) for each \( i \) in \( V \) and \( j \) in \( n(i) \). Thus, we have \( \sum_{j \in n(i)} y_{ij}^* + g_i x_{i}^* \geq g_i \) because \( x^* \) satisfies constraint (2) \( \sum_{j \in n(i)} x_{j}^* + g_i x_{i}^* \geq g_i \). Thus, we have a feasible solution for LP2. Next, given any feasible solution of LP2, denoted by \( (x^*, y^*) \), the \( x^* \) part is a feasible solution for LP1 as a result of constraints (5) and (6).

Second, we show that LP3 is at least as strong as LP1. Given any feasible solution of LP3, denoted by \( (x^*, y^*) \), the \( x^* \) part is a feasible solution for LP1 because of constraints (9) and (12). However, not all feasible solutions of LP1 can be converted into a feasible solution for LP3. Figure 1 provides a counter example. Recall that \( x_1 = \frac{1}{3}, x_2 = 0, x_3 = 1, x_4 = 1 \) and \( x_5 = 1 \) is a feasible and optimal solution for LP1 with an objective value of 4\( \frac{1}{3} \). However, this set of \( x \) values is not feasible for LP3 (with any set of \( y \) values). Solving LP3 for the instance after transformation (as shown in Figure 2(c)), we get an integral solution \( x_1 = 1, x_2 = 0, x_3 = 0, x_4 = 0, x_5 = 1, y_{1a} = 1, y_{ab} = 1, y_{1c} = 1, y_{c2} = 1, y_{a3} = 1, y_{a4} = 1, y_{5d} = 1 \) and the remaining \( y \) variables are zeros with an objective value of 5. □

BIP3 is actually the strongest possible formulation for the PIDS problem on trees. This means that we can obtain an optimal integral solution by solving LP3 instead of BIP3. Theorem 2 proves
this result. It constructs an integral primal feasible solution to LP3 using the dynamic programming algorithm in Section EC.1, a dual feasible solution for the dual problem to LP3 and shows that these solutions satisfy the complementary slackness (CS) conditions. Let $\text{conv}(X)$ denote the convex hull of the feasible PIDS vectors $x$, and let EPIDS denote the feasible region of LP3.

**Theorem 2.** For trees, LP3 has an optimal solution with $x$ variables taking binary values and $\text{Proj}_x(\text{EPIDS}) = \text{conv}(X)$.

**Proof.** Included in Section EC.2.

### 3. A Strong Formulation on the Node Selection Variables

In this section, we follow a method, proposed by Balas and Pulleyblank (1983), which is based on a theorem of the alternatives to project the extended formulation BIP3 onto the natural node-selection (i.e., $x$ variable) space by projecting out all arc (i.e., $y$) variables. As will be evident in our computational experiments, this formulation has great advantages in computational efficiency (in terms of scaling up), compared to BIP3.

Because $y_{id} + y_{di} = 1$ in LP3, we first project out all $y_{id}$ variables, setting them to $1 - y_{di}$ and obtain the following formulation whose feasible region is denoted as $P_{\text{extended}}$.

Minimize $$\sum_{i \in V} b_i x_i$$

Subject to:

1. $$-y_{di} + x_j \geq 0 \quad \forall j \in V, i \in n(j)$$
2. $$-y_{di} - x_i \geq -1 \quad \forall i \in V, d \in a(i)$$
3. $$\sum_{d \in a(i)} y_{di} + g_i x_i \geq g_i \quad \forall i \in V$$
4. $$0 \leq x_i \leq 1, y_{di} \geq 0 \quad \forall i \in V, d \in a(i)$$

We define a projection cone $W$, described by $(w, u, v)$, which satisfies the following linear inequalities:

$$w_i - u_{id} - v_{id} \leq 0 \quad \forall i \in V, d \in a(i)$$

$$w_i \geq 0, u_{id} \geq 0, v_{id} \geq 0 \quad \forall i \in V, d \in a(i)$$

Here, $u_{id}$, $v_{id}$ and $w_i$ are dual multipliers corresponding to constraints (16), (17) and (18), respectively. If $P_{\text{extended}}$ is written in matrix notation as $\{(x, y) : Ax + Gy \geq b, (x, y) \geq 0\}$, based on Balas and Pulleyblank (1983), then any feasible vector $(w, u, v)$ to $W$ defines a valid inequality: $(w, u, v)^T Ax \geq (w, u, v)^T b$ to the projection of $P_{\text{extended}}$ (in the space of the node-selection ($x$) variables). Furthermore, the projection of $P_{\text{extended}}$ is defined by the valid inequalities projected by the extreme rays of $W$. In Theorem 3, we identify the extreme rays of $W$. First, let us provide
some additional definitions. Recall that a polyhedral cone $C$ is the intersection of a finite number of half-spaces through the origin, and a pointed cone is one in which the origin is an extreme point. A ray of a cone $C$ is the set $R(r)$ of all non-negative multipliers of some $r \in C$, called the direction (vector) of $R(r)$. A vector $r \in C$ is extreme, if for any $r^1, r^2 \in C$, $r = \frac{1}{2}(r^1 + r^2)$ implies $r^1, r^2 \in R(r)$. A ray $R(r)$ is extreme if its direction vector $r$ is extreme.

**Theorem 3.** The vector $r = (w, u, v) \in W$ is extreme if and only if there exists a positive $\alpha$ such that one of the following three cases holds:

*Case 1.* $u_{id} = \alpha$ for one $\{i, d\} \in E_t$. All other $w, u, v$ are 0.

*Case 2.* $v_{id} = \alpha$ for one $\{i, d\} \in E_t$. All other $w, u, v$ are 0.

*Case 3.* $w_i = \alpha$ for one $i \in V$. Then for $d \in a(i)$, either $u_{id} = \alpha$ or $v_{id} = \alpha$. All other $w, u, v$ are 0.

**Proof.** Sufficiency. Let $r \in W$ be of the form Case 1 and assume that $r = \frac{1}{2}(r^1 + r^2)$ for some $r^1, r^2 \in W$. Then, except for $u_{id}^1$ and $u_{id}^2$, all other components are 0. Then, $r^1, r^2$ are in $R(r)$. Thus, $r$ is extreme.

Case 2 is similar to Case 1.

For Case 3, let $r \in W$ be of the form Case 3 and assume that $r = \frac{1}{2}(r^1 + r^2)$ for some $r^1, r^2 \in W$. Thus, for any component of $r$ with the value 0, its corresponding components in $r^1$ and $r^2$ are also 0. Given $i$ and $d$, let $p_{id}^l$, $l = 1, 2$, represent the positive element between $u_{id}^l$ and $v_{id}^l$, $l = 1, 2$ (since only one of the two can be positive in the three cases). Then, we have $w_i^l + w_i^2 = 2\alpha$ and $p_{id}^1 + p_{id}^2 = 2\alpha$, for all $d \in a(i)$. For a pair $d_1$ and $d_2$, we have $p_{id_1}^l > p_{id_2}^l$ if and only if $p_{id_1}^2 < p_{id_2}^1$. However, constraint (20) stipulates that $w_i^1 \leq \min\{p_{id_1}^1, p_{id_1}^2\}$, $l = 1, 2$. Hence, $p_{id_1}^1 = p_{id_2}^1 = \alpha_1$, $l = 1, 2$, for all $d_1, d_2 \in a(i)$. Otherwise, either constraint (20) would be violated or we would have $w_i^1 + w_i^2 < 2\alpha$. Therefore, $r^1, r^2$ are in $R(r)$. Thus, $r$ is extreme.

Necessity. Let $r$ be an extreme vector of $W$. Let $S^w = \{i \in V : w_i > 0\}$, $S^u = \{\{i, d\} \in E_t : u_{id} > 0\}$ and $S^v = \{\{i, d\} \in E_t : v_{id} > 0\}$ based on this $r$. In the following proof, to prove that a given ray $r$ is not an extreme one, we construct two feasible rays, $r^1$ and $r^2$, which are different in at least one component. After constructing $r^1, r^2$ is set as $2r - r^1$. Then, $r = \frac{1}{2}(r^1 + r^2)$ by design. We will illustrate the different steps in the necessity proof with the instance in Figure 2(c).
First, we consider the situation, where $S^u = \emptyset$. Suppose that $|S^u| + |S^v| > 1$; let $r^1$ contain all but one of the positive components in $r$ with their values doubled. Thus, if $|S^u| + |S^v| > 1$, $r$ is not extreme, contrary to the assumption. We conclude that if $S^u = \emptyset$, then $|S^u| + |S^v| = 1$ and thus, $r$ is either in the form of Case 1 or Case 2. Figure 3 illustrates this situation. The bold line represents the positive $u$ and $v$ components in a vector $r$, and the positive components of $r$ are displayed below the pictures.

Now consider the case when $S^u \neq \emptyset$. Suppose that $|S^u| > 1$; without loss of generality, let $i \in S^u$. Then, $r^1$ has the value $w_i^1 = 2w_i$ and $u_{id}^1 = 2u_{id}$, $v_{id}^1 = 2v_{id}$ for all $d \in a(i)$ and 0s for the other components. Thus, when $|S^u| > 1$, $r$ is not extreme. Figure 4 illustrates this situation. The shaded nodes represent the positive $w$ components in a vector $r$. Suppose that $|S^u| = 1$ and $i \in S^u$; define $S_j = \{(j,d) \in E_i : p_{jd} > 0 \& j \in V \setminus \{i\}\}$. Thus, $S_j$ contains edges with positive $u$ or $v$ components that are not adjacent to node $i$. When $S_j \neq \emptyset$, let $r^1$ have $w_i^1 = 2w_i$ and $u_{id}^1 = 2u_{id}$, $v_{id}^1 = 2v_{id}$ for all $d \in a(i)$ and 0s in the other components. Then, $r^2$ contains the positive components in $S_j$, while $r^1$ does not. Thus, when $|S^u| = 1$ and $S_j \neq \emptyset$, $r$ is not extreme. Figure 5 illustrates this situation with $S_j = \\{(1, a)\}\$.

Suppose that $|S^u| = 1$ and $i \in S^u$; define $S_1 = \{(i,d) \in E_i : u_{id} > 0 \oplus v_{id} > 0\}$, where only one of the $u$ and $v$ variables associated with an edge $\{i, d\}$ is positive and $S_2 = \{(i,d) \in E_i : u_{id} > 0 \& v_{id} > 0\}$, where both $u$ and $v$ variables associated with an edge $\{i, d\}$ are positive. Suppose that $S_2 \neq \emptyset$; then, we define $\theta^1 = 2 \min\{w_i, u_{id}, v_{id} : \{i,d\} \in S_2\}$ and make $r^1$ have $w_i^1 = \theta^1$. For $\{i,d\} \in S_2$, we have $u_{id}^1 = \theta^1$. Also, for $\{i,d\} \in S_1$, if $u_{id} > 0$, we have $u_{id}^1 = \theta^1$. Otherwise, we have $u_{id}^1 = \theta^1$. The remaining components are 0. Then, $r^1$ does not have any edges that have both $u$ and $v$ variables.

![Figure 4](image-url) When $|S^w| > 1$, $r$ is not extreme.

![Figure 5](image-url) When $|S^w| = 1$ and $S_j \neq \emptyset$ (some $u$'s and $v$'s not adjacent to $S$ have a positive value), $r$ is not extreme.
taking positive values but \( \mathbf{r}^2 \) does. Thus, when \(|S^w| = 1\) and \(S_2 \neq \emptyset\), \( \mathbf{r} \) is not extreme. Therefore, we must have \(|S_1| = \text{deg}(i)\). Otherwise, constraint (20) would not be respected. Figure 6 illustrates this situation. Here, \( S_2 = \{(3, a)\} \) and \( \alpha_1 \) is less than \( \alpha_2 \) and \( \alpha_3 \).

Suppose that \(|S^w| = 1\) and \(|S_1| = \text{deg}(i)\); let \( w_i = \alpha \) and define \( S^+ = \{\{i, d\} : p_{id} > \alpha\}\). When \( S^+ \neq \emptyset\), without loss of generality, let \( \{i, d\} \in S^+ \) and \( u_{id} > 0\); we can make \( \mathbf{r}^1 \) have \( u_{id} = 2(u_{id} - \alpha) \) and 0s in the other components. Then, \( \mathbf{r}^1 \) has only one positive component that is not \( w_i \). Thus, when \(|S^w| = 1\), \(|S_1| = \text{deg}(i)\) and \( S^+ \neq \emptyset\), \( \mathbf{r} \) is not extreme. Consequently, we must have \( S^+ = \emptyset\). Figure 7 illustrates this situation. Here, \( S^+ = \{(3, a)\} \) and \( \beta \) is a positive value.

Therefore, if \( S^w \neq \emptyset\), then \(|S^w| = 1\), \(|S_1| = \text{deg}(i)\) and \( S_j = S_2 = S^+ = \emptyset\). Thus, \( \mathbf{r} \) is in Case 3. \( \Box \)

Applying Theorem 2 in Balas and Pulleyblank (1983), Case 1 and Case 2 extreme directions give the trivial constraints: \( 0 \leq x_i \leq 1 \) for all \( i \in V \). Case 3 extreme directions generate the following valid inequality in the original graph \( G \):

\[
(g_i - q)x_i + \sum_{j \in S} x_j \geq g_i - q \quad \forall i \in V, q = 0, 1, \ldots, \text{deg}(i), S \in C_i^{\text{deg}(i) - q}.
\]

Here, we use \( C_i^{\text{deg}(i) - q} \) to denote the set of all combinations with \( \text{deg}(i) - q \) elements chosen from node \( i \)'s neighbors. \( S \) is one combination picked from \( C_i^{\text{deg}(i) - q} \). For a given \( i \), if \( q \geq g_i \), Case 3 extreme directions generate constraints that are redundant. Thus, the projection of \( P_{\text{extended}} \) onto the \( \mathbf{x} \) space is the following:

\[
g_i x_i + \sum_{j \in n(i)} x_j \geq g_i \quad \forall i \in V
\]

\[
(g_i - q)x_i + \sum_{j \in S} x_j \geq g_i - q \quad \forall i \in V, q = 1, 2, \ldots, g_i - 1, S \in C_i^{\text{deg}(i) - q}
\]

\[
0 \leq x_i \leq 1 \quad \forall i \in V
\]
Constraint (23) is obtained from constraint (22) when \( q = 0 \). We list it separately to emphasize that constraint (23) is identical to constraint (2) in BIP1. Constraint (24) represents the new inequalities obtained from the projection. To illustrate this set of valid inequalities, using node 1 in Figure 1(a) as an example, we have \( g_1 = 3 \) and \( n(i) = \{2, 3, 4\} \). Thus, when \( q = 1 \), \( C_{3^{-1}}^{1-2} = \{\{2, 3\}, \{2, 4\}, \{3, 4\}\} \). When \( q = 2 \), \( C_{3^{-2}}^{1-1} = \{\{2\}, \{3\}, \{4\}\} \). Except for the lower- and upper-bound constraints, node 1’s corresponding constraints in the projection are as follows:

\[
\begin{align*}
3x_1 + x_2 + x_3 + x_4 &\geq 3 \\
2x_1 + x_2 + x_3 &\geq 2, & \text{for } q = 0, \\
x_1 + x_2 + x_3 + x_4 &\geq 2, & \text{for } q = 1, \\
x_1 + x_2 &\geq 1, & \text{for } q = 2.
\end{align*}
\]

Among these constraints, \( x_1 + x_2 \geq 1 \) is violated by the aforementioned solution with \( x_1 = \frac{1}{3} \) and \( x_2 = 0 \). Intuitively, for a node \( i \) and given the \( q \) value, the set of inequalities (24) can be interpreted as follows: either node \( i \) is selected or at least \( g_i - q \) nodes are selected among node \( i \)’s \( \deg(i) - q \) neighbors. Since we now have the projection of EPIDS (feasible region of LP2) onto the space of the \( x \) variables, Theorem 2 implies that constraints (23), (24) and (25) give the complete description of the polytope for the PIDS problem on trees. We state this as Theorem 4.

**Theorem 4.** The polytope for the PIDS problem on trees is described by constraints (23), (24) and (25).

When we replace (25) by their binary counterparts, we obtain a new formulation (that we refer to as BIP4) for the PIDS problem (Theorem 4 showed that we can drop the binary restriction for trees).

\[
\text{(BIP4)} \quad \text{Minimize } \sum_{i \in V} b_i x_i
\]

\[
\text{Subject to: } \quad (23), (24)
\]

\[
x_i \in \{0, 1\} \quad \forall i \in V
\]

Let LP4 be the linear relaxation of BIP4. Since the feasible region of LP4 is the projection of \( P_{\text{extended}} \), \( z_{LP3} = z_{LP4} \), where \( z_{LP4} \) denotes the optimal value of the linear relaxation of BIP4. We note that constraint (24) is exponential in size. The following proposition shows that the separation problem can be solved in polynomial time. Recall that \( \delta \) denotes the largest degree among all nodes in \( V \), i.e., \( \delta = \max\{\deg(i) : i \in V\} \).

**Proposition 2.** The valid inequalities (24) can be separated in \( O(|V|\delta \log \delta) \) time.
Algorithm 1 Separation Algorithm for Inequality Set (24)

Require: A solution \( x^* \) and a PIDS instance.

1. for \( i \in V \) do
2. \( S \leftarrow n(i) \) and sort nodes in \( S \) in descending order by their \( x^* \) values.
3. for \( q = 1, 2, \ldots, g_i - 1 \) do
4. \( m_q = \arg \max \{x^*_i : i \in S\} \) and \( S \leftarrow S \setminus m_q \).
5. if \( (g_i - q)x^*_i + \sum_{j \in S} x^*_j < g_i - q \) then
6. \( \text{Add} (g_i - q)x^*_i + \sum_{j \in S} x^*_j \geq g_i - q. \)
7. else
8. Break
9. end if
10. end for
11. end for

Proof. Given a fractional solution \( x^* \), a node \( i \) in \( V \) and a specific \( q \), where \( q = 1, 2, \ldots, g_i - 1 \), the corresponding separation procedure of inequality (24) can be formulated as the following optimization problem:

\[
\begin{align*}
\text{Minimize} & \quad (g_i - q)x^*_i + \sum_{j \in n(i)} x^*_j z_j \\
\text{Subject to} & \quad \sum_{j \in n(i)} z_j = \deg(i) - q \\
& \quad z_j \in \{0, 1\} \quad \forall j \in n(i)
\end{align*}
\]

(26) (27) (28)

For each node \( i \) in \( V \), if node \( i \) is in the set \( S \), the binary variable \( z_i \) is 1. Otherwise, it is 0. If the optimal objective value is smaller than \( g_i - q \), we have a violated constraint. Otherwise, we change the value of \( q \) or the value of \( i \). This optimization problem has one constraint and can be solved by taking the smallest \( \deg(i) - q \) values of \( x^*_j \) among node \( i \)’s neighbors \( (j \in n(i)) \). We can use Algorithm 1 to separate the whole inequality set (24).

First, the solution \( x^* \) satisfies \( g_i x^*_i + \sum_{j \in S} x^*_j \geq g_i \) for all \( i \) in \( V \). For the inequality \( (g_i - q)x^*_i + \sum_{j \in S} x^*_j < g_i - q \), as \( q \) increases by 1, the left-hand side decreases by \( x^*_i + x^*_m \), where \( x^*_m \) is the \( q \)th largest value of \( x^*_j \) for all \( j \) in \( n(i) \), and the right-hand side decreases by 1. For the current iteration \( q_0 \), if it is the first time that \( (g_i - q_0)x^*_i + \sum_{j \in S} x^*_j > g_i - q_0 \), then, it means \( x^*_i + x^*_m < 1 \). Then, in a future iteration, we will not find any violated constraint for this particular \( i \) because \( x^*_i + x^*_m < x^*_i + x^*_m < 1 \) when \( q > q_0 \). Therefore, in Algorithm 1, we use Break in line 8. For each node, we sort its neighbors, which takes at most \( O(\delta \log \delta) \) steps, and make at most \( \delta \) comparisons. The process is repeated for \(|V| \) nodes. Thus, the overall time complexity is \( O(|V| \delta \log \delta) \).

4. Polyhedral Study of the PIDS Problem

In this section, we conduct a polyhedral study of the PIDS problem on arbitrary graphs based on BIP4. The convex hull of the incidence vectors of all PIDs of \( G \), denoted by \( P_T(G) \), is called the PIDS polytope of \( G \). Thus, the PIDS problem is equivalent to the linear program \( \min\{b^T x : x \in \)
$P_T(G)$. Also, let $1 \in \mathbb{R}^{|V|}$ denote the vector that contains all 1s, and $e_i \in \mathbb{R}^{|V|}$ denote the vector that has 1 in $i$th position, but 0s in all other positions. First we show that $P_T(G)$ is full dimensional (i.e., the dimension of $P_T(G)$ is $|V|$).

**Theorem 5.** $P_T(G)$ is full dimensional.

**Proof.** We prove this by showing that there are $|V| + 1$ affinely independent points in $P_T(G)$. The first point is $x_0 = 1$. Another $|V|$ points can be obtained as: $x_i = 1 - e_i$ for all $i$ in $V$. We obtain $|V|$ linearly independent points by $(x_i - x_0) = -e_i$ for all $i$ in $V$. Thus, $x_0$ and $x_i$ for all $i$ in $V$ are affinely independent. Furthermore, all of these points are feasible in $P_T(G)$. First, $x_0$ means that we pick all nodes. Second, $x_i$ means that we pick all nodes but a node $i$. Thus, node $i$’s threshold requirement is satisfied because all of its neighbors are in the PIDS. Therefore, $P_T(G)$ has $|V| + 1$ affinely independent points. $\Box$

Knowing the dimension of $P_T(G)$, we study the facet-defining conditions for constraints (23), (24) and (25). We start with constraint (25). We show that $x_i \leq 1$ for all $i$ in $V$ are facet defining, and $x_i \geq 0$ for all $i$ in $V$ define faces for $P_T(G)$. Then, we present the conditions under which $x_i \geq 0$ for all $i$ in $V$ are facet defining for $P_T(G)$.

**Proposition 3.** The trivial constraint $x_i \leq 1$ for all $i$ in $V$ are facet defining for $P_T(G)$.

**Proof.** Given a node $i$ in $V$, when $x_i = 1$, we can find the following $|V|$ affinely independent points: The first one is $x_0 = 1$. Then, we can have $(|V| - 1)$ more points as $x_j = 1 - e_j$ for all $j$ in $V \setminus \{i\}$. From these points, we can obtain $(|V| - 1)$ linearly independent points as $(x_j - x_0) = -e_j$ for all $j$ in $V \setminus \{i\}$. Thus, $x_0$ and $x_j$ for all $j$ in $V \setminus \{i\}$ are affinely independent. They are feasible points, as shown in Theorem 5. Consequently, $x_i \leq 1$ is a facet of $P_T(G)$. $\Box$

**Proposition 4.** For a node $i$ in $V$, its corresponding trivial constraint $x_i \geq 0$ is a face of $P_T(G)$. Furthermore, $x_i \geq 0$ is facet defining for $P_T(G)$ if the following conditions are satisfied:

1. $g_i \leq \deg(i) - 1$.
2. For a node $j$ in $n(i)$, $g_j \leq \deg(j) - 1$ if it does not share neighbors with node $i$ (i.e., $n(j) \cap n(i) = \emptyset$). Otherwise, $g_j \leq \deg(j) - 2$

**Proof.** $x_i \geq 0$ is satisfied with equality because of the feasible point $x_i = 1 - e_i$. Thus, $x_i \geq 0$ is a face of $P_T(G)$. When Conditions 1 and 2 of Proposition 4 are satisfied, we can show that $x_i \geq 0$ is a facet of $P_T(G)$. Let $F(i) = \{j \in n(i) : g_j \leq \deg(j) - 1 \text{ if } n(i) \cap n(j) = \emptyset, g_j \leq \deg(j) - 2 \text{ if } n(i) \cap n(j) \neq \emptyset\}$ be the subset of node $i$’s neighbors that satisfy Condition 2. We can find the points: $x_i = 1 - e_i$, and $x_j = x_i - e_j$ for all $j$ in $F(i)$, and $x_k = x_i - e_k$ for all $k$ in $\{V \setminus n(i)^+\}$, where $n(i)^+ = n(i) \cup i$. First, $x_j = x_i - e_j$ for all $j$ in $F(i)$ are not feasible if $g_j = \deg(j)$. However, they are feasible when $g_j \leq \deg(i) - 1$. Second, $x_i$ and $x_k$ for all $k$ in $\{V \setminus n(i)^+\}$ are feasible and distinct.
We have $|V| - |n(i)| - 1 + |F(i)| + 1$ feasible points in $P_T(G)$. Also, $x_j - x_i = -e_j$ for all $j$ in $\{V \setminus n(i) + \cup F(i)\}$ are linearly independent. Thus, when $|F(i)| = |n(i)|$ (i.e., Condition 2 is satisfied), we have $|V|$ affinely independent points.

Next, we study constraints (23) and (24). Recall that together, they correspond to constraint (22), where constraint (23) is obtained from constraint (22) when $q = 0$, and constraint (24) is obtained with the remaining $q$ values. We will prove the necessary and sufficient facet-defining conditions for constraint (22). For ease of exposition, for a node $i$ with a given $g_i$ and $q$ value, we will use the notation $a_i$ to refer to $g_i - q$ in constraint (22) (rather than repeatedly writing $g_i - q$, we will use $a_i = g_i - q$ as a short-hand notation).

Figure 8 provides an example to illustrate the notation we will use. The number beside a node is its threshold value. Here, we have $g_i = 3$, $q = 1$. Thus, $a_i = 2$ and $S = \{j_1, j_2, j_3\}$. Let $N_{all} = \{j \in n(i): g_j = \text{deg}(j)\}$ (i.e., the set of node $i$’s neighbors whose threshold is equal to their degree). In Figure 8, we have $N_{all} = \{j_3, j_4\}$. Let $H_1 = \{k \in n(S) \setminus i: g_k > \text{deg}(k) - |n(k) \cap S|\}$. Here, $n(S)$ denotes the nodes adjacent to the nodes in $S$. Thus, $H_1$ contains nodes that must be selected in the PIDS if no node in $S$ is selected (since the number of neighbors they have outside $S$, i.e., $\text{deg}(k) - |n(k) \cap S|$, is less than their threshold). In Figure 8, $H_1 = \{k_2, k_3\}$. For node $k_2$, $g_{k_2} = 2$ and $\text{deg}(k_2) - |n(k_2) \cap S| = 1$. For node $k_3$, $g_{k_3} = 2$ and $\text{deg}(k_3) - |n(k_3) \cap S| = 1$. Finally, for a node $k$ in $H_1$, $\gamma_k = g_k - (\text{deg}(k) - |n(k) \cap S|)$ calculates the fewest number of node $k$’s neighbors in $S$ that must be selected in the PIDS when node $k$ is not selected in the PIDS.

Our main result is the following theorem.

**Theorem 6.** Inequality (22) is facet defining if and only if the following four conditions are satisfied:

1. $\text{deg}(j) - |n(j) \cap S| \geq g_j$ for all $j$ in $S$.
2. $N_{all}$ is an empty set.
3. $1 \leq g_i - q \leq |S| - 1$ if $|S| \geq 2$ and $g_i - q = 1$ if $|S| = 1$. 

\[ \begin{array}{c|c}
 a_i & 2 \\
 S & \{j_1, j_2, j_3\} \\
 N_{all} & \{j_3, j_4\} \\
 H_1 & \{k_2, k_3\} \\
 S^+ & S \cup \{i\} = \{j_1, j_2, j_3\} \\
 S_{all} & S \cap N_{all} = \{j_3\} \\
 S \setminus S_{all} & \{j_1, j_2\} \\
 C_{S \setminus S_{all}} & \{\{j_1\}, \{j_2\}\} \\
 H_C & \{k \in n(S) \setminus i: g_k > |n(k) \cap S^+| + |n(k) \cap (C \cup S_{all})|\} \\
 \text{Given } C = \{j_1\} \text{ and } C \cup S_{all} = \{j_1, j_3\}, H_C = \{k_3\} \\
 \text{Given } C = \{j_2\} \text{ and } C \cup S_{all} = \{j_2, j_3\}, H_C = \{k_1\} \\
 H_0 & \{k \in n(S^+) \setminus N_{all}: g_k > \text{deg}(k) - |n(k) \cap S^+|\} = \{k_1, k_2, k_3\} \\
 \end{array} \]
4. \( \gamma_j \leq g_i - q \) for each node \( j \) in \( H_1 \).

We first prove several lemmas before proving Theorem 6. We will largely use Approach 2 on page 144 of Wolsey (1998) to show that inequality (22) is facet defining with the above necessary and sufficient conditions. First, we need to show that there are at least \(|V|\) feasible points satisfying the given inequality (22) with equality. Lemmas 1, 2 and 3 help us find such feasible points in \( P_T(G) \) with inequality (22) binding. The first lemma states that if \( x_i = 1 \) and the given inequality is binding, a node \( j \) in \( S \) must have at least \( g_j \) neighbors that are not in \( S \).

**Lemma 1.** If inequality (22) is binding and \( x_i = 1 \), we must have: \( \deg(j) - |n(j) \cap S| \geq g_j \) for all \( j \) in \( S \).

**Proof.** Assume that this is not true. Given that \( x_i = 1 \) and the inequality is binding, we cannot select any node \( j \) in \( S \). Furthermore, to satisfy node \( j \)'s threshold requirement, at least \( g_j \) of node \( j \)'s neighbors should be selected because \( x_j = 0 \). However, given \( \deg(j) - |n(j) \cap S| < g_j \), we must select some nodes in \( S \). A contradiction is found. \( \square \)

The second lemma states that if \( x_i = 0 \) and inequality (22) is binding, at most \( a_i \) nodes in \( S \) can have their threshold equal to their degree. Let \( S_{\text{all}} = S \cap N_{\text{all}} \). In Figure 8, we have \( S_{\text{all}} = \{j_3\} \).

**Lemma 2.** If inequality (22) is binding and \( x_i = 0 \), we must have \( |S_{\text{all}}| \leq a_i \).

**Proof.** Assume that this is not true. Given that \( x_i = 0 \) and \( |S_{\text{all}}| > a_i \), we must have \( x_j = 1 \) for all \( j \) in \( S_{\text{all}} \). Then, \( a_i x_i + \sum_{j \in S} x_j \geq \sum_{j \in S_{\text{all}}} x_j > a_i \). A contradiction is found. \( \square \)

In Lemma 3, we show that in order to find \(|V|\) affinely independent points that satisfy inequality (22) at equality (which is necessary to show that inequality (22) is facet defining), we must satisfy the conditions of Lemmas 1 and 2.

**Lemma 3.** If inequality (22) is facet defining, (i) \( \deg(j) - |n(j) \cap S| \geq g_j \) for all \( j \) in \( S \), and (ii) \( |S_{\text{all}}| \leq a_i \).

**Proof.** First, if for a given \( i \) and \( S \), inequality (22) violates both Lemmas 1 and 2, it cannot be facet defining because inequality (22) cannot be satisfied as an equality. Second, if it only satisfies Lemma 1, we can only have the inequality binding when \( x_i = 1 \). Then, at most, \(|V| - |S|\) affinely independent points in \( P_T(G) \) can be found because \( x_j = 0 \) for all \( j \) in \( S \). Third, if it only satisfies Lemma 2, we can only have the inequality binding when \( x_i = 0 \). Thus, at least one node in \( S \), denoted by \( j \), violates the condition that \( \deg(j) - |n(j) \cup S| \geq g_j \). Then, if node \( j \) is not selected, at least one other node in \( S \) must be selected. That means we can find at most \(|V| - 1\) affinely independent points. Taken together, this means that in order to obtain \(|V|\) affinely independent points that satisfy inequality (22) at equality, we must satisfy both the requirement of Lemma 1 \((\deg(j) - |n(j) \cap S| \geq g_j \text{ for all } j \in S)\) and Lemma 2 \((|S_{\text{all}}| \leq a_i)\). \( \square \)
Given Lemma 3, we can characterize the feasible points in $P_T(G)$ that satisfy inequality (22) at equality. First, when $x_i = 1$, we have $x_j = 0$ for all $j \in S$. Recall that $H_1 = \{k \in n(S) \mid i : g_k > deg(k) - |n(k) \cap S|\}$. For a node $k$ in $H_1$, it has more than $deg(k) - g_k$ neighbors in $S$. Thus, if all nodes in $S$ are not selected, we must select node $k$ (i.e., $x_k = 1$) in a feasible point in $P_T(G)$.

Thus, we can find $T_1 = |V| - |S| - |H_1|$ points in $P_T(G)$ that satisfy inequality (22) at equality as follows. Let $x_0 = 1 - \sum_{j \in S} e_j$, (i.e., select all nodes in $V \setminus S$ in the PIDS;) and $x_j = x_0 - e_j$ for all $j$ in $\{V \setminus (S^+ \cup H_1)\}$ (i.e., from the solution $x_0$, remove a single node that is not $i$ or in $H_1$). In the example of Figure 8, $x_0$ has nodes $i, j_4, k_1, k_2, k_3, k_4, k_5$. Since $H_1 = \{k_2, k_3\}$, we can get four additional feasible points by removing $j_4, k_1, k_3$, and $k_5$ one at a time from $x_0$.

Second, when $x_i = 0$, we have $x_j = 1$ for all $j \in N_{\text{all}}$ (because they require all of their neighbors to be selected in order to meet their threshold, and one of their neighbors, node $i$, is not selected). Then, to satisfy inequality (22) at equality, we need to choose exactly $a_i - |S_{\text{all}}|$ nodes from the set $S \setminus S_{\text{all}}$ in the PIDS. Let $C_{S \setminus S_{\text{all}}}^{a_i - |S_{\text{all}}|}$ be the set of all combinations. In Figure 8, $C_{S \setminus S_{\text{all}}}^{a_i - |S_{\text{all}}|} = \{\{j_1\}, \{j_2\}\}$. Then, for a given $C \in C_{S \setminus S_{\text{all}}}^{a_i - |S_{\text{all}}|}$, we can have $x_0^C = 1 - \sum_{j \in S^+ \cup \{C \cup S_{\text{all}}\}} e_j$ (i.e., select all nodes except those in $S^+ \setminus \{C \cup S_{\text{all}}\}$). For the instance in Figure 8, if $C = \{j_1\}$, $x_0^C$ selects all nodes except for $i$ and $j_2$. If $C = \{j_2\}$, $x_0^C$ selects all nodes except for $i$ and $j_1$.

Next, we can obtain additional feasible solutions (satisfying inequality (22) at equality) by removing nodes one at a time from $x_0^C$. However, some nodes should not be removed. Otherwise, the resulting solution either no longer satisfies inequality (22) at equality or becomes infeasible to the PIDS. First, we cannot remove any node in $N_{\text{all}}$ or in $S$ (removing a node in $S$ will cause the inequality to no longer be satisfied at equality). Second, for a node $k \in n(S) \setminus i$, if its threshold is strictly greater than the number of its neighbors selected in $x_0^C$, removing node $k$ results in an infeasible solution. Let $H_C = \{k \in n(S) \setminus i : g_k > |n(k) \setminus S^+| + |n(k) \cap \{C \cup S_{\text{all}}\}|\}$. In the example of Figure 8, given $C = \{j_1\}$ and $x_0^C$ as described above, $H_C = \{k_3\}$. Thus, we can obtain four additional feasible points by removing $k_1, k_2, k_4$, and $k_5$ one at a time from $x_0^C$. Similarly, if $C = \{j_2\}$, $H_C = \{k_1\}$. Thus, we can obtain four additional feasible points by removing $k_2, k_3, k_4$, and $k_5$ one at a time from $x_0^C$. Overall, for a feasible point $x_0^C$ associated with a $C$ in $C_{S \setminus S_{\text{all}}}^{a_i - |S_{\text{all}}|}$, we obtain points $x_j^C = x_0^C - e_j$ for all $j$ in $\{V \setminus (S^+ \cup N_{\text{all}} \cup H_C)\}$ (i.e., from the solution $x_0^C$, remove a single node that is not in $C$, $N_{\text{all}}$, $H_C$). Overall, we can find $T_0 = \sum_{C \in C_{S \setminus S_{\text{all}}}^{a_i - |S_{\text{all}}|}}(|V| - |S| - |H_C|) - |N_{\text{all}} \setminus S_{\text{all}}|$ feasible points in the forms $x_0^C$ and $x_j^C$ in $P_T(G)$.

We form a linear equation system based on these feasible points $x_0$, $x_j$, $x_0^C$ and $x_j^C$ with $|V| + 1$ unknowns ($\mu_0, \mu$), where $\mu$ corresponds to the coefficients in the left-hand side of inequality (22), and $\mu_0$ corresponds to the right-hand side value. Feasible point $x_0$ yields equation (29), feasible
point(s) $x_j$ yields equation (30), feasible point $x_0^C$ yields equation (31) and feasible point(s) $x_j^C$ yields equation (32).

$$\mu_i + \sum_{h \in V \setminus S^+} \mu_h = \mu_0$$  \hspace{1cm} (29)

$$\mu_i + \sum_{h \in V \setminus S^+} \mu_h - \mu_j = \mu_0 \quad \forall j \in \{V \setminus (S^+ \cup H_1)\}$$  \hspace{1cm} (30)

$$\sum_{k \in C \cup S_{all}} \mu_j + \sum_{h \in V \setminus S^+} \mu_h = \mu_0 \quad \forall C \in C_{S \setminus S_{all}}^{a_i-|S_{all}|}$$  \hspace{1cm} (31)

$$\sum_{k \in C \cup S_{all}} \mu_j + \sum_{h \in V \setminus S^+} \mu_h - \mu_j = \mu_0 \quad \forall C \in C_{S \setminus S_{all}}^{a_i-|S_{all}|}, j \in \{V \setminus (S^+ \cup N_{all} \cup H_C)\}$$  \hspace{1cm} (32)

Let $(\pi_0, \pi)$ be a vector whose positions are labeled from 0 to $|V|$. Following Approach 2 on page 144 of Wolsey (1998), we need to show that $(\mu_0, \mu)$ yields equation (30) and feasible point(s) $x_0^C$ yields equation (32).

**Lemma 4.** If inequality (22) is facet defining, we must have $\gamma_j \leq a_i$ for each node $j$ in $H_1$.

**Proof.** Equation (29) minus the $j$th equation from the equation set (30) yields $u_j = 0$ for all $j$ in $\{V \setminus (S^+ \cup H_1)\}$. If inequality (22) is facet defining, we must also have the coefficients $u_j = 0$ for $j \in H_1$. Notice, for a given $C$ in $C_{S \setminus S_{all}}^{a_i-|S_{all}|}$, equation (31) minus the $j$th equation from the equation set (32) yields $u_j = 0$ for all $j$ in $\{V \setminus (S^+ \cup N_{all} \cup H_C)\}$. If a node $j \in H_1$ does not belong to $H_C$ for some $C \in C_{S \setminus S_{all}}^{a_i-|S_{all}|}$, then $j \in V \setminus (S^+ \cup N_{all} \cup H_C)$ for that particular $C$, and we can establish that its coefficient $u_j = 0$.

We claim that if a node $j \in H_1$ does not belong to $H_C$ for some $C \in C_{S \setminus S_{all}}^{a_i-|S_{all}|}$, then $\gamma_j \leq a_i$. Otherwise, if $\gamma_j > a_i$ it cannot be removed from any $x_0^C$ because we can only select $a_i$ nodes from $S$ if $x_i = 0$ (i.e., $j$ is in all sets $H_C$ for $C \in C_{S \setminus S_{all}}^{a_i-|S_{all}|}$).

Next, we show the conditions leading to the uniqueness of $(\pi_0, \pi)$.

**Lemma 5.** If inequality (22) is facet defining, we must have the following:

1. $N_{all}$ is an empty set.
2. $1 \leq a_i \leq |S| - 1$ if $|S| \geq 2$ and $a_i = 1$ if $|S| = 1$.

**Proof.** Given Lemma 4, we have 0s for all positions in $V \setminus S^+$. It means that $N_{all} = S_{all}$. We can simplify the equation system of (29) to (32) as:

$$\mu_i = \mu_0$$  \hspace{1cm} (33)

$$\sum_{j \in S_{all}} \mu_j + \sum_{j \in C} \mu_j = \mu_0 \quad \forall C \in C_{S \setminus S_{all}}^{a_i-|S_{all}|}$$  \hspace{1cm} (34)
We know that $C_{S \setminus S_{all}}^{a_i - |S_{all}|}$ is the set of all combinations for choosing $a_i - |S_{all}|$ nodes from the set $S \setminus S_{all}$. Thus, $\mu_j = \frac{m_j}{a_i}$ for all $j$ in $S$ is a solution to (33) and (34). However, it is not the unique solution. Depending on the value of $\sum_{j \in S_{all}} \mu_j$, we can have infinitely many solutions by setting $u_j = \frac{\mu_j - \sum_{j' \in S_{all}} \mu_{j'}}{a_i - |S_{all}|}$ for all $j$ in $S \setminus S_{all}$. For the uniqueness of the desired $(\pi_0, \pi)$, we need two things. First, $S_{all} = \emptyset$. Thus, the solution does not depend on the value of $\sum_{j \in S_{all}} \mu_j$ anymore. Second, when $|S| \geq 2$, we need $a_i \leq |S| - 1$. Then, from $C_{S \setminus S_{all}}^{a_i - |S_{all}|}$, we can take any two combinations $C_1$ and $C_2$: they have the same nodes, except for nodes $j_1$ and $j_2$ such that one has node $j_1$ but not node $j_2$, and one has node $j_2$ but not node $j_1$ (i.e., $C_1 \setminus j_1 = C_2 \setminus j_2$, $C_1 \setminus j_2 = j_1$ and $C_2 \setminus j_1 = j_2$). Take their corresponding (34)s and subtracting one from the other. That gives $u_{j_1} = u_{j_2}$. Repeating this process, we have $u_{j_1} = u_{j_2}$ for any two distinct $j_1$ and $j_2$ in $S$. When $|S| = 1$, we have $u_j = \mu_0$ because $a_i = 1$. Then, $u_j = \frac{m_j}{a_i}$ for all $j$ in $S$. Thus, $\pi_i = \pi_0 = a_i$ and $\pi_j = 1$ for all $j$ in $S$. □

Now, we are ready to present the proof of Theorem 6.

Proof of Theorem 6: Necessity. Proved by Lemmas 3, 4 and 5.

Sufficiency. First, we show that if Conditions 1-4 of Theorem 6 are satisfied, we can find at least $|V|$ feasible points that satisfy inequality (22) at equality. We can find $T_1 = |V| - |S| - |H_1|$ feasible points by setting $x_i = 1$ because of Condition 1. Also, there are $T_0 = |C_S^{a_i}|(|V| - |S|) - \sum_{C \in C_S^{a_i}} |H_C|$ feasible points by setting $x_i = 0$ because of Condition 2. Then,

$$T_0 + T_1 = |V| - |S| - |H_1| + \sum_{C \in C_S^{a_i}} (|V| - |S| - |H_C|)$$

(35)

Let $H_0 = \{k \in n(S) \setminus N_{all} : g_k > deg(k) - |n(k) \cap S^+|\}$. Excluding nodes in $N_{all}$, $H_0$ contains $S$’s neighbors that must be selected in the PIDS if none of the nodes in $S^+$ is selected. In the example of Figure 8, $H_0 = \{k_1, k_2, k_3\}$. Thus, for a given $C$ in $C_S^{a_i}$, $H_0 \setminus H_C$ provides the set of $S$’s neighbors that can be removed one by one from $x_0^C$. Given Condition 4, each node $j$ in $H_1$ can be removed from an $x_0^C$ for some $C$ in $C_S^{a_i}$. Thus, $H_1 \subseteq \cup_{C \in C_S^{a_i}} H_0 \setminus H_C$. Hence, $\sum_{C \in C_S^{a_i}} |H_0| - |H_C| \geq |H_1|$ because $H_C \subseteq H_0$ by definition. Then, for each $C$ in $C_S^{a_i}$, we add and subtract $|H_0|$ to the right-hand side of (35) as follows:

$$T_0 + T_1 = |V| - |S| + \sum_{C \in C_S^{a_i}} (|V| - |S| - |H_0|) + \sum_{C \in C_S^{a_i}} (|H_0| - |H_C|) - |H_1|$$

(36)

$$\geq |V| - |S| + |C_S^{a_i}|(|V| - |S| - |H_0|)$$

(37)

$$\geq |V| - |S| + |C_S^{a_i}|$$

(38)

$$\geq |V|$$

(39)

We can go from (36) to (37) by using the fact that $\sum_{C \in C_S^{a_i}} |H_0| - |H_C| \geq |H_1|$. We can go from (37) to (38) by noting that $|V| - |S| - |H_0| \geq 1$. This is true because at least node $i$ is left in
<table>
<thead>
<tr>
<th>Graph Name</th>
<th>Source</th>
<th># of Nodes</th>
<th># of Edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gnutella</td>
<td>SNAP</td>
<td>10,876</td>
<td>39,994</td>
</tr>
<tr>
<td>Ning</td>
<td>BGU</td>
<td>9,727</td>
<td>40,570</td>
</tr>
<tr>
<td>Hamsterster</td>
<td>Konect</td>
<td>1,788</td>
<td>12,476</td>
</tr>
<tr>
<td>Escorts</td>
<td>Konect</td>
<td>10,106</td>
<td>39,016</td>
</tr>
<tr>
<td>Anybeat</td>
<td>N.R.</td>
<td>12,645</td>
<td>49,132</td>
</tr>
<tr>
<td>Advogato</td>
<td>N.R.</td>
<td>5,042</td>
<td>39,277</td>
</tr>
<tr>
<td>Delicious</td>
<td>N.R.</td>
<td>536,108</td>
<td>1,365,961</td>
</tr>
<tr>
<td>Youtube</td>
<td>BGU</td>
<td>1,134,890</td>
<td>2,987,624</td>
</tr>
<tr>
<td>Lastfm</td>
<td>Konect</td>
<td>1,191,805</td>
<td>4,519,330</td>
</tr>
<tr>
<td>Flixster</td>
<td>N.R.</td>
<td>2,523,386</td>
<td>7,918,801</td>
</tr>
</tbody>
</table>

Table 1 Source and size of real-world graphs.

V \ (S \cup H_0). Finally, we can go from (38) to (39) because Condition 3 of Theorem 6 implies that \(|C^a_i| \geq |S|\). (Observe that when \(|S| = 1\) and \(a_i = 1\), \(|C^a_i| = |S|\). When \(|S| \geq 2\) and \(1 \leq a_i \leq |S| - 1\), first consider \(a_i \leq \lfloor \frac{|S|}{2} \rfloor\), \(|C^a_i| = \left( \frac{|S|}{a_i} \right) = \left( \frac{|S|}{a_i-1} \right) \prod_{h=2}^{a_i} \frac{|S|+h-1}{h} \geq |S| \prod_{h=2}^{a_i} \frac{|S|+h-1}{h} \geq |S|\) since \(\frac{|S|+h-1}{h} \geq 1\) for all \(h \leq a_i \leq \lfloor \frac{|S|}{2} \rfloor\). Next, consider that \(a_i > \lfloor \frac{|S|}{2} \rfloor\); we have \(|S| = a_i \leq \lfloor \frac{|S|}{2} \rfloor\). Then, \(|C^a_i| = \left( \frac{|S|}{a_i} \right) = \left( \frac{|S|}{|S|-a_i} \right) \geq |S|\). Thus, we have shown that there are at least \(|V|\) feasible points in \(P_T(G)\) satisfying inequality (22) with equality.

Using these \(T_0 + T_1\) points, we form the linear equation system as (29) to (32). Arguing in a similar manner as Lemmas 4 and 5, we show that when Conditions 2, 3 and 4 are satisfied, \((\mu_0, \mu) = \lambda(\pi_0, \pi)\), where \(\lambda \neq 0\), and \((\pi_0, \pi)\), which has \(a_i\) in the 0th and \(i\)th positions, 1 in the \(j\)th position for all \(j \in S\) and 0s in all other positions, is the only solution for the linear equations system from (29) to (32). Thus, inequality (22) is facet defining for \(P_T(G)\) when these conditions are satisfied.
Group (BGU, see Lesser et al. 2013), the Koblenz Network Collection (KONECT, see Kunegis 2017), and the Network Repository (N.R., see Rossi and Ahmed 2015). In all of our graphs, nodes represent users and edges are connections between users. We first convert any directed graphs into undirected ones (i.e., replacing an arc \((i,j)\) by an edge \(\{i,j\}\) if there is an arc between node \(i\) and node \(j\)). If multiple connected components exist in a graph, we use the biggest connected component in our computational experiments. In Table 1, we list each graph, along with its source repository and the number of nodes and edges it contains.

Our first set of experiments are built on seven large real-world networks. *Gnutella* is a large file sharing peer-to-peer network (the first decentralized peer-to-peer network of its kind). We consider one snapshot of the Gnutella network collected on August 4, 2002. The Gnutella network on August 4, 2002, has 10,876 nodes and 39,994 edges. *Ning* is an online platform for people and organizations to create custom social networks. Snapshots of the friendship and group affiliation networks from Ning were harvested during September 2012. It has 9,727 nodes and 40,570 edges. *Hamsterster* contains friendships between users of the website hamsterster.com. It has 1,788 nodes and 12,476 edges. *Escorts* is the bipartite network of buyers and their escorts. Nodes are buyers and escorts. An edge denotes a transaction between a buyer and an escort. We treat buyers and escorts in the same way and do not distinguish between them when we select a PIDS. Escorts has 10,106 nodes and 39,016 edges. *Anybeat* is an online community, a public gathering place where one can interact with people from around one’s neighborhood or across the world. The data is the friendship network and has 12,645 nodes and 49,132 edges. *Advogato* is based on the friendship network of Advogato.org. It has 5,042 nodes and 39,277 nodes. *Delicious* is a social bookmarking web service for storing, sharing and discovering web bookmarks. This contains the friendship network with 536,108 nodes and 1,365,961 edges.

Our last experiment focuses on BIP4 and three very large real-world social networks. *Youtube* is a video-sharing website on which users can upload, share, and view videos. This data contains the friendship network crawled on Youtube. An edge means that a user subscribes to the other users’ content. It has 1,134,890 nodes and 2,987,624 edges. *Last.fm* is a relevant online service in music-based social networking. The idea of Last.fm is to create a recommendation system based on plugins for all kinds of music listening platforms. It is a music social network that allows users to create a profile and augment it with the music tracks that they listen to. *Lastfm* represents the friendship among the users collected in November 2008. It has 1,191,805 nodes and 4,519,330 edges. *Flixster* is the social network of flixster.com, a movie rating site on which people can meet others with a similar movie tastes. This contains the friendship network crawled in December 2010, which has 2,523,386 nodes and 7,918,801 edges.
The threshold value $g_i$ associated with a node $i$ in a given network is generated from a discrete uniform distribution between $[1, \text{deg}(i)]$. By this method, we ensure that if all neighbors of a node are active, this node will become active as well. Additionally, weight $b_i$ is generated from a discrete uniform distribution between $[1, 50]$. Then, for each social network, ten instances are generated. Thus, there are 100 instances in total, given that we have 10 real-world social networks.

### 5.2. Investigating the Strength of the LP Relaxations

We have already shown that BIP3 and BIP4 are stronger than BIP1 and BIP2 in Theorem 1. Now, we empirically evaluate how much stronger BIP3 and BIP4 are compared to BIP1 and BIP2 in this section. In our implementation for BIP3, we remove the constraint $y_{id} + y_{di} = 1$ and use only variables $y_{di}$ (so $y_{id}$ is replaced by $1 - y_{di}$ in the model). In this way, we reduce the size of the model by $|E_d|$ constraints and $|E_d|$ variables.
We use 70 real-world social network instances, based on the seven graphs Gnutella, Ning, Hamsterster, Escorts, Anybeat, Advogato and Delicious, to compare the strength of the LP relaxations of the four formulations. Recall that LP1, LP2, LP3 and LP4 denote the LP relaxations of BIP1, BIP2, BIP3 and BIP4, respectively, and $z_{LP1}$, $z_{LP2}$, $z_{LP3}$ and $z_{LP4}$ denote their objective values. As we showed earlier, $z_{LP1} = z_{LP2}$ and $z_{LP3} = z_{LP4}$. We consider the relative improvement of LP3 and LP4 over LP1 and LP2 calculated as $\frac{z_{LP3} - z_{LP1}}{z_{LP1}} \times 100$. Figure 9 plots the average, minimum and maximum value of the relative improvement over the ten instances for each of the seven real-world graphs. On average, the improvement is about 20%, with the biggest improvement being over 30% for Hamsterster, and the smallest being about 11% for Delicious. It is clear that our formulations (BIP3 and BIP4) are able to improve the LP relaxation significantly. However, the improvement comes at some cost. In Table 2, we report the running times of LP1, LP2, LP3 and LP4 in seconds and plot the average running time in Figure 10. For LP4, the running time includes both the separation and solving time. First, LP1 and LP2 are very close in terms of the running time. Second, although LP3 and LP4 improve the value of the LP relaxation objective significantly, they need much more time than LP1 and LP2. LP3 usually needs two orders of magnitude more running time than LP1 and LP2. This is caused by the much bigger size of the formulation. LP1 has $|V|$ variables and $|V|$ constraints. However, LP3 has $|V| + 2|E|$ variables and $4|E| + |V|$ constraints (even after we remove the constraint $y_{id} + y_{di} = 1$ and used only variables $y_{di}$). Comparing LP3 and LP4, LP4 has a faster running time.

Overall, LP3 and LP4 need much more time, while they are able to provide stronger LP bounds than LP1 and LP2. A natural question is whether the extra effort involved in solving the LP
<table>
<thead>
<tr>
<th></th>
<th>BIP1</th>
<th></th>
<th>BIP2</th>
<th></th>
<th>BIP3</th>
<th></th>
<th>BIP4</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Avg</td>
<td>Min</td>
<td>Max</td>
<td>Avg</td>
<td>Min</td>
<td>Max</td>
<td>Avg</td>
<td>Min</td>
</tr>
<tr>
<td>Gnutella</td>
<td>9.01%</td>
<td>8.60%</td>
<td>9.49%</td>
<td>8.97%</td>
<td>8.53%</td>
<td>9.41%</td>
<td>0.17%</td>
<td>0.09%</td>
</tr>
<tr>
<td>Ning</td>
<td>5.11%</td>
<td>4.28%</td>
<td>6.03%</td>
<td>3.94%</td>
<td>3.56%</td>
<td>4.31%</td>
<td>0.18%</td>
<td>0.06%</td>
</tr>
<tr>
<td>Hamsterster</td>
<td>12.64%</td>
<td>11.42%</td>
<td>13.86%</td>
<td>10.20%</td>
<td>8.89%</td>
<td>11.65%</td>
<td>0.59%</td>
<td>0.00%</td>
</tr>
<tr>
<td>Escorts</td>
<td>7.42%</td>
<td>6.90%</td>
<td>8.05%</td>
<td>7.15%</td>
<td>6.75%</td>
<td>7.71%</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>Anybeat</td>
<td>1.58%</td>
<td>1.17%</td>
<td>2.02%</td>
<td>1.41%</td>
<td>1.23%</td>
<td>1.57%</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>Advogato</td>
<td>12.38%</td>
<td>10.38%</td>
<td>13.70%</td>
<td>9.57%</td>
<td>8.57%</td>
<td>10.79%</td>
<td>0.69%</td>
<td>0.34%</td>
</tr>
<tr>
<td>Delicious</td>
<td>2.23%</td>
<td>2.17%</td>
<td>2.31%</td>
<td>2.18%</td>
<td>2.12%</td>
<td>2.26%</td>
<td>0.62%</td>
<td>0.53%</td>
</tr>
</tbody>
</table>

Table 3  Optimality gap (%) of BIP1, BIP2, BIP3 and BIP4.

Figure 11  Average optimality gap (%) of BIP1, BIP2, BIP3 and BIP4.

relaxations of BIP3 and BIP4 yields a computational benefit (against using BIP1 and BIP2) when solving them as IP problems. We answer this question in the next experiment.

5.3. Testing the Performance of BIP1, BIP2, BIP3 and BIP4

In this section, we test the performance of BIP1, BIP2, BIP3 and BIP4. Recall that in our implementation for BIP3, we remove the constraint \( y_{id} + y_{di} = 1 \) and use only variables \( y_{di} \). Next, in our implementation for BIP4, we start with constraint (23) and use CPLEX’s callbacks to add the violated constraint (24) dynamically since constraint (24) is exponentially sized. Given that the graphs we consider contain millions of nodes and edges, we only add violated constraints (24) at the root node. Note that while we do not add constraint (24) after the root node, the upper and lower bounds after the root node are still globally valid in the search process (because all solutions satisfy constraint (23)). Lastly, in order to focus on the effect of the four formulations, we turn off CPLEX’s cuts and allow only one thread. Other than that, we keep the default setting for CPLEX.

For each instance, the running time is capped at 3600 seconds (1 hour), unless stated otherwise.
The optimality gap is reported in Table 3. The first column has the identifier for each graph; then, ’Avg”, “Min” and “Max” give the average, minimum and maximum values for each major column, respectively. Let $z_{BFS}$ and $lb$ be the objective value of the best feasible solution and the lower bound obtained by CPLEX when the time limit is reached, respectively. The optimality gap is calculated as $rac{z_{BFS} - lb}{z_{BFS}} \times 100$. Figure 11 shows the average optimality gap for these four formulations across all seven graphs. BIP1 cannot solve any instance to optimality. The optimality gap of BIP1 is much bigger than that of BIP3 and BIP4. For example, the average optimality gap of BIP1 is over 12% for Hamsterster, compared to 0.59% for BIP3 and 0.35% for BIP4. Compared to BIP3 and BIP4, BIP2 behaves in a similar fashion to BIP1, although it performs slightly better than BIP1. Given these results, we can answer the question raised at the end of the last section. BIP3 and BIP4 outperform BIP1 and BIP2.

At first glance, BIP3 and BIP4 have a similar performance in Table 3. Next, we take a closer look at BIP3 and BIP4. Table 4 shows the average optimality gaps with time limits of 300, 600, 900, and 3600 seconds. We can observe that BIP4 is able to close the gap much faster than BIP3. For example, on the Gnutella instances, the optimality gap of BIP3 is 98.73%, 1.58%, 0.26% and 0.17% as the time limit increases, while BIP4 has an optimality gap of 0.21% after 300 seconds. This is even more dramatic for the Delicious instances. After 900 seconds, BIP3 has a 98.86% average optimality gap. In contrast, BIP4 has an average optimality gap of 0.64% after 300 seconds. Furthermore, Table 5 contains the running times of BIP3 and BIP4 for those solved instances. The column “Solved #” presents the number of solved instances. In addition to all instances solved by

<table>
<thead>
<tr>
<th>Time Limit</th>
<th>BIP3</th>
<th>BIP4</th>
</tr>
</thead>
<tbody>
<tr>
<td>300</td>
<td>98.73%</td>
<td>0.21%</td>
</tr>
<tr>
<td>600</td>
<td>1.58%</td>
<td>0.18%</td>
</tr>
<tr>
<td>900</td>
<td>0.26%</td>
<td>0.15%</td>
</tr>
<tr>
<td>3600</td>
<td>0.17%</td>
<td>0.11%</td>
</tr>
</tbody>
</table>

Table 4   Average optimality gap (%) of BIP3 and BIP4 with 300s, 600s, 900s and 3600s time limits.

<table>
<thead>
<tr>
<th>Solved #</th>
<th>Avg</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>4</td>
<td>2553.9</td>
<td>1536.3</td>
<td>3272.0</td>
</tr>
</tbody>
</table>

Table 5   Running time (in seconds) of BIP3 and BIP4 on solved instances.
BIP3, BIP4 is able to solve four more Ning instances. For the instances solved by both BIP3 and BIP4, BIP4 is much faster, compared to BIP3. This can be seen in Figure 12, which shows the average running time on instances solved by both BIP3 and BIP4. For Hamsterster and Anybeat instances, the minimum running time of BIP3 is greater than the maximum running time of BIP4. Thus, as we noted earlier, the larger size of BIP3 deteriorates the performance as the size of the instances becomes larger. As we will see, the difference becomes rather apparent on the very large-scale real-world social networks.

Next, we consider the thirty instances on the three very large real-world social networks: Youtube, Lastfm and Flixster. Given our results on the first 70 instances discussed earlier, only BIP3 and BIP4 are applied to these 30 instances. When BIP3 is applied to these instances, it does not solve the LP relaxation at the root node for any instance within the time limit. Hence, we cannot even obtain a dual bound or gap with BIP3. Needless to say, when the LP relaxation of an IP formulation cannot be solved, the model is not viable because the search process is predicated on solving the initial LP relaxation. Consequently, BIP3 is not a viable option for very large instances. However, BIP4 is able to find and prove good quality solutions for all of these instances. The results for BIP4 are shown under the column “Optimality Gap” in Table 6. BIP4 solves 24 out of 30 instances optimally. For the remaining six Youtube instances, the optimality gap is less than 0.02%. Table 6 reports the running time on the solved instances.
Based on the above experiments, we have demonstrated that BIP4 is a high-quality formulation for the PIDS problem. It has both theoretical desirable properties, and is also capable of solving problems with 2.5 million nodes and 8 million edges.

6. Conclusions

In this paper, we studied an influence maximization problem whereby direct influence plays the dominant role in influence propagation. This problem, referred to as the PIDS problem, generalizes the celebrated dominating set problem and has applications in multiple settings, including technology diffusion, support social networks, and settings where rapid influence propagation is desired.

We propose a strong and compact extended formulation for the PIDS problem. Then, we project it onto the node variable space. We also show that the extended formulation is the strongest possible formulation for the PIDS problem on trees. Thus, its projection on the natural node variable space gives a complete description of the polytope for the PIDS problem on trees. We derive a new set of valid inequalities for the PIDS problem and provide a polynomial-time separation procedure for it. Facet-defining conditions of the proposed valid inequalities are derived for arbitrary graphs. We conduct an extensive computational study on real-world graphs to show the efficacy of the proposed formulations. On a test-bed of 100 real-world graph instances, our results show that our approach can find and prove high-quality solutions quickly for very large graphs (up to approximately 2.5 million nodes and 8 million edges). We find solutions that are on average 0.2% from optimality and solve 51 out of the 100 instances to optimality.

Our approach starts with a valid integer programming formulation with a small number of variables and constraints. Then, we dynamically add constraints (from an exponential family of facet-defining constraints) that are violated by fractional solutions to the relaxation, to iteratively strengthen the formulation. In this way, on the one hand, the method provides a strong dual bound in terms of the LP relaxation; while on the other hand, the formulation has a very manageable size (even after adding the violated constraints). The nice feature of this approach is that even if we stop adding violated constraints once a suitably tight optimality gap has been achieved, and simply focus on branching thereafter, the approach remains valid, and as demonstrated by our experiments, is computationally efficient.

We now discuss several variants and extensions of the PIDS problem. In the PIDS problem, influence is only allowed to propagate one step, while in the WTSS problem, influence is allowed to propagate indefinitely (or \(|V| - 1\) steps). These represent two extremes: one where only direct influence plays a role or where rapid influence is desired, and the other, where indefinitely long chains of indirect influence are permitted or where rapid influence is not a requirement. In between
these lies the idea of general latency constraints. In the event that second order, third order, and in general \( n \)th order influences play a role, we can formulate an influence maximization problem with latency constraints where we are allowed a prespecified number of steps/time periods for the influence to propagate through the network. This is a problem of significant practical relevance, but remains a challenging next step (since the formulation for the PIDS problem and the WTSS problem cannot be applied directly).

Another variant of the PIDS problem is the positive-influence target-dominating set (PITD) problem (proposed by Tong et al. 2017). In the PITD problem, it is desirable to influence a particular subset of nodes called the target (instead of the entire graph) and the goal is to find a PIDS that influences the target. Our BIP4 formulation applies in this setting, with the proviso that constraints (23) and (24) are only defined for the nodes in the target.

One extension of the PIDS problem is to consider a setting where partial payments to a node are permitted (similar to Fischetti et al. 2016, G¨unne¸c et al. 2020). Instead of having two ways that nodes are influenced, i) either the entire amount \( b_i \) is paid to a node \( i \) that is directly influenced or, ii) nothing is paid (to a node that is not directly influenced) and the node \( i \) requires \( g_i \) neighbors to be in the PIDS; it is also possible to influence a node with a partial payment (between 0 and \( b_i \)) and a correspondingly fewer number of neighbors (between 1 and \( g_i \)) required to be directly influenced (i.e., in the PIDS). Raghavan and Zhang (2020) study this problem and refer to it as the PIDS with Partial Payments (PIDS-PP) problem. They develop a strong formulation for the PIDS-PP problem by effectively characterizing influence types propagated on the network (along with edge-splitting). They discuss the projection of this strong formulation onto a “payment space” and their computational experiences on a large set of real-world networks with these formulations.

Another natural generalization is the budget-constrained version, where there is a specified budget, and the goal is to maximize the number of nodes influenced (as opposed to requiring 100% domination of the graph) with this budget. Related to this is a profit-maximization or prize-collecting variant of the PIDS problem. Shakarian et al. (2014) discuss such a variant in an application related to policing interventions to reduce gang violence. In the graph theoretical problem, in addition to the weight, each node also has a profit. Instead of minimizing the cost of a PIDS activating all nodes, the goal is to capture the tradeoff between the total weight and the ultimate profit resulting from the influence propagation process of a set of selected nodes. All of the variants discussed here are rich avenues for future research. Our work on the PIDS problem provides the first step along this research pathway.

References


Electronic Companion

EC.1. Algorithm for the PIDS Problem on Trees

In this section, we present a dynamic programming (DP) algorithm to solve the PIDS problem on trees. The DP algorithm decomposes the problem into subproblems, starting from the leaves of the tree. A subproblem is defined on a star network, which has a single central node and (possibly) multiple child nodes. For each star subproblem, the DP algorithm solves the PIDS problem for two cases. Consider the link that connects the star to the rest of the tree. We will refer to the node adjacent to the central node on this link as its parent. In the first case, the parent is not selected, whereas in the second case, the parent is selected. This process of solving star subproblems for two cases, followed by contraction of the star node, is repeated until we are left with a single star. The last star only requires the solution of one case; where the parent is not selected. After we exhaust all subproblems, a backtracking method is used to combine the solution candidates from the star subproblems and identify a final solution for the tree.

Algorithm 2 provides the pseudocode of the proposed algorithm. To create an ordering amongst the subproblems considered in the algorithm, it is convenient (but not necessary) to arbitrarily pick a root node (which we will denote by $r$). We will then prioritize the subproblems in order of how far their central nodes are from the root node of the tree (i.e., at every step among the remaining subproblems, we consider a subproblem whose central node is farthest from the root node). We call this a bottom-up traversal of the tree. This ordering can easily be determined a priori by conducting a breadth-first search (BFS) from the root node and considering the non-leaf nodes of the tree in reverse BFS order. The global variable $TC$ has the total cost of the optimal solution.

We now discuss how to solve the PIDS problem on a star. To better illustrate the algorithm, we consider the instance in Figure EC.1. Let $L$ be the set of all leaf nodes in the original tree. Let $c$ denote the central node of a star (all of the other nodes are child nodes) and refer to this star as star $c$. $L(c)$ denotes the set of all children of node $c$. There are two cases to consider. First, we consider the case where the parent of the central node $c$ is not selected in the optimal solution. Let

Algorithm 2 Algorithm for the PIDS problems on trees

1: Arbitrarily pick a node as the root node of the tree and let $TC = 0$.
2: Define the order of the star problems based on the bottom-up traversal of the tree.
3: Let $L$ be the set of leaf nodes in $G$ and $b_1^i = b_i$ and $b_2^i = 0$ for all $i$ in $L$.
4: for each star subproblem do
5: \hspace{1cm} StarHandling
6: end for
7: SolutionBacktrack
Figure EC.1 A PIDS problem instance.

$X_{NPS}^c$ represent the set of nodes selected in the solution to star $c$, and let $C_{NPS}^c$ denote the total cost of the solution for star $c$. Analogously, we consider the case where the parent of the central node $c$ is selected in the optimal solution with $X_{PS}^c$ representing the set of nodes selected in the solution to star $c$ and $C_{PS}^c$ denoting the total cost of the solution for star $c$.

In Figure EC.1, node 7 is selected as the root. Following the bottom-up ordering of the tree, we consider star 1, 2 and 3 first. Notice that all of these stars have their children in $L$. We will refer to stars where all children are members of $L$ as “bottom stars”. Because the influence diffusion process only takes place for one time period, if the parent of any node $i \in L$ is not selected, then node $i$ must be selected. This allows for a straightforward calculation to solve the bottom stars. Either the central node of the bottom star must be selected or all children in the bottom star must be selected. Specifically, for the case where the parent of the central node $c$ of a bottom star is not selected, we compare the cost of $b_c$ against the cost of $\sum_{j \in L(c)} b_j$. If $|L(c)| \geq g_c$ and $b_c > \sum_{j \in L(c)} b_j$, then all of the children in the bottom star are selected with $X_{NPS}^c = L(c)$ and $C_{NPS}^c = \sum_{j \in L(c)} b_j$. Otherwise, the central node $c$ of the bottom star is selected with $X_{NPS}^c = \{c\}$ and $C_{NPS}^c = b_c$.

While, for the case where the parent of the central node $c$ of a bottom star is selected, we compare the cost of $b_c$ against the cost of $\sum_{j \in L(c)} b_j$. If $b_c$ is greater, all of the children are selected with $X_{PS}^c = L(c)$ and $C_{PS}^c = \sum_{j \in L(c)} b_j$. Otherwise, the central node $c$ is selected with $X_{PS}^c = \{c\}$ and $C_{PS}^c = b_c$.

To illustrate, consider star 1. When node 1’s parent (node 6) is not selected in the optimal solution, we compare the cost of node 1 (with a payment of 3, which will also activate its children nodes 11 and 12) against the cost of selecting all of the leaf nodes of the star (i.e., nodes 11 and 12) for a cost of 2 units. Also, $|L(1)| > g_1$. Thus, the solution is $X_{NPS}^1 = \{11, 12\}$ and $C_{NPS}^1 = 2$. When node 1’s parent (node 6) is selected, we make the same comparison. Thus, the solution is $X_{PS}^1 = \{11, 12\}$ and $C_{PS}^1 = 2$.

Next, once a star’s solution candidates are determined, the star is contracted into a single child node for its parent’s star subproblem. It may appear that we have considered all possible solution
candidates $X_c^{NPS}$ and $X_c^{PS}$ for a given star $c$ in the optimal solution. However, that is not necessarily the case. Consider star 2. Here $X_2^{PS} = X_2^{NPS} = \{13, 14\}$. In both cases, the leaf nodes 13 and 14 are selected. Although star 2 does not need its central node 2 to be selected in either case (because influence only propagates to the neighbors of the selected nodes), star 5 may need node 2 to be selected in order to activate its central node 5. This may not be captured in the solutions $X_c^{NPS}$ and $X_c^{PS}$ computed so far for a given star. Hence, in addition to $C_c^{NPS}$ and $C_c^{PS}$, we also use the cost $b_c$ of the solution that selects the central node of star $c$. Notice that $C_c^{PS} \leq C_c^{NPS} \leq b_c$. Therefore, in the optimal solution, we must incur a cost of at least $C_c^{PS}$ for star $c$. This amount is added to the total cost $TC$. The remaining incremental amounts $b^1_c = C_c^{NPS} - C_c^{PS}$ and $b^2_c = b_c - C_c^{NPS}$ are computed and used to solve the next star subproblem. Thus, $b^1_c = 0$ and $b^2_c = 1$. It is similar to stars 2 and 3. In star 2, we have $X_2^{PS} = X_2^{NPS} = \{13, 14\}$, $C_2^{PS} = C_2^{NPS} = 2$ and $b^2_2 = 0$, $b^2_2 = 1$. In star 3, we have $X_3^{PS} = X_3^{NPS} = \{3\}$, $C_3^{PS} = C_3^{NPS} = 1$ and $b^3_3 = 0$, $b^3_3 = 0$. So far, $TC = 5$.

Unlike bottom stars, the star we consider now contains both contracted stars as leaf nodes (nodes 1, 2 and 3), as well as the leaf nodes in $L$ (nodes 8, 9 and 10). For convenience, we can also compute $b^1_i$ and $b^2_i$ for any leaf node $i \in L$. Take node 8 as an example. If its parent is not selected, node 8 must be selected with the cost 2. If its parent is selected, the cost is 0 because a leaf node requires one neighbor. Then, $b^2_8 = 2$, and $b^2_8 = 0$. Thus, $b^1_i = b_i$ and $b^2_i = 0$ for a leaf node $i$ in the original tree. After contracting stars 1, 2 and 3, we obtain a smaller tree and need to consider three stars, as shown in Figure EC.2(a).

We are now ready to discuss how to solve the PIDS problem on a star (earlier, our discussion was limited to solving the problem on the bottom stars; the ensuing discussion applies to all stars). Consider a star $c$ and the case where the parent of node $c$ is not selected in the optimal solution. We have two alternatives. Either we select the central node $c$ with cost $b_c$ to activate the entire star (if the central node $c$ is selected, all children $i \in \{L(c) \setminus L\}$ follow the solution $X_i^{PS}$ whose cost is already included in $TC$) or select a subset of nodes in $L(c)$ as cheaply as possible that activates the entire star.

We need to compute the cost of the alternative, where a minimum cost subset of the nodes in $L(c)$ are selected to activate the entire star. If the central node $c$ of the star is not selected, we must at least incur the cost $B_c^{NPS} = \sum_{i \in L(c)} b^1_i$, since all of the children $i \in L(c)$ must at least incur

\[ B_c^{NPS} = \sum_{i \in L(c)} b^1_i \]
the cost of the solution \( X^\text{NPS} \) when their parent is not selected. Then, we must select \( g_c \) nodes in \( L(c) \). Therefore, we sort nodes in \( L(c) \) in ascending order of their \( b^2_i \) values. The cost of the solution depends on the size of the set \( L(c) \): **Case 1** (|\( L(c) \)| < \( g_c \)): Node \( c \) must be selected. **Case 2** (|\( L(c) \)| ≥ \( g_c \)): We select the first \( g_c \) nodes in \( L(c) \) in ascending order of their \( b^2_i \) value (we denote this set as \( S_{g_c} \)) and total cost of \( B^\text{NPS}_c = \sum_{i \in S_{g_c}} b^2_i \). Comparing \( b_c \), the cost of selecting central node \( c \), against the cost of the solution just obtained provides us the solution to the PIDS on the given star.

Now, we consider star \( c \) and the case where the parent of node \( c \) is selected in the optimal solution. Again, we have two alternatives. Either we select the central node \( c \) with cost \( b_c \) to activate the entire star or select a subset of nodes in \( L(c) \) as cheaply as possible that activates the entire star. The cost of the alternative, where a minimum cost subset of nodes in \( L(c) \) are selected to activate the entire star, is calculated identically as the above two cases with the change such that \( g_c \) is updated to \( g_c - 1 \) (to account for the fact that star \( c \)'s parent has been selected), and is thus able to influence it.

Algorithm 3 provides the pseudocode associated with this calculation procedure. At its core is the function `SOLVESTAR` that finds the optimal solution for a given star. When the procedure is applied to star 6, we have \(|L(6)| = 2\) and \( B^\text{NPS}_6 = \sum_{i \in L(6)} b^2_i = 2\). For NoParent-Selected, \( X^\text{NPS}_6 = \{1, 8\} \), and \( C^\text{NPS}_6 = 3\). For Parent-Selected, \( X^\text{PS}_6 = \{8\} \), and \( C^\text{PS}_6 = 2\). Thus, \( TC = 7\). Contracting star 6 gives \( b^1_6 = 1 \) and \( b^2_6 = 1\). For star 5, \(|L(5)| = 2\). First, \( b_5 = 1 < 2 = B^\text{NPS}_5\). Thus, we select node 5. Then, \( X^\text{NPS}_5 = X^\text{PS}_5 = \{5\} \), and \( C^\text{NPS}_5 = C^\text{PS}_5 = 1\). Thus, \( TC = 8\). Contracting star 5 gives \( b^1_5 = 0 \) and \( b^2_5 = 0\). For star 4, \(|L(4)| = 2\). First, \( b_4 = 2 > 1 = B^\text{NPS}_4\). Thus, we consider two cases. For the case where no parent is selected, we select node 4. The cost is 2. Thus, \( X^\text{NPS}_4 = \{4\} \), and \( C^\text{NPS}_4 = 2\). For the case of parent selected, the solution of not selecting node 4 is to select node 3 and 10. The cost is 1. Then, \( X^\text{PS}_4 = \{3, 10\} \), and \( C^\text{PS}_4 = 1\). Thus, \( TC = 9\). Contracting star 7 gives \( b^1_7 = 1 \) and \( b^2_7 = 0\). Now, we only have one star left, as shown in Figure EC.2(b). Star 7 has \(|L(7)| = 3\) and \( B^\text{NPS}_7 = 2 < 4\). Then, we only need to consider the case of no parent selected. Thus, the solution of not selecting node 7 is to select nodes 4, 5 and 6. The cost is 3, which is a smaller cost than node 7. Thus, \( X^\text{NPS}_7 = \{4, 5, 6\} \), and \( C^\text{NPS}_7 = 3\). Thus, \( TC = 12\).

After we obtain the solution of the last star, which has the root node as its central node, we invoke a backtracking procedure to choose the solution from the candidates for each star subproblem and piece them together to obtain the final solution for this tree. Once the last star subproblem is solved, for each child node in this star, we know if it is selected or not and if its parent node is selected or not. For instance, if the central node is selected, all stars with the central node in \( \{L(c) \setminus L\} \) will pick the Parent-Selected candidate. Otherwise, first, if a node \( i \) in \( \{L(c) \setminus L\} \) is selected, we can proceed to the nodes in \( L(i) \) and pick the Parent-Selected candidate. Second, if
Algorithm 3 StarHandling

Require: star $c$.
1: $(X^N_{cPS}, C^N_{cPS}) \leftarrow \text{SolveStar}(\text{star } c, \text{NoParent-Selected}).$
2: if star $c$ is the last star then
3: \quad $TC = TC + C^N_{cPS}$
4: else
5: \quad $(X^N_{cPS}, C^N_{cPS}) \leftarrow \text{SolveStar}(\text{star } c, \text{Parent-Selected}).$
6: \quad The contracted node has $b^1_c = C^N_{cPS} - C^P_{cPS}$ and $b^2_c = b^c - C^N_{cPS}$.
7: \quad $TC = TC + C^P_{cPS}$
8: end if
9: function SolveStar(a star $c$, Flag)
10: \quad if Flag == Parent-Selected then
11: \quad \quad $g_c = g_c - 1.$
12: \quad end if
13: \quad $B^N_{cPS} = \sum_{i \in L(c)} b^1_i$ and let $S_{g_c}$ be the set of the cheapest $g_c$ nodes in $L(c)$ by $b^2$.
14: \quad if $|L(c)| < g_c$ then
15: \quad \quad $C = b_c$
16: \quad else
17: \quad \quad $C = \min \{ b_c, B^N_{cPS} + \sum_{i \in S_{g_c}} b^2_i \}.$
18: \quad end if
19: \quad if $C$ is $b_c$, let $X \leftarrow b.$
20: \quad if $C$ is $B^N_{cPS} + \sum_{i \in S_{g_c}} b^2_i$, let $X \leftarrow S_{g_c}$
21: \quad return $X, C.$
22: end function

![Figure EC.3](image-url) The solution obtained by our DP algorithm. Shaded nodes are selected in the PIDS.

A node $i$ in $\{L(c) \setminus L\}$ is not selected, star $i$ will pick the NoParent-Selected candidate. With this information, we can now proceed down the tree, incorporating the solution candidate at a node based on the solution of its parent star. This backtracking procedure is described in Algorithm 4 SolutionBacktrack. Let $r$ denote the root of the tree (as determined by Algorithm 2), and let a binary vector $x^*$ denote the selected nodes with a value of 1s. The final solution is in Figure EC.3. The nodes selected in the PIDS are shaded.

**Proposition EC.1.** The PIDS problem on trees can be solved in $O(|V|)$ time.
Algorithm 4 SolutionBacktrack

1: Let $x^* = 0$. Then, call PIEcing($r$, $x^*$, NoParent-Selected) for the root node $r$.
2: function PIEcing($c$, $x$, Flag)
3:     If Flag == ParentSelected, $X' = X^{PS}_c$. Otherwise, $X' = X^{NPS}_c$.
4:     $X ← X ∪ X'$ and $x_i = 1 ∀ i ∈ X'$.
5:     if $c ∈ X'$ then
6:         ∀ $j ∈ \{L(c) \backslash L\}$ call PIEcing($j$, $x$, Parent-Selected).
7:     else
8:         ∀ $i ∈ \{X' \backslash L\}$ call PIEcing($j$, $x$, NoParent-Selected).
9:         for $i ∈ \{X' \backslash L\}$ do
10:             ∀ $j ∈ \{L(i) \backslash L\}$ call PIEcing($j$, $x$, Parent-Selected).
11:         end for
12:     end if
13:     return $x$.
14: end function

Proof. Correctness of the algorithm can be established via induction, using identical arguments to the preceding discussion. We now discuss the running time. There are at most $|V|$ stars in a tree. For each star $c$, let $deg(c)$ denote its degree. We need to find the $g_c$ cheapest children, and it takes time $O(deg(c))$ (Finding the $g_c$th order statistics can be done in $O(deg(c))$ by the Quickselect method in Chapter 9 of Stein et al. (2009). Then, it takes $O(deg(c))$ to go through the list to collect the $g_c$ cheapest children.) For the whole tree, this is bounded by $O(|V|)$. In the backtracking procedure, we only pick the final solution, and it takes time $O(|V|)$. Therefore, the running time for the dynamic algorithm is linear with respect to the number of nodes. □

EC.2. Proof of Theorem 2

By relaxing the binary variables, the LP relaxation of BIP3 is referred to as LP3 and is given below:

\[
\text{Minimize } \sum_{i ∈ V} b_i x_i \text{  (EC.1)}
\]

Subject to: ($t_{ij}$) \[x_i - y_{ij} ≥ 0 \quad ∀ i ∈ V, j ∈ n(i) \text{  (EC.2)}\]

($u_{id}$) \[y_{id} - x_i ≥ 0 \quad ∀ i ∈ V, d ∈ a(i) \text{  (EC.3)}\]

($v_{id}$) \[-y_{id} - y_{di} = -1 \quad ∀ \{i, d\} ∈ E_t \text{  (EC.4)}\]

($w_i$) \[\sum_{d ∈ a(i)} y_{di} + g_i x_i ≥ g_i \quad ∀ i ∈ V \text{  (EC.5)}\]

\[x_i ≥ 0 \quad ∀ i ∈ V \text{  (EC.6)}\]

\[y_{id}, y_{di} ≥ 0 \quad ∀ \{i, d\} ∈ E_t \text{  (EC.7)}\]

The dual of LP3 is as follows:

\[
\text{Maximize } \sum_{i ∈ V} g_i w_i - \sum_{(t, d) ∈ E_t} v_{td} \text{  (EC.8)}
\]
We have $t$, $u$, $v$ and $w$ as dual variables for the constraint sets (EC.2), (EC.3), (EC.4), and (EC.5), respectively. We refer to the dual linear program as DLP3.

It should be clear that the solution of the DP algorithm in Section EC.1 provides a feasible solution to BIP3, and thus LP3. Recall that $a(i)$ is the set of node $i$’s neighbors in $G_t$, and $n(i)$ is that in $G$. For $x$ variables, if a node $i$ is in the PIDS, $x_i = 1$. Otherwise, $x_i = 0$. To obtain $y$ variables’ values, if $x_i = 1$, set $y_{id} = 1$ for all $d$ in $a(i)$. Then, for all $j$ in $n(i)$, if $x_j = 0$, set $y_{dj} = 1$, where $d$ is the dummy node inserted in between node $i$ and node $j$. For the remaining undecided edges $\{i,d\}$, we set $y_{id} = 1$ and $y_{di} = 0$. Thus, we obtain a feasible solution for LP3 based on the solution returned by the DP algorithm. In this proof, we show that we can construct a dual feasible solution to DLP3, and this pair of primal and dual solutions satisfies the complementary slackness (CS) conditions as follows:

\[
(u_{id} - v_{id})y_{id} = 0 \quad \forall i \in V, d \in a(i) \quad (EC.15)
\]
\[
(x_i - y_{di})t_{ij} = 0 \quad \forall i \in V, j \in n(i) \quad (EC.16)
\]
\[
(y_{id} - x_i)u_{id} = 0 \quad \forall i \in V, d \in a(i) \quad (EC.17)
\]
\[
g_i - \sum_{d \in n(i)} y_{di} - g_i x_i \quad w_i = 0 \quad \forall i \in V \quad (EC.18)
\]
\[
(b_i - \sum_{j \in n(i)} t_{ij} + \sum_{d \in a(i)} u_{id} - g_i w_i) x_i = 0 \quad \forall i \in V \quad (EC.19)
\]
\[
(-t_{ji} - v_{id} + w_i) y_{di} = 0 \quad \forall d \in D, i \in a(d) \quad (EC.20)
\]

First of all, we always have $u_{id} = v_{id}$ for all $\{i,d\}$ in $E_t$ to satisfy the dual constraint (EC.10) and CS condition (EC.15). Second, in DLP3, only $t$ variables interact between two nodes in $V$. If
we fix their values first, we can isolate each node $i$ in $V$ and assign values to the corresponding $u_{id}$, $v_{id}$ and $w_i$ variables. Following the bottom-up order in the execution of the DP algorithm in Section EC.1, we first assign values for all $t$ variables. Starting from the bottom of the tree, let node $i$ be the current node and node $h$ be its parent node in the original tree $G$. Recall that $b_1^i$, $b_2^i$, $X_i^{NPS}$, $C_i^{NPS}$, $X_i^{PS}$, and $C_i^{PS}$ are obtained in the DP in Section EC.1. We set $t_{ih} = b_2^i$ and $t_{hi} = b_1^i$, as shown in Figure EC.4(a). For condition (EC.16), it requires $t_{ih} = 0$ when $x_i = 1$ and $y_{dh} = 0$. It means that the corresponding $x_h = 1$. Given that node $h$ and node $i$ are both in PIDS, it implies that $C_i^{PS} = b_i$. Thus, $b_1^i = b_2^i = 0$. Therefore, $t_{ih} = 0$ and $t_{hi} = 0$, as shown in Figure EC.4(b). For other situations, we have $x_i = y_{dh}$. Thus, condition (EC.16) is satisfied. Consequently, we can focus on the remaining CS conditions (EC.17), (EC.18), (EC.19), and (EC.20).

Now, three cases are considered to assign the associated dual variables for a node $i$ in $V$. All $u$, $v$ and $w$ variables are initialized as zeros. Then, in the following proof, we only change those variables that need to be non-zeros.

Case 1: Suppose that node $i$ is a leaf node and node $h$ is its parent node in $G$. It means that $x_i = y_{id}$, as shown in Figure EC.5(a). Also, $t_{ih} = 0$ and $t_{hi} = b_i$ because $b_1^i = b_i$ and $b_2^i = 0$. Set $w_i = b_i$. All primal and dual constraints are binding for conditions (EC.17), (EC.18), (EC.19), and (EC.20). Thus, they are satisfied.

Next, we consider the non-leaf nodes in $G$. There are two cases for them.

Case 2: Suppose that node $i$ is not a leaf node in $G$ and $x_i = 0$, as shown in Figure EC.5(b). Let $S_i^d = \{j \in n(i) : x_j = 1\}$, which denotes the set of nodes that are selected and adjacent to node $i$ in the original graph, and $S_i^d = \{d \in a(i) \cap a(j) : j \in S_i^d\}$, which denotes the set of dummy nodes adjacent to node $i$ and the nodes in $S_i^d$. Then, let $w_i = \max\{t_{ji} : j \in S_i^d\}$, which is the biggest $t_{ji}$ value among the nodes in $S_i^d$. Then, let $u_{id} = v_{id} = w_i - t_{ji}$ for all $j$ in $S_i^d$ and $d$ in $S_i^d$. Condition (EC.17) is satisfied because $y_{id} = x_i = 0$ for all $d$ in $S_i^d$ and $u_{id}$ are zero for all $d$ in $a(i) \setminus S_i^d$. When there are exactly $g_i$ incoming arcs, constraint (EC.5) is binding. When there are more than $g_i$ incoming arcs, $w_i = 0$ because only nodes with zero $t_{ji}$ are included in $S_i^d$. When $|S_i^d| > g_i$, nodes with
positive \( t_{ji} \) can be removed from \( S_i^d \) to obtained a better solution. A contradiction exists here. Thus, condition (EC.18) is satisfied. Recall that \( B_i^{NPS} = \sum_{L(i)} b_j^1 \). In constraint (EC.9), its left-hand side is:

\[
B_i^{NPS} + b_i^2 - \sum_{j \in S_i^d} (w_i - t_{ji}) + g_i w_i \quad \text{(Note: } \sum_{j \in n(i)} t_{ij} = B_i^{NPS} + b_i^2, u_{id} = w_i - t_{ji})
\]

\[
\begin{cases}
B_i^{NPS} + \sum_{j \in X_i^{NPS}} b_j^2 + b_i^2 - (|S_i^d| - g_i) w_i = C_i^{NPS} + b_i^1 + (|S_i^d| - g_i) w_i, \text{ if } h \notin S_i^d \quad \text{(Note: } S_i^d = X_i^{NPS}) \\
B_i^{NPS} + \sum_{j \in X_i^{NPS}} b_j^2 + b_i^1 + b_i^2 - (|S_i^d| - g_i) w_i = C_i^{NPS} + b_i^1 + b_i^2, \text{ if } h \in S_i^d \quad \text{(Note: } S_i^d = X_i^{PS} \cup \{h\})
\end{cases}
\]

\[
\begin{cases}
\begin{array}{l}
b_i \quad \text{(Note: } |S_i^d| = g_i \text{ or } w_i = 0 \text{ when } |S_i^d| > g_i) \\
b_i \quad \text{(Note: } b_i = C_i^{NPS} + b_i^1 + b_i^2 \text{ based on the definitions of } b_i^1 \text{ and } b_i^2) = b_i.
\end{array}
\end{cases}
\]

Thus, condition (EC.19) is satisfied because constraint (EC.9) is respected and \( x_i = 0 \). Condition (EC.20) is satisfied because constraint (EC.11) is binding for all \( d \) in \( S_i^d \) and \( y_{di} = 0 \) for all \( d \) in \( a(i) \setminus S_i^d \).

Case 3: Suppose that node \( i \) is not a leaf node in \( G \) and \( x_i = 1 \). It means that \( y_{i id} = 1 \) and \( y_{di} = 0 \) for all \( d \) in \( a(i) \). Then, CS conditions (EC.17), and (EC.18) are satisfied because those corresponding primal constraints are binding. Because \( x_i = 1 \), constraint (EC.9) must be binding. Let LHS denote the value of the left-hand side of constraint (EC.9). So far, LHS = \( \sum_{j \in n(i)} t_{ij} = B_i^{NPS} + b_i^2 \). If LHS = \( b_i \), we are done. Otherwise, the idea of the following proof is to show that a dual solution can be first constructed to ensure that LHS \( \geq b_i \), and then the dual solution can be adjusted to have LHS = \( b_i \). Recall that \( X_i^{PS} \) is the Parent-Selected solution for node \( i \) in the DP. Based on \( X_i^{PS} \), we consider two situations. First, suppose that \( X_i^{PS} \neq \{i\} \) and \( |X_i^{PS}| = g_i - 1 \), as shown in Figure EC.6(a). It implies that \( X_i^{PS} \neq X_i^{NPS} \) because \( X_i^{PS} \) is \( \{i\} \) or has \( |X_i^{NPS}| = g_i \). Set \( w_i \) as the smallest \( t_{ji} \) value for all \( j \) in \( n(i) \). Then, set \( u_{d id} = v_{id} = \max \{w_i - t_{ji}, 0\} \) for all \( d \) in \( a(i) \).

Thus,

\[
- \sum_{d \in a(i)} u_{d id} + g_i w_i = \begin{cases}
\sum_{j \in X_i^{NPS}} b_j^2 & \text{if } X_i^{PS} \neq \{i\} \\
\sum_{j \in X_i^{PS}} b_j^2 + b_i^1 & \text{if } X_i^{PS} = \{i\}
\end{cases}
\]

Then,

\[
\text{LHS} = \begin{cases}
B_i^{NPS} + b_i^2 + \sum_{j \in X_i^{NPS}} b_j^2 & \text{if } X_i^{PS} \neq \{i\} \\
B_i^{NPS} + b_i^2 + \sum_{j \in X_i^{PS}} b_j^2 + b_i^1 & \text{if } X_i^{PS} = \{i\}
\end{cases} = C_i^{NPS} + b_i^2 = b_i
\]
On the right part of Figure EC.7, it has non-zero dual variables, except for variable \( v \). Thus, for Case 1, we have nodes 8, 9, 10, 11, 12, 13, 14, 15, and 16. Thus, set \( w_8 = 2, w_9 = 2, w_{10} = w_{11} = w_{12} = w_{13} = w_{14} = w_{15} = w_{16} = 1 \). For Case 2, we have nodes 1, 2, and 7. For node 1, \( |S_1^i| = 3 \). Thus, \( w_1 = 0 \). For node 2, \( |S_2^i| = 3 \). Thus, \( w_2 = 0 \). For node 7, \( |S_7^i| = 3 = g_7 \). Thus, \( w_7 = \max\{t_{4,7}, t_{5,7}, t_{6,7}\} = 1 \), and \( u_{7,m} = 0, u_{7,n} = u_{7,o} = 1 \). For Case 3, we have nodes 3, 4, 5, and 6. For node 3, \( \sum_{j \in n(3)} t_{3,j} = 2 = b_3 \). For node 4, \( \sum_{j \in n(4)} t_{4,j} = 1 < 2 = b_4 \). Set \( w_4 = 1 = t_{7,4}, w_{4,k} = 1 = t_{3,4} \). For node 5, \( \sum_{j \in n(5)} t_{5,j} = 2 > b_5 \). Set \( u_{5,i} = 1 \). For node 6, \( \sum_{j \in n(6)} t_{6,j} = 3 < 4 = b_6 \). Set \( w_6 = 1 \). The total objective value is 12, which is exactly the same as that of the solution obtained by the DP.