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

INFORMS JOURNAL ON COMPUTING
Vol. 34, No. 3, May–June 2022, pp. 1345–1365
ISSN 1091-9856 (print), ISSN 1526-5528 (online)

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

S. Raghavan,[a] Rui Zhang[b,*]

[a] Robert H. Smith School of Business & Institute for Systems Research, University of Maryland, College Park, Maryland 20742; [b] Leeds School of Business, University of Colorado, Boulder, Colorado 80309
*Corresponding author

Contact: raghavan@umd.edu, https://orcid.org/0000-0002-9656-5596 (SR); rui.zhang@colorado.edu, https://orcid.org/0000-0002-4029-6585 (RZ)

Received: December 7, 2020
Revised: July 10, 2021; October 1, 2021
Accepted: October 9, 2021
Published Online in Articles in Advance: February 8, 2022

https://doi.org/10.1287/ijoc.2021.1144

Abstract. 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 with a set V of nodes and a set E of edges, 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 not in T is adjacent to at least g_i members of T. When g_i is one 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-dening 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.

Summary of Contribution: In influence maximization problems, a decision maker wants to target individuals strategically to cause a cascade at a minimum cost over a social network. These problems have attracted significant attention as their applications can be found in many different domains including epidemiology, healthcare, marketing, and politics. However, computationally solving large-scale influence maximization problems to near optimality remains a substantial challenge for the computing community, which thus represent significant opportunities for the development of operations-research based models, algorithms, and analysis in this interface. This paper studies the positive influence dominating set (PIDS) problem, an influence maximization problem on social networks that generalizes the celebrated dominating set problem. It focuses on developing exact methods for solving large instances to near optimality. In other words, the approach results in strong bounds, which then provide meaningful comparative benchmarks for heuristic approaches. The paper first shows that straightforward generalizations of well-known formulations for the dominating set problem do not yield strong (i.e., computationally viable) formulations for the PIDS problem. It then strengthens these formulations by proposing a compact extended formulation and derives its projection onto the space on the natural node-selection variables, resulting in two equivalent (stronger) formulations for the PIDS problem. The projected formulation on the natural node-variables contains a new class of valid inequalities that are shown to be facet-defining for the PIDS problem. These theoretical results are complemented by in-depth computational experiments using a branch-and-cut framework, on a testbed of 100 real-world graph instances, with up to approximately 2.5 million nodes and 8 million edges. They demonstrate the effectiveness of the proposed formulation in solving large scale problems finding solutions that are on average 0.2% from optimality and solving 51 of the 100 instances to optimality.

History: Accepted by David L. Alderson, Area Editor for Network Optimization: Algorithms & Applications.
Supplemental Material: The e-companion is available at https://doi.org/10.1287/ijoc.2021.1144.

Keywords: 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 reached more than 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).

Kempe et al. (2003) initiated the study of influence maximization problems as optimization problems. In a probabilistic setting, they considered a budgeted version of the problem (i.e., given a budget k, identify the k individuals to directly target so as to maximize the number of nodes influenced in the social network). Their seminal work has inspired a large research community studying influence maximization problems and their variants. One stream of work, initiated by Chen (2009), has focused on influence maximization problems in a deterministic setting with a cost minimization, which is 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 one and the degree of the node, denoted by $deg(i)$. All nodes are inactive initially. A selected subset of nodes, the target set, are activated (i.e., switched to an active state or influenced). 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 may cost the decision maker differing amounts 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 (more than 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 difference between the magnitude of direct influence and indirect influence is in the orders of magnitude.

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 one and its degree $deg(i)$; 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 $\sum_{i \in T} b_i$). 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).

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 with 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 2020), 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.

In a seminal paper, Kempe et al. (2003) initiated the study of influence maximization problems as network optimization problems. They proposed a general class of influence propagation models that includes both the linear threshold (in this model a node i is influenced/activated if the sum of the influences from it's active neighbors exceeds node i's threshold) and independent cascade (in this model a freshly activated node i has a one-time probability p_{ij} to activate an inactive neighbor node j) models. In a probabilistic setting, given a budget of k seed products, they try to identify the k individuals to directly target to maximize the number of nodes activated in the social network. They show that it is NP-hard to find the optimal seed set. Relying on the submodularity property of the objective function (which is because of the particular nature of the probabilistic assumptions on the threshold values), they developed a $(1-1/e)$-approximation algorithm for the problem. Chen (2009) initiated another stream of work by focusing on influence maximization problems in a deterministic setting. They proposed the TSS problem, which has a cost minimization aspect (i.e., instead of the decision maker being given a budget k, the desire is to find the minimum number of nodes to target in the network so that the entire network is influenced), and the influence propagation follows a linear threshold model. However, the submodularity property is no longer possessed in a deterministic setting. In fact, Chen (2009) shows that the TSS problem is APX-hard. The comprehensive reviews by Chen et al. (2013), Li et al. (2018), and Banerjee et al. (2020) nicely summarize the most relevant work (that has appeared in the computer science literature) on this topic.

More recently, there are a stream of papers in the operations research literature focused on developing optimization models and solution algorithms that explicitly accounts for differences in the individual behavior of the nodes and the influence propagation process in the network. Wu and Küçükyavuz (2018), Güney (2019a, b), and Ghayour-Baghbani et al. (2021) focus on influence maximization problems in a probabilistic setting from a maximization perspective. It is worth noting that Güney (2019a) shows that the randomized algorithm based on their proposed Linear Programming (LP) formulation has a constant worst-case bound, which is as good as the current state of the art $(1-1/e)$. Güney et al. (2021) present a branch-and-cut algorithm for the influence maximization problem under the independent cascade model by modeling it as a stochastic maximal covering location problem. There are a growing number of papers on the deterministic side as well, all focusing on the cost minimization aspect. Raghavan and Zhang (2019, 2021b) study the WTSS problem. Günneç et al. (2020a, b) study the least cost influence problem (LCIP), which broadens the scope of the WTSS problem by allowing for partial payments to nodes that are not directly targeted. The LCIP seeks to minimize the sum of the costs of direct targeting and partial payments provided to influence the entire network. Fischetti et al. (2018) further generalized the LCIP to allow for a nonlinear influence structure. They proposed a novel set covering based formulation, which has both an exponential number of variables and an exponential number of constraints. Using this formulation, they described an exact approach and a heuristic approach on arbitrary graphs.

The PIDS problem falls in the category of influence maximization problems in a deterministic setting with a cost minimization aspect. It 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 δ 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 LP 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 ρ 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 (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. Baïou 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. Furthermore, it generally requires $\rho = 0.5$ or the value of ρ 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. We also show (in Section EC.2.1 of the e-companion) 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 of the e-companion). Although the extended formulation is strong, it needs artificial variables defined on the edge space. Thus, in Section 3, we project the extended formulation 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 testbed of 100 real-world graph instances, our approach finds solutions that are on average 0.2% from optimality and solves 51 of the 100 instances to optimality. Finally, Section 6 provides some concluding remarks.

Table 1 compares our contributions in this paper to our previous work on the WTSS problem (Raghavan and Zhang 2019, 2021b) and the LCIP (Günneç et al. 2020a, b). The first two columns focus on problem characteristics. Both the LCIP and WTSS problem allow indirect and direct influence, whereas the PIDS problem only considers direct influence. Although the LCIP allows for partial payments to nodes that are not directly targeted, the WTSS and PIDS problem do not. The high-level strategy applied to these three problems is somewhat similar in the sense that we derive useful results and insights from the special case of a tree graph, which turns out to be polynomial solvable, and then apply these insights to arbitrary graphs where the problem is NP-hard. However, the execution of this strategy and its manifestations result in quite distinct outcomes and technical proofs. For the LCIP on trees, we provide a greedy algorithm that

Table 1. Comparison Between the Current Paper and the Authors' Previous Works on the WTSS Problem and the LCIP

	Indirect influence	Partial payment	Formulation ideas	Projection	Facet-defining conditions	Trees	Cycles
WTSS	Yes	No	Edge splitting; acyclic propagation graph	Yes, valid only on trees	No	DP; polytope	DP; polytope
LCIP	Yes	Yes	Influence type; acyclic propagation graph	No	No	DP; greedy	No
PIDS	No	No	Edge splitting; propagation limited to immediate neighbors	Yes, valid on AG	Yes	DP; polytope	No

Note. DP, dynamic programming; AG, arbitrary graphs.

solves the problem in polynomial time. We also use a dynamic programming (DP) framework to provide linear time algorithms to the solve the LCIP, PIDS, and WTSS problems on trees. Although conceptually the dynamic programming paradigm is similar, its application results in quite different algorithms (as the resulting subproblems that need to be solved are quite different). The approach to derive strong (in fact integral) formulations on trees and extending them to arbitrary graphs varies. In the LCIP the idea is to categorize the influence types propagated on the arcs of the network. On the other hand, in the WTSS and PIDS problems, the main idea is edge-splitting to obtain an extended formulation, which is then projected onto the space of the natural node variables. Although the edge-splitting idea is similar, the resulting formulations are quite different from each other, leading to distinct technical proofs (and computational results on arbitrary graphs). For example, the PIDS problem extended formulation requires constraints to ensure that no indirect influence is propagated in the graph, which is not necessary for the (LCIP or) WTSS problem. We project the extended formulations onto the natural node-selection variable space for both the PIDS and WTSS problem to obtain their polytope on trees. In contrast to the PIDS problem, the projected formulation for the WTSS problem is not valid on arbitrary graphs. Formulations for the LCIP and WTSS problem on arbitrary graphs take a different approach than the PIDS problem, where the influence categorization (for the LCIP) and the edge-splitting (for the WTSS problem) formulations are augmented with a set of constraints that ensure that the influence propagation network is acyclic. Necessary and sufficient facet-defining conditions for the set of projected valid inequalities are only derived for the PIDS problem. Finally, note that, for the WTSS problem, we are able to build upon the polytope for the tree case to derive the polytope for the cycle case.

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.

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 $x_1 = 1$ if node i is in the PIDS and is zero otherwise. Let $n(i)$ denote the set of node i's neighbors.

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

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

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

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 zero.

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

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

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

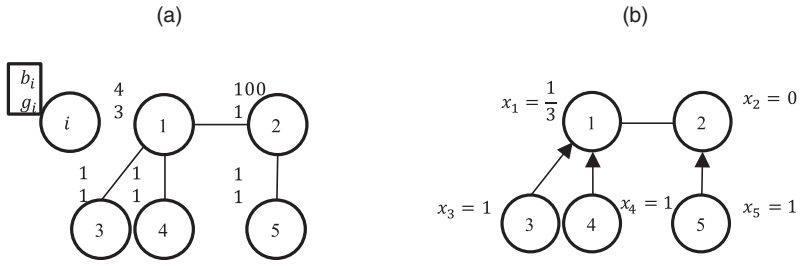
$$x_i, y_{ij} \in \{0,1\} \quad \forall i \in V, j \in n(i) \tag{7}$$

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 one 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 of the e-companion), 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

Figure 1. An Example Showing That LP1 and LP2 Obtain Fraction Solutions on Trees



Notes. (a) A PIDS instance. (b) A fractional solution returned by LP1 and LP2.

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. Because 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 (b). 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. This edge-splitting idea has also been applied to obtain a stronger formulation for the WTSS problem (Raghavan and Zhang 2019, 2021b).

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 one if node i is selected in the PIDS and zero 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), we 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 one and zero 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.

$$\text{(BIP3)} \quad \text{Minimize} \quad \sum_{i \in V} b_i x_i \tag{8}$$

$$\text{Subject to :} \quad x_i \geq y_{dj} \qquad \forall i \in V, j \in n(i) \tag{9}$$

$$x_i \leq y_{id} \qquad \forall i \in V, d \in a(i) \tag{10}$$

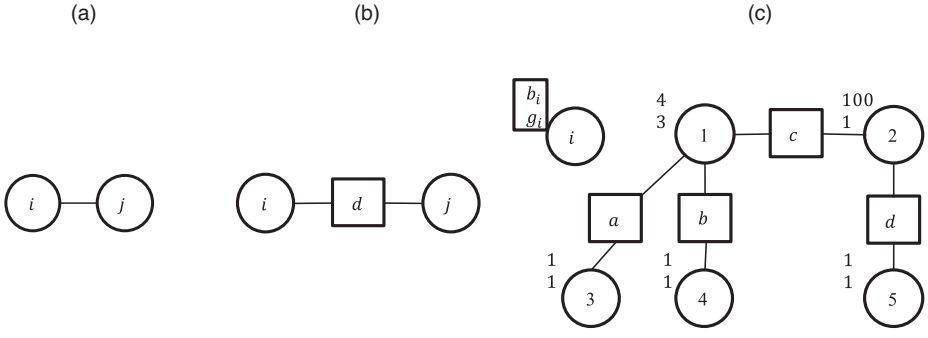
$$y_{id} + y_{di} = 1 \qquad \forall \{i,d\} \in E_t \tag{11}$$

$$\sum_{d \in a(i)} y_{di} + g_i x_i \geq g_i \qquad \forall i \in V \tag{12}$$

$$x_i \in \{0,1\} \qquad \forall i \in V \tag{13}$$

$$y_{id}, y_{di} \in \{0,1\} \qquad \forall \{i,d\} \in E_t \tag{14}$$

Figure 2. Illustration of the Graph Transformation Procedure



Notes. (a) An original edge. (b) A transformed edge. (c) Transformed graph of Figure 1(a).

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 zero. 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 is not selected in the PIDS. This is repeated for every node selected in the PIDS. Thus, Constraints (9) and (10) are satisfied. In the procedure described, at most, one of y_{id} and y_{di} has a nonzero 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 one in the procedure described. Last, 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. $z_{LP1} = z_{LP2} \le z_{LP3}$.

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 $\sum_{j\in n(i)} y^*_{ji} + g_i x^*_i \ge g_i$ because x^* satisfies Constraint (2) ($\sum_{j\in n(i)} x^*_j + g_i x^*_i \ge g_i$). 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 counterexample. Recall that $x_1 = \frac{1}{3}$, $x_2 = 0$, $x_3 = 1$, $x_4 = 1$, and $x_5 = 1$ are feasible and optimal solutions 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 $x_1 = 1$, $x_2 = 0$, $x_3 = 0$, $x_4 = 0$, $x_5 = 1$, $y_{1a} = 1$, $y_{1b} = 1$, $y_{1c} = 1$, $y_{c2} = 1$, $y_{a3} = 1$, $y_{a4} = 1$, $y_{2d} = 1$, $y_{5d} = 1$, and the remaining y variables are zeros with an objective value of five. □

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 of the e-companion, a dual feasible solution for the dual problem to LP3, and shows that these solutions satisfy the complementary slackness (CS) conditions. Let conv(X) denote the convex hull of the feasible PIDS vectors $\mathbf{x}$, 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* $\text{Proj}_{\mathbf{x}}(EPIDS) = \text{conv}(X)$.

Proof. Included in Section EC.2.1 of the e-companion. □

3. 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 with 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_{extended}.

$$\text{Minimize} \quad \sum_{i\in V} b_i x_i \tag{15}$$

$$\text{Subject to}: \quad -y_{di} + x_j \ge 0 \qquad \forall j \in V,\ i \in n(j) \tag{16}$$

$$-y_{di} - x_i \ge -1 \qquad \forall i \in V, d \in a(i) \tag{17}$$

$$\sum_{d\in a(i)} y_{di} + g_i x_i \ge g_i \qquad \forall i \in V \tag{18}$$

$$0 \le x_i \le 1, y_{di} \ge 0 \qquad \forall i \in V,\ d \in a(i) \tag{19}$$

We define a projection cone W, described by $(\mathbf{w}, \mathbf{u}, \mathbf{v})$, which satisfies the following linear inequalities.

$$w_i - u_{id} - v_{id} \le 0 \qquad \forall i \in V,\ d \in a(i) \tag{20}$$

$$w_i \ge 0, u_{id} \ge 0, v_{id} \ge 0 \qquad \forall i \in V,\ d \in a(i) \tag{21}$$

Here, u_{id}, v_{id}, and w_i are dual multipliers corresponding to Constraints (16), (17), and (18), respectively. If P_{extended} is written in matrix notation as $\{(\mathbf{x}, \mathbf{y}) : A\mathbf{x} + G\mathbf{y} \ge \mathbf{b}, (\mathbf{x}, \mathbf{y}) \ge \mathbf{0}\}$, based on Balas and Pulleyblank (1983), then any feasible vector $(\mathbf{w}, \mathbf{u}, \mathbf{v})$ to W defines a valid inequality: $(\mathbf{w},\mathbf{u},\mathbf{v})^T A\mathbf{x} \ge (\mathbf{w},\mathbf{u},\mathbf{v})^T \mathbf{b}$ to the projection of P_{extended} (in the space of the node-selection (x) variables). Furthermore, the projection of P_{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 $R(\mathbf{r})$ of all nonnegative multipliers of some $\mathbf{r} \in C$, called the direction (vector) of $R(\mathbf{r})$. A vector $\mathbf{r} \in C$ is extreme, if for any $\mathbf{r}^1, \mathbf{r}^2 \in C$, $\mathbf{r} = \frac{1}{2}(\mathbf{r}^1 + \mathbf{r}^2)$ implies $\mathbf{r}^1, \mathbf{r}^2 \in R(\mathbf{r})$. A ray $R(\mathbf{r})$ is extreme if its direction vector $\mathbf{r}$ is extreme.

Theorem 3. *The vector $\mathbf{r} = (\mathbf{w}, \mathbf{u}, \mathbf{v}) \in W$ is extreme if and only if there exists a positive α 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 zero.

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

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 zero.

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

Case 2 is similar to Case 1.

For Case 3, let $\mathbf{r} \in W$ be of the form Case 3 and assume that $\mathbf{r} = \frac{1}{2}(\mathbf{r}^1 + \mathbf{r}^2)$ for some $\mathbf{r}^1, \mathbf{r}^2 \in W$. Thus, for any component of $\mathbf{r}$ with the value 0, its corresponding components in $\mathbf{r}^1$ and $\mathbf{r}^2$ are also zero. Given i and d, let p^l_{id}, $l = 1, 2$, represent the positive element between u^l_{id} and v^l_{id}, $l = 1, 2$ (because only one of the two can be positive in the three cases). Then, we have $w^1_i + w^2_i = 2\alpha$ and $p^1_{id} + p^2_{id} = 2\alpha$, for all $d \in a(i)$. For a pair d_1 and d_2, we have $p^1_{id_1} > p^1_{id_2}$ if and only if $p^2_{id_1} < p^2_{id_1}$. However, Constraint (20) stipulates that $w^l_i \le \min\{p^l_{id_1}, p^l_{id_2}\}$, $l = 1, 2$. Hence, $p^l_{id_1} = p^l_{id_2} = \alpha_l$, $l = 1, 2$, for all $d_1, d_2 \in a(i)$. Otherwise, either Constraint (20) would be violated, or we would have $w^1_i + w^2_i < 2\alpha$. Therefore, $\mathbf{r}^1$, $\mathbf{r}^2$ are in $R(\mathbf{r})$. Thus, $\mathbf{r}$ is extreme.

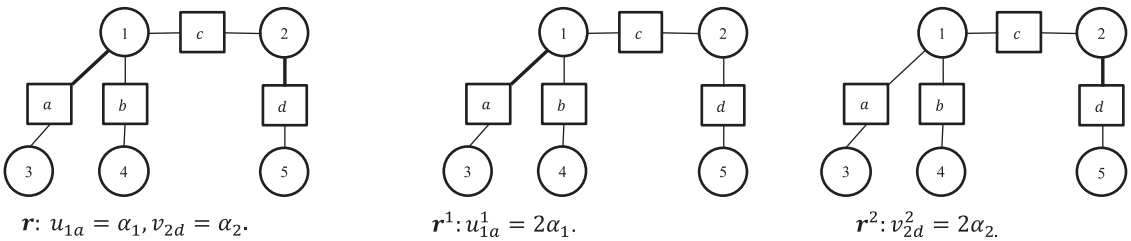
Necessity. Let $\mathbf{r}$ be an extreme vector of W. Let $S^w = \{i \in V : w_i > 0\}$, $S^u = \{\{i, d\} \in E_t : u_{id} > 0\}$ and $S^v = \{\{i, d\} \in E_t : v_{id} > 0\}$ based on this $\mathbf{r}$. In the following proof, to prove that a given ray $\mathbf{r}$ is not an extreme one, we construct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

Now consider the case when $S^w \ne \emptyset$. Suppose that $|S^w| > 1$; without loss of generality, let $i \in S^w$. Then, $\mathbf{r}^1$ has the value $w^1_i = 2w_i$ and $u^1_{id} = 2u_{id}$, $v^1_{id} = 2v_{id}$ for all $d \in a(i)$ and zeros for the other components. Thus, when $|S^w| > 1$, $\mathbf{r}$ is not extreme. Figure 4 illustrates this situation. The shaded nodes represent the positive w components in a vector $\mathbf{r}$.

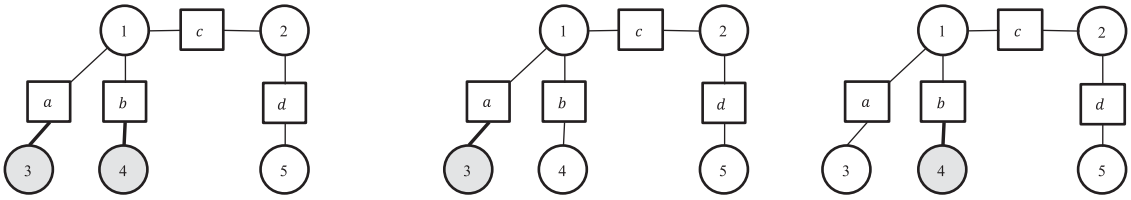
Suppose that $|S^w| = 1$ and $i \in S^w$; define $S_j = \{\{j, d\} \in E_t : p_{jd} > 0 \,\&\, j \in V \backslash \{i\}\}$. Thus, S_j contains edges with positive u or v components that are not adjacent to node i. When $S_j \ne \emptyset$, let $\mathbf{r}^1$ have $w^1_i = 2w_i$ and $u^1_{id} =$

Figure 3. $S^w = \emptyset$ and $|S^u| + |S^v| > 1$



Note. Figure shows that if $S^w = \emptyset$, $\mathbf{r}$ must be of the form Case 1 or Case 2.

Figure 4. When $|S^w| > 1$, $\mathbf{r}$ Is Not Extreme



$\boldsymbol{r}: w_3 = \alpha_1, u_{3a} = \alpha_2$
$w_4 = \alpha_3, v_{4b} = \alpha_4.$

$\boldsymbol{r^1}: w_3^1 = 2\alpha_1, u_{3a}^1 = 2\alpha_2.$

$\boldsymbol{r^2}: w_4^2 = 2\alpha_3, v_{4b}^2 = 2\alpha_4.$

$2u_{id}$, $v_{id}^1 = 2v_{id}$ for all $d \in a(i)$ and zeros in the other components. Then, $\mathbf{r^2}$ contains the positive components in S_j, whereas $\mathbf{r^1}$ does not. Thus, when $|S^w| = 1$ and $S_j \neq \emptyset$, $\mathbf{r}$ is not extreme. Figure 5 illustrates this situation with $S_j = \{\{1, a\}\}$.

Suppose that $|S^w| = 1$ and $i \in S^w$; define $S_1 = \{\{i,d\} \in E_t : u_{id} > 0 \oplus v_{id} > 0\}$, where only one of the u and v variables associated with an edge $\{i, d\}$ is positive and $S_2 = \{\{i,d\} \in E_t : u_{id} > 0 \,\&\, v_{id} > 0\}$, 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 $\mathbf{r^1}$ 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 $v_{id}^1 = \theta^1$. The remaining components are zero. Then, $\mathbf{r^1}$ does not have any edges that have both u and v variables 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 $|S_1| = deg(i)$. Otherwise, Constraint (20) would not be respected. Figure 6 illustrates this situation. Here, $S_2 = \{\{3, a\}\}$ and α_1 is less than α_2 and α_3.

Suppose that $|S^w| = 1$ and $|S_1| = deg(i)$; 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 $u_{id}^1 = 2(u_{id} - \alpha)$ and zeros in the other components. Then, $\mathbf{r^1}$ has only one positive component that is not w_i. Thus, when $|S^w| = 1$, $|S_1| = deg(i)$ 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 β is a positive value.

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

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 inequalities in the original graph G:

$$(g_i - q)x_i + \sum_{j \in S} x_j \geq g_i - q \quad \forall i \in V, \quad q = 0, 1, \ldots, deg(i), S \in C_i^{deg(i)-q}. \tag{22}$$

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

$$g_i x_i + \sum_{j \in n(i)} x_j \geq g_i \qquad \forall i \in V \tag{23}$$

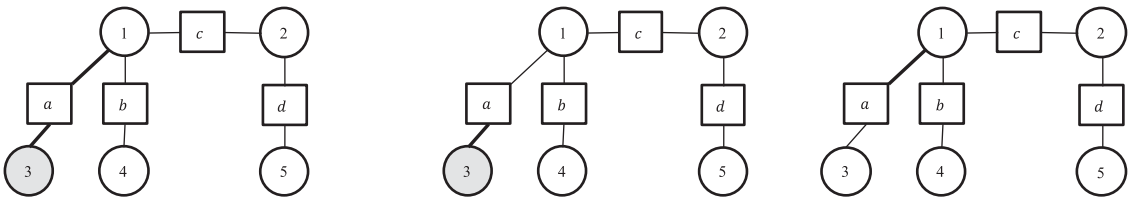
$$(g_i - q)x_i + \sum_{j \in S} x_j \geq g_i - q \qquad \forall i \in V,$$

$$q = 1, 2, \ldots, g_i - 1, S \in C_i^{deg(i)-q} \tag{24}$$

$$0 \leq x_i \leq 1 \qquad \forall i \in V \tag{25}$$

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 $q = 1$, $C_1^{3-1=2} = \{\{2,3\}, \{2,4\}, \{3,4\}\}$. When $q = 2$, $C_1^{3-2=1} = \{\{2\}, \{3\}, \{4\}\}$. Except for

Figure 5. When $|S^w| = 1$ and $S_j \neq \emptyset$ (Some us and vs Not Adjacent to S Have a Positive Value), $\mathbf{r}$ Is Not Extreme

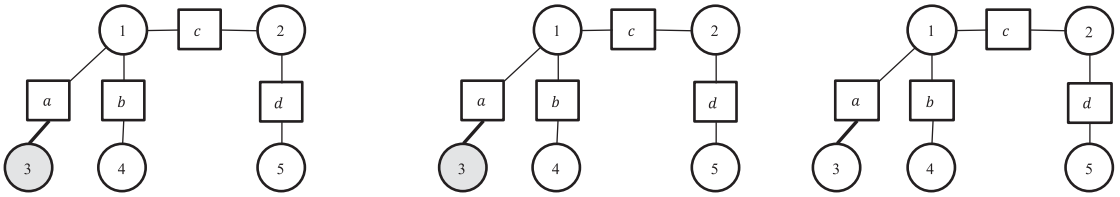


$\boldsymbol{r}: w_3 = \alpha_1, u_{3a} = \alpha_2, v_{1a} = \alpha_3.$

$\boldsymbol{r^1}: w_3^1 = 2\alpha_1, u_{3a}^1 = 2\alpha_2.$

$\boldsymbol{r^2}: v_{1a}^2 = 2\alpha_3.$

Figure 6. When $|S^w| = 1$ and $S_2 \neq \emptyset$ (u_{id} and v_{id} Are Nonzero for Some Node $i \in S$), $\mathbf{r}$ Is Not Extreme



$\boldsymbol{r}: w_3 = \alpha_1, u_{3a} = \alpha_2, v_{3a} = \alpha_3.$ $\quad \boldsymbol{r^1}: w_3^1 = \theta^1, u_{3a}^1 = \theta^1.$ $\quad \boldsymbol{r^2}: u_{3a}^2 = 2\alpha_2 - \theta^1, v_{3a}^2 = 2\alpha_3 - \theta^1.$

the lower- and upper-bound constraints, node 1's corresponding constraints in the projection are as follows:

$$\begin{aligned} &3x_1 + x_2 + x_3 + x_4 \geq 3 && \text{for } q = 0,\\ &2x_1 + x_2 + x_3 \geq 2,\ 2x_1 + x_2 + x_4 \geq 2,\\ &\quad 2x_1 + x_3 + x_4 \geq 2 && \text{for } q = 1,\\ &x_1 + x_2 \geq 1,\ x_1 + x_3 \geq 1,\ x_1 + x_4 \geq 1 && \text{for } q = 2. \end{aligned}$$

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. Because 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).

$$\begin{aligned} \text{(BIP4)}\quad \text{Minimize} \quad & \sum_{i \in V} b_i x_i \\ \text{Subject to :} \quad & (23),\ (24) \\ & x_i \in \{0,1\} \quad \forall i \in V \end{aligned}$$

Let LP4 be the linear relaxation of BIP4. Because the feasible region of LP4 is the projection of P_{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 δ denotes the largest degree among all nodes in V, that is, $\delta = \max\{deg(i) : i \in V\}$.

Proposition 2. *The valid inequalities* (24) *can be separated in* $O(|V|\ \delta \log \delta)$ *time.*

Proof. Given a fractional solution $\mathbf{x}^*$, 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:

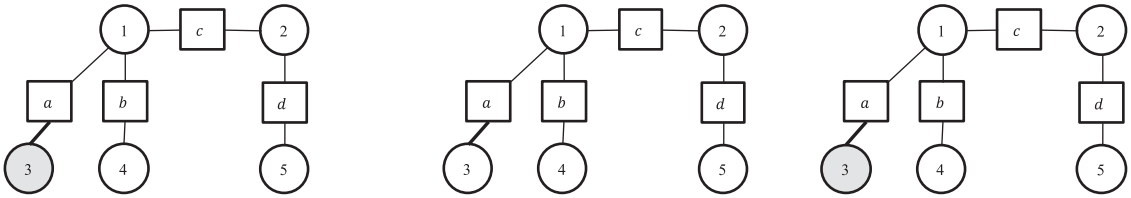
$$\text{Minimize} \quad (g_i - q)x_i^* + \sum_{j \in n(i)} x_j^* z_j, \tag{26}$$

$$\text{Subject to :} \quad \sum_{j \in n(i)} z_j = deg(i) - q, \tag{27}$$

$$z_j \in \{0,1\} \qquad \forall j \in n(i). \tag{28}$$

For each node i in V, if node i is in the set S, the binary variable z_i is one. Otherwise, it is zero. If the optimal objective value is smaller than $g_i - q$, we have a violated constraint. Otherwise, we change either 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).

Figure 7. When $|S^w| = 1$ and $S^+ \neq \emptyset$ (Either u_{id} or v_{id} Is Greater Than α for Some Node $i \in S$), $\mathbf{r}$ Is Not Extreme



$\boldsymbol{r}: w_3 = \alpha, u_{3a} = \alpha + \beta.$ $\quad \boldsymbol{r^1}: u_{3a}^2 = 2\beta.$ $\quad \boldsymbol{r^2}: w_3^1 = 2\alpha, u_{3a}^1 = 2\alpha.$

Algorithm 1 (Separation Algorithm for Inequality Set (24))

Require: A solution $\mathbf{x}^*$ and a PIDS instance.

1: **for** $i \in V$ **do**
2: Let $S \leftarrow n(i)$ and sort nodes in S in descending order by their x_i^* values.
3: **for** $q = 1, 2, \ldots, g_i - 1$ **do**
4: let $m_q = \arg\max\{x_i^* : i \in S\}$ and $S \leftarrow S \backslash m_q$.
5: **if** $(g_i - q)x_i^* + \sum_{j \in S} x_j^* < g_i - q$ **then**
6: 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**

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 one, the left-hand side decreases by $x_i^* + x_{m_q}^*$, where $x_{m_q}^*$ is the qth largest value of x_j^* for all j in $n(i)$, and the right-hand side decreases by one. 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 $x_i^* + x_{m_{q_0}}^* < 1$. Then, in a future iteration, we will not find any violated constraint for this particular i because $x_i^* + x_{m_q}^* \leq x_i^* + x_{m_{q_0}}^*$ 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 δ 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 PIDSs 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 $\mathbf{1} \in \mathbb{R}^{|V|}$ denote the vector that contains all ones, and $\mathbf{e}_i \in \mathbb{R}^{|V|}$ denote the vector that has one in ith position but zeros 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. *The term $P_T(G)$ 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 $\mathbf{x}_0 = \mathbf{1}$. Another $|V|$ points can be obtained as: $\mathbf{x}_i = \mathbf{1} - \mathbf{e}_i$ for all i in V. We obtain $|V|$ linearly independent points by $(\mathbf{x}_i - \mathbf{x}_0) = -\mathbf{e}_i$ for all i in V. Thus, $\mathbf{x}_0$ and $\mathbf{x}_i$ for all i in V are affinely independent. Furthermore, all of these points are feasible in $P_T(G)$. First, $\mathbf{x}_0$ means that we pick all nodes. Second, $\mathbf{x}_i$ 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. □

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 $\mathbf{x}_0 = \mathbf{1}$. Then, we can have $(|V| - 1)$ more points as $\mathbf{x}_j = \mathbf{1} - \mathbf{e}_j$ for all j in $V \backslash \{i\}$. From these points, we can obtain $|V| - 1$ linearly independent points as $(\mathbf{x}_j - x_0) = -\mathbf{e}_j$ for all j in $V \backslash \{i\}$. Thus, $\mathbf{x}_0$ and $\mathbf{x}_j$ for all j in $V \backslash \{i\}$ are affinely independent. They are feasible points, as shown in Theorem 5. Consequently, $x_i \leq 1$ is a facet of $P_T(G)$. □

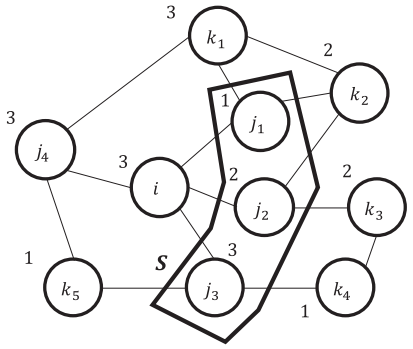
Proposition 4. *For a node i in V, its corresponding trivial constraint $x_i \geq 0$ is a face of $P_T(G)$. Furthermore, $x_i \geq 0$ is facet defining for $P_T(G)$ if the following conditions are satisfied:*

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. The term $x_i \geq 0$ is satisfied with equality because of the feasible point $\mathbf{x}_i = \mathbf{1} - \mathbf{e}_i$. 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 $F(i) = \{j \in n(i) : g_j \leq deg(j) - 1$ if $n(i) \cap n(j) = \emptyset$, $g_j \leq deg(j) - 2$ 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: $\mathbf{x}_i = \mathbf{1} - \mathbf{e}_i$, and $\mathbf{x}_j = \mathbf{x}_i - \mathbf{e}_j$ for all j in $F(i)$, and $\mathbf{x}_k = \mathbf{x}_i - \mathbf{e}_k$ for all k in $\{V \backslash n(i)^+\}$, where $n(i)^+ = n(i) \cup i$. First, $\mathbf{x}_j = \mathbf{x}_i - \mathbf{e}_j$ for all j in $F(i)$ are not feasible if $g_i = deg(i)$. However, they are feasible when $g_i \leq deg(i) - 1$. Second, $\mathbf{x}_i$ and $\mathbf{x}_k$ for all k in $\{V \backslash n(i)^+\}$ are feasible and distinct points in $P_T(G)$. Therefore, we have $|V| - |n(i)| - 1 + |F(i)| + 1$ feasible points in $P_T(G)$. Also, $\mathbf{x}_j - \mathbf{x}_i = -\mathbf{e}_j$ for all j in $\{V \backslash n(i)^+ \cup F(i)\}$ 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

Figure 8. Illustration of Notation in the Facet-Defining Proof of Inequality (22)



$a_i =$	2
$S =$	$\{j_1, j_2, j_3\}$
$N_{\text{all}} = \{j \in n(i) : g_j = deg(j)\} =$	$\{j_3, j_4\}$
$S^+ = S \cup \{i\} =$	$\{i, j_1, j_2, j_3\}$
$H_1 = \{k \in n(S) \setminus i : g_k > deg(k) - \lvert n(k) \cap S \rvert\} =$	$\{k_2, k_3\}$
$S_{\text{all}} = S \cap N_{\text{all}} =$	$\{j_3\}$
$S \setminus S_{\text{all}} =$	$\{j_1, j_2\}$
$C_{S \setminus S_{\text{all}}}^{a_i - \lvert S_{\text{all}} \rvert} =$	$\{\{j_1\}, \{j_2\}\}$
$H_C = \{k \in n(S) \setminus i : g_k > \lvert n(k) \setminus S^+ \rvert + \lvert n(k) \cap \{C \cup S_{\text{all}}\} \rvert\}$	
Given $C = \{j_1\}$ and $C \cup S_{\text{all}} = \{j_1, j_3\}$, $H_C =$	$\{k_3\}$
Given $C = \{j_2\}$ and $C \cup S_{\text{all}} = \{j_2, j_3\}$, $H_C =$	$\{k_1\}$
$H_0 = \{k \in n(S^+) \setminus N_{\text{all}} : g_k > deg(k) - \lvert n(k) \cap S^+ \rvert\} =$	$\{k_1, k_2, k_3\}$

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 shorthand 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 $N_{\text{all}} = \{j \in n(i) : g_j = deg(j)\}$ (i.e., the set of node i's neighbors whose threshold is equal to their degree). In Figure 8, we have $N_{\text{all}} = \{j_3, j_4\}$. Let $H_1 = \{k \in n(S) \setminus i : g_k > deg(k) - |n(k) \cap S|\}$. 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 (because the number of neighbors they have outside S, that is, $deg(k) - |n(k) \cap S|$, is less than their threshold). In Figure 8, $H_1 = \{k_2, k_3\}$. For node k_2, $g_{k_2} = 2$ and $deg(k_2) - |n(k_2) \cap S| = 1$. For node k_3, $g_{k_3} = 2$ and $deg(k_3) - |n(k_3) \cap S| = 1$. Finally, for a node k in H_1, $\gamma_k = g_k - (deg(k) - |n(k) \cap S|)$ 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. $deg(j) - |n(j) \cap S| \geq g_j$ *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* $|S| = 1$.
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 p. 144 of Wolsey (1998) to show that Inequality (22) is facet defining with the previous 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 1–3 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. Included in Section EC.2.2 of the e-companion. □

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. Included in Section EC.2.3 of the e-companion. □

In Lemma 3, we show that 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. Included in Section EC.2.4 of the e-companion. □

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 $H_1 = \{k \in n(S) \setminus i : g_k > deg(k) - |n(k) \cap S|\}$. 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 $\mathbf{x}_0 = \mathbf{1} - \sum_{j \in S} \mathbf{e}_j$, (i.e., select all nodes in $V \setminus S$ in the PIDS;) and $\mathbf{x}_j = \mathbf{x}_0 - \mathbf{e}_j$ for all j in $\{V \setminus (S^+ \cup H_1)\}$ (i.e., from the solution $\mathbf{x}_0$, remove a single node that is not i or in H_1). In the example of Figure 8, $\mathbf{x}_0$ has nodes $i, j_4, k_1, k_2, k_3, k_4, k_5$. Because $H_1 = \{k_2, k_3\}$, we can get four additional feasible points by removing j_4, k_1, k_3, and k_5 one at a time from $\mathbf{x}_0$.

Second, when $x_i = 0$, we have $x_j = 1$ for all j in N_{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 $S \backslash S_{\text{all}}$ in the PIDS. Let $C_{S\backslash S_{\text{all}}}^{a_i-|S_{\text{all}}|}$ be the set of all combinations. In Figure 8, $C_{S\backslash S_{\text{all}}}^{a_i-|S_{\text{all}}|} = \{\{j_1\},\{j_2\}\}$. Then, for a given $C \in C_{S\backslash S_{\text{all}}}^{a_i-|S_{\text{all}}|}$, we can have $\mathbf{x}_0^C = \mathbf{1} - \sum_{j\in S^+\backslash\{C\cup S_{\text{all}}\}} \mathbf{e}_j$ (i.e., select all nodes except those in $S^+\backslash\{C \cup S_{\text{all}}\}$). For the instance in Figure 8, if $C = \{j_1\}$, $\mathbf{x}_0^C$ selects all nodes except for i and j_2. If $C = \{j_2\}$, $\mathbf{x}_0^C$ 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 $\mathbf{x}_0^C$. 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_{all} or in S (removing a node in S will cause the inequality to no longer be satisfied at equality). Second, for a node $k \in n(S)\backslash i$, if its threshold is strictly greater than the number of its neighbors selected in $\mathbf{x}_0^C$, removing node k results in an infeasible solution. Let $H_C = \{k \in n(S)\backslash i : g_k > |n(k)\backslash S^+| + |n(k) \cap \{C \cup S_{\text{all}}\}|\}$. In the example of Figure 8, given $C = \{j_1\}$ and $\mathbf{x}_0^C$ as described previously, $H_C = \{k_3\}$. Thus, we can obtain four additional feasible points by removing k_1, k_2, k_4, and k_5 one at a time from $\mathbf{x}_0^C$. 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 $\mathbf{x}_0^C$. Overall, for a feasible point $\mathbf{x}_0^C$ associated with a C in $C_{S\backslash S_{\text{all}}}^{a_i-|S_{\text{all}}|}$, we obtain points $\mathbf{x}_j^C = \mathbf{x}_0^C - \mathbf{e}_j$ for all j in $\{V\backslash(S^+ \cup N_{\text{all}} \cup H_C)\}$ (i.e., from the solution $\mathbf{x}_0^C$, remove a single node that is not in C, N_{all}, H_C). Overall, we can find $T_0 = \sum_{C\in C_{S\backslash S_{\text{all}}}^{a_i-|S_{\text{all}}|}} (|V| - |S| - |H_C| - |N_{\text{all}}\backslash S_{\text{all}}|)$ feasible points in the forms $\mathbf{x}_0^C$ and $\mathbf{x}_j^C$ in $P_T(G)$.

We form a linear equation system based on these feasible points $\mathbf{x}_0$, $\mathbf{x}_j$, $\mathbf{x}_0^C$ and $\mathbf{x}_j^C$ with $|V|+1$ unknowns $(\mu_0, \boldsymbol{\mu})$, where $\boldsymbol{\mu}$ corresponds to the coefficients in the left-hand side of Inequality (22), and μ_0 corresponds to the right-hand side value. Feasible point $\mathbf{x}_0$ yields Equation (29), feasible point(s) $\mathbf{x}_j$ yields Equation (30), feasible point $\mathbf{x}_0^C$ yields Equation (31), and feasible point(s) $\mathbf{x}_j^C$ yields Equation (32).

$$\mu_i + \sum_{h\in V\backslash S^+} \mu_h = \mu_0 \tag{29}$$

$$\mu_i + \sum_{h\in V\backslash S^+} \mu_h - \mu_j = \mu_0 \qquad \forall j \in \{V\backslash(S^+ \cup H_1)\} \tag{30}$$

$$\sum_{k\in C\cup S_{\text{all}}} \mu_j + \sum_{h\in V\backslash S^+} \mu_h = \mu_0 \qquad \forall C \in C_{S\backslash S_{\text{all}}}^{a_i-|S_{\text{all}}|} \tag{31}$$

$$\sum_{k\in C\cup S_{\text{all}}} \mu_j + \sum_{h\in V\backslash S^+} \mu_h - \mu_j = \mu_0 \qquad \forall C \in C_{S\backslash S_{\text{all}}}^{a_i-|S_{\text{all}}|},\ j \in \{V\backslash(S^+ \cup N_{\text{all}} \cup H_C)\} \tag{32}$$

Let $(\pi_0, \boldsymbol{\pi})$ be a vector whose positions are labeled from zero to $|V|$. Following approach 2 on p. 144 of Wolsey (1998), we need to show that $(\mu_0, \boldsymbol{\mu}) = \lambda(\pi_0, \boldsymbol{\pi})$, $\lambda \neq 0$; with $(\pi_0, \boldsymbol{\pi})$ with a_i in the zeroth and ith positions, one in the jth position for all j in S, and zeros in all other positions is the only solution to the linear equation system (29)–(32). In Lemmas 4 and 5, we show the conditions under which the vector $(\pi_0, \boldsymbol{\pi})$ (as defined previously) is the only solution to the linear equation system. First, in Lemma 4, we show the necessary condition leading to zeros for all positions in $V\backslash S^+$.

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

Proof. Included in Section EC.2.5 of the e-companion. □

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

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

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

Proof. Included in Section EC.2.6 of the e-companion. □

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

Proof of Theorem 6. *Necessity.* Proved by Lemmas 3–5.

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,

$$T_0 + T_1 = |V| - |S| - |H_1| + \sum_{C\in C_S^{a_i}} (|V| - |S| - |H_C|). \tag{33}$$

Let $H_0 = \{k \in n(S)\backslash N_{\text{all}} : g_k > deg(k) - |n(k) \cap S^+|\}$. 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 $C_S^{a_i}$, $H_0\backslash H_C$ provides the set of S's neighbors that can be removed one by one from $\mathbf{x}_0^C$. Given Condition 4, each node j in H_1 can be removed from an $\mathbf{x}_0^C$ for some C in $C_S^{a_i}$. Thus, $H_1 \subseteq \cup_{C\in C_S^{a_i}} H_0\backslash H_C$. Hence, $\sum_{C\in C_S^{a_i}} |H_0| - |H_C| \ge |H_1|$ 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 (33) as follows:

$$T_0 + T_1 = |V| - |S| + \sum_{C \in C_S^{a_i}} (|V| - |S| - |H_0|) + \sum_{C \in C_S^{a_i}} (|H_0| - |H_C|) - |H_1|, \tag{34}$$

$$\geq |V| - |S| + |C_S^{a_i}| (|V| - |S| - |H_0|), \tag{35}$$

$$\geq |V| - |S| + |C_S^{a_i}|, \tag{36}$$

$$\geq |V|. \tag{37}$$

We can go from (34) to (35) using the fact that $\sum_{C \in C_S^{a_i}} |H_0| - |H_C| \geq |H_1|$. We can go from (35) to (36) by noting that $|V| - |S| - |H_0| \geq 1$. This is true because at least node i is left in $V \backslash (S \cup H_0)$. Finally, we can go from (36) to (37) because Condition 3 of Theorem 6 implies that $|C_S^{a_i}| \geq |S|$. (Observe that when $|S| = 1$ and $a_i = 1$, $|C_S^{a_i}| = |S|$. When $|S| \geq 2$ and $1 \leq a_i \leq |S| - 1$, first consider $a_i \leq \lfloor \frac{|S|}{2} \rfloor$, $|C_S^{a_i}| = \binom{|S|}{a_i} = \frac{|S|!}{a_i!(|S|-a_i)!} = \binom{|S|}{a_i - 1} \frac{|S|+1-a_i}{a_i} = \binom{|S|}{1} \prod_{h=2}^{a_i} \frac{|S|+1-h}{h}$ $= |S| \prod_{h=2}^{a_i} \frac{|S|+1-h}{h} \geq |S|$ because $\frac{|S|+1-h}{h} \geq 1$ 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, $|C_S^{a_i}| = \binom{|S|}{a_i} = \binom{|S|}{|S| - a_i} \geq |S|$.) 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, $(\mu_0, \boldsymbol{\mu}) = \lambda(\pi_0, \boldsymbol{\pi})$, where $\lambda \neq 0$, and $(\pi_0, \boldsymbol{\pi})$, which has a_i in the zeroth and ith positions, one in the jth position for all j in S, and zeros 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. □

5. Computational Study

In this section, we discuss our computational experience with these four formulations on some large real-world social networks. Our computational experiments have three goals: (i) to examine the strength of the formulations (to evaluate the benefit obtained by BIP3 and BIP4), (ii) to examine the performance of the four formulations, and (iii) to solve exactly or to near optimality simulated instances on very large real-world networks. We make these evaluations on a set of seven large real-world social networks first, before embarking on solving instances on three very large real-world social networks with up to about 2.5 million nodes and 8 million edges. Our computational experiments are on a machine with the following specifications: Intel Xeon E5-2630V4 processor, 64 GB RAM, and Ubuntu operating system. Furthermore, we use CPLEX 12.8 with the Python API.

5.1. Description of Real-World Social Networks

We have 10 real-world social networks: Gnutella, Ning, Hamsterster, Escorts, Anybeat, Advogato, Delicious, Youtube, Lastfm, and Flixster. They are taken from the Stanford Large Network Data set Collection (SNAP; Leskovec and Krevl 2014), the BGU Social Networks Security Research Group (BGU; Lesser et al. 2013), the Koblenz Network Collection (KONECT; Kunegis 2017), and the Network Repository (N.R.; Rossi and Ahmed 2015). In all 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 2, we list each graph, along with its source repository and the number of nodes and edges of the largest component it contains.

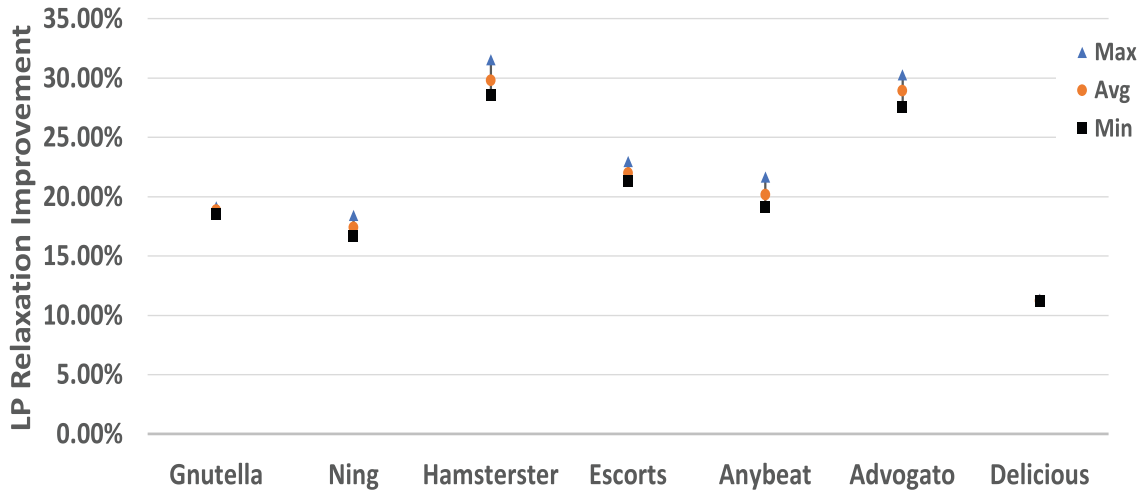
Our first set of experiments are built on seven large real-world networks. *Gnutella* is a large file sharing peer-to-peer network (the first decentralized peer-to-peer network of its kind). We consider one snapshot of the Gnutella network collected on August 4, 2002. *Ning* is an online platform for people and organizations to create custom social networks. *Hamsterster* contains friendships between users of the website hamsterster.com. *Escorts* is the bipartite network of buyers and their escorts. *Anybeat* is the friendship network of an online community. *Advogato* is based on the friendship network of Advogato.org. *Delicious* is a social bookmarking web service for storing, sharing and discovering web bookmarks.

Our last experiment focuses on BIP4 and three very large real-world social networks. *Youtube* contains the friendship network crawled on Youtube. An edge means that a user subscribes to the other users' content. *Lastfm* represents the friendship among the users on a music social network. *Flixster* is the social

Table 2. Source and Size of the Largest Components of Real-World Graphs

Graph name	Source	No. of nodes	No. of edges
Gnutella	SNAP	10,876	39,994
Ning	BGU	9,727	40,570
Hamsterster	Konect	1,788	12,476
Escorts	Konect	10,106	39,016
Anybeat	N.R.	12,645	49,132
Advogato	N.R.	5,042	39,277
Delicious	N.R.	536,108	1,365,961
Youtube	BGU	1,134,890	2,987,624
Lastfm	Konect	1,191,805	4,519330
Flixster	N.R.	2,523,386	7,918,801

Figure 9. (Color online) Relative Improvement of the LP Relaxation of LP4 over That of LP1



network of flixster.com, a movie rating site on which people can meet others with a similar movie tastes.

The threshold value g_i associated with a node i in a given network is generated from a discrete uniform distribution between $[1, deg(i)]$. 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. The URL http://dx.doi.org/10.17632/ywfgkk5pky.1 provides these 100 instances.

5.2. Investigating the Strength of the LP Relaxations

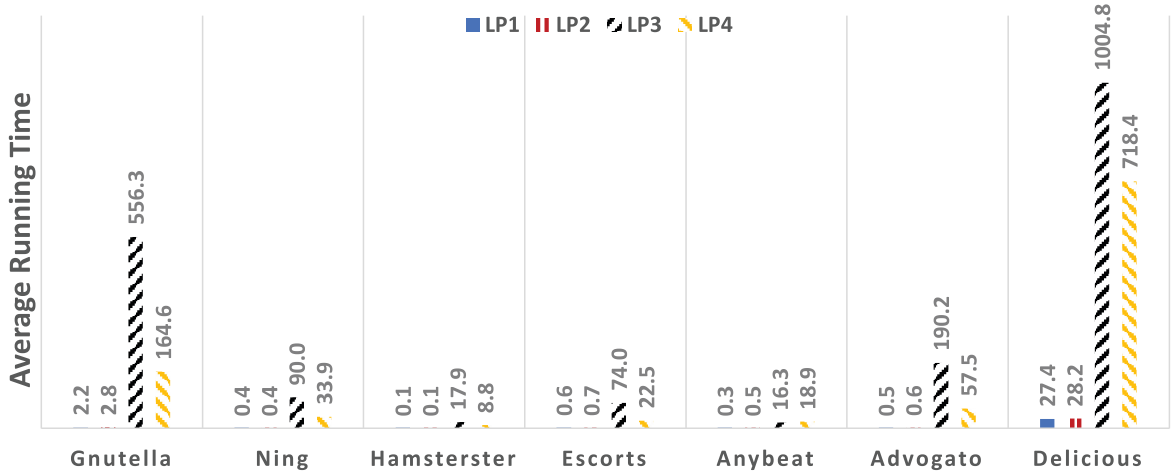
We have already shown that BIP3 and BIP4 are stronger than BIP1 and BIP2 in Theorem 1. Now, we empirically evaluate how much stronger BIP3 and BIP4 are compared with BIP1 and BIP2 in this section. In our implementation for BIP3, we remove the constraint $y_{id} + y_{di} = 1$ and use only variables y_{di} (so y_{id} is replaced by $1 - y_{di}$ in the model). In this way, we reduce the size of the model by $|E_t|$ constraints and $|E_t|$ 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 $z_{LP3} = z_{LP4}$. Thus, we consider the relative improvement of LP4 over LP1 calculated as $\frac{z_{LP4} - z_{LP1}}{z_{LP1}} \times 100$. Figure 9 plots the average, minimum, and maximum value of the relative improvement over the 10 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. Table 3 reports the running times, and Figure 10 plots the average running time. For LP4, the running time includes both the separation and solving time. First, although LP3 and LP4 improve

Table 3. Running Time (in Seconds) of LP1, LP2, LP3, and LP4

	LP1			LP2			LP3			LP4		
	Average	Minimum	Maximum	Average	Minimum	Maximum	Average	Minimum	Maximum	Average	Minimum	Maximum
G04	2.2	1.9	2.4	2.8	2.4	3.1	556.3	479.3	649.9	164.6	150.3	179.7
Ning	0.4	0.3	0.4	0.4	0.4	0.5	90.0	65.3	151.6	33.9	26.1	40.6
Hamsterster	0.1	0.1	0.1	0.1	0.1	0.1	17.9	15.6	19.6	8.8	6.7	10.9
Escorts	0.6	0.6	0.8	0.7	0.7	0.8	74.0	55.6	100.8	22.5	17.8	28.0
Anybeat	0.3	0.2	0.3	0.5	0.4	0.5	16.3	11.3	26.2	18.9	14.9	24.4
Advogato	0.5	0.4	0.6	0.6	0.5	0.7	190.2	163.9	228.1	57.5	43.0	70.5
Delicious	27.4	24.2	31.9	28.2	24.0	32.1	1,004.8	890.7	1,146.5	718.4	597.7	806.4

Figure 10. (Color online) Average Running Time (in Seconds) of LP1, LP2, LP3, and LP4



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. 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. A natural question is whether the extra effort involved in solving the LP relaxations of BIP3 and BIP4 yields a computational benefit (against using BIP1 or 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

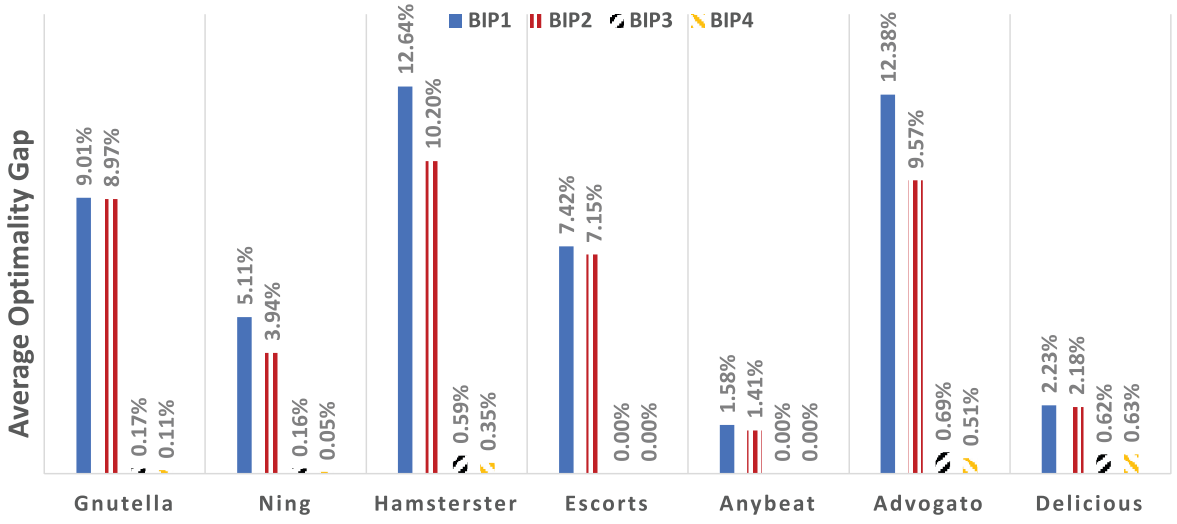
In this section, we test the performance of BIP1, BIP2, BIP3, and BIP4. Recall that in our implementation for BIP3, we remove the constraint $y_{id} + y_{di} = 1$ and use only variables y_{di}. Next, in our implementation for BIP4, we start with Constraint (23) and use CPLEX's callbacks to add violated Constraint (24) dynamically because 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. Although 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)). Last, in order to focus on the effect of the four formulations, we follow common practice in the literature (Avella et al. 2015, Leitner et al. 2015, Chopra et al. 2017, Leitner et al. 2019, Schmidt et al. 2021) and use a single thread and turn off CPLEX's cuts. Other than that, we keep the default setting for CPLEX. For each instance, the running time is capped at 3,600 seconds (one hour), unless stated otherwise.

We consider the four IP formulations on the first set of 70 real-world social network instances, based on the seven graphs: Gnutella, Ning, Hamsterster, Escorts, Anybeat, Advogato, and Delicious. The optimality gap is reported in Table 4. 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

Table 4. Optimality Gap (%) of BIP1, BIP2, BIP3, and BIP4

	BIP1			BIP2			BIP3			BIP4		
	Average	Minimum	Maximum	Average	Minimum	Maximum	Average	Minimum	Maximum	Average	Minimum	Maximum
Gnutella	9.01%	8.60%	9.49%	8.97%	8.53%	9.41%	0.17%	0.09%	0.32%	0.11%	0.01%	0.21%
Ning	5.11%	4.28%	6.03%	3.94%	3.56%	4.31%	0.16%	0.06%	0.26%	0.05%	0.00%	0.13%
Hamsterster	12.64%	11.42%	13.86%	10.20%	8.89%	11.65%	0.59%	0.00%	1.45%	0.35%	0.00%	1.04%
Escorts	7.42%	6.90%	8.05%	7.15%	6.75%	7.71%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Anybeat	1.58%	1.17%	2.02%	1.41%	1.23%	1.57%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Advogato	12.38%	10.38%	13.70%	9.57%	8.57%	10.79%	0.69%	0.34%	1.33%	0.51%	0.16%	1.99%
Delicious	2.23%	2.17%	2.31%	2.18%	2.12%	2.26%	0.62%	0.53%	0.71%	0.63%	0.44%	0.90%

Figure 11. (Color online) Average Optimality Gap (%) of BIP1, BIP2, BIP3, and BIP4



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 $\frac{z_{BFS}-lb}{z_{BFS}} \times 100$. 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 more than 12% for Hamsterster compared with 0.59% for BIP3 and 0.35% for BIP4. BIP2 behaves in a similar fashion to 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 4. Next, we take a closer look at BIP3 and BIP4. Table 5 shows the average optimality gaps with time limits of 300, 600, 900, and 3,600 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, whereas 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. The main reason BIP3 takes much longer is because solving the LP relaxations take much longer, and thus it handles fewer branch-and-bound nodes (than BIP4 in the same amount of time).

Furthermore, Table 6 contains the running times of BIP3 and BIP4 for those solved instances. The column Solved no. presents the number of solved instances. In addition to all instances solved by BIP3, BIP4 is able to solve four more Ning instances. For the instances solved by both BIP3 and BIP4, BIP4 is much faster compared with 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

Table 5. Average Optimality Gap (%) of BIP3 and BIP4 with 300-, 600-, 900-, and 3,600-Second Time Limits

	BIP3				BIP4			
Time limit	300	600	900	3,600	300	600	900	3,600
Gnutella	98.73%	1.58%	0.26%	0.17%	0.21%	0.18%	0.15%	0.11%
Ning	0.40%	0.28%	0.24%	0.16%	0.16%	0.12%	0.10%	0.05%
Hamsterster	1.11%	1.11%	0.96%	0.59%	0.77%	0.61%	0.55%	0.35%
Escorts	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Anybeat	0.03%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Advogato	14.22%	1.33%	1.20%	0.69%	0.85%	0.71%	0.65%	0.51%
Delicious	98.86%	98.86%	98.86%	0.62%	0.64%	0.63%	0.63%	0.63%

Table 6. Running Time (in Seconds) of BIP3 and BIP4 on Solved Instances

	BIP3				BIP4			
	Solved no.	Average	Minimum	Maximum	Solved no.	Average	Minimum	Maximum
Ning	0	NA	NA	NA	4	2,553.9	1,536.3	3,272.0
Hamsterster	3	1,524.7	908.3	2,450.9	3	376.2	213.3	487.5
Escorts	10	35.3	22.1	47.8	10	14.8	10.6	24.5
Anybeat	10	191.6	55.6	444.6	10	22.6	19.1	26.4

Note. NA, Not Applicable.

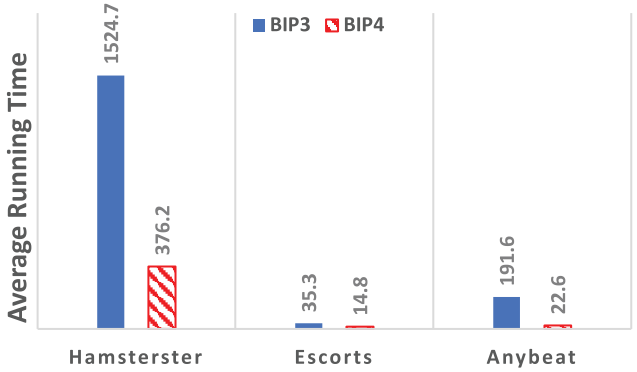
becomes rather apparent on the very large-scale real-world social networks.

Next, we consider the 30 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 7. BIP4 solves 24 of 30 instances optimally. For the remaining six Youtube instances, the optimality gap is less than 0.02%. Table 7 reports the running time on the solved instances.

Based on these experiments, we 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.

Before concluding this section, we briefly comment on the structure of the solutions. Specifically, Table 8 presents for each of the 10 graphs, the average, minimum, and maximum fraction of nodes selected in the PIDS across the 10 instances. This fraction is calculated by dividing the number of nodes selected in the best feasible solution to a PIDS problem instance by the total number of nodes. For each graph, one can see the fraction of nodes selected in the PIDS are close across the 10 instances. However, there is significant variation across the 10 different graphs, with the average value ranging from 3.81% to 36.66% (even for graphs with similar density there seems to be significant variation). Naturally, this leads to the research question whether there are factors that can explain this variation? This is not an easy question to answer, and an interesting research problem in its own right. To properly analyze this, one needs to work systematically with a huge number of smaller simulated graph instances where it is possible to vary graph structures, consider the various ways of generating a node's weight (b_i) and its threshold value (g_i), the relationship between them, and control different aspects of the simulated graphs. The detailed study of this question is left for future research.

Figure 12. (Color online) Average Running Time (in Seconds) of BIP3 and BIP4 on Solved Instances



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

Table 7. Performance of BIP4 on Very Large Real-World Instances

	Optimality gap			Running time on solved instances			
	Average	Minimum	Maximum	Solved no.	Average	Minimum	Maximum
Youtube	0.01%	0.00%	0.02%	4	1253.1	960.1	1551.1
Lastfm	0.00%	0.00%	0.00%	10	330.1	293.1	395.3
Flixster	0.00%	0.00%	0.00%	10	366.3	338.9	447.7

proposed formulations. On a testbed 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 of the 100 instances to optimality.

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 nth 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 (because 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. 2018, Günneç et al. 2020b). 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 (2021a) 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. 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.

Table 8. Fraction of Nodes Selected in the PIDS

	Gutella	Ning	Hamsterster	Escorts	Anybeat	Advogato	Delicious	Youtube	Lastfm	Flixster
Average	36.66%	31.58%	33.44%	32.74%	16.98%	32.94%	15.28%	23.31%	6.53%	3.81%
Minimum	36.38%	31.18%	32.72%	32.55%	16.77%	32.59%	15.27%	23.29%	6.52%	3.81%
Maximum	36.89%	31.95%	34.28%	32.92%	17.30%	33.56%	15.31%	23.33%	6.54%	3.82%

Acknowledgments

This paper is partly based on the third chapter of the author's doctoral dissertation (Zhang 2016).

References

Avella P, Boccia M, Wolsey LA (2015) Single-item reformulations for a vendor managed inventory routing problem: Computational experience with benchmark instances. *Networks* 65(2): 129–138.

Baïou M, Barahona F (2014) The dominating set polytope via facility location. *Proc. Combinatorial Optimization: 3rd Internat. Sympos.* (Springer, New York), 38–49.

Balas E, Pulleyblank W (1983) The perfectly matchable subgraph polytope of a bipartite graph. *Networks* 13(4):495–516.

Balasundaram B, Butenko S, Hicks IV (2011) Clique relaxations in social network analysis: The maximum *k*-plex problem. *Oper. Res.* 59(1):133–142.

Banerjee S, Jenamani M, Pratihar DK (2020) A survey on influence maximization in a social network. *Knowledge Inform. Systems* 62: 3417–3455.

Bond RM, Fariss CJ, Jones JJ, Kramer AD, Marlow C, Settle JE, Fowler JH (2012) A 61-million-person experiment in social influence and political mobilization. *Nature* 489(7415): 295–298.

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

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

Chopra S, Filipecki B, Lee K, Ryu M, Shim S, Van Vyve M (2017) An extended formulation of the convex recoloring problem on a tree. *Math. Programming* 165(2):529–548.

Dhawan A, Rink M (2015) Positive influence dominating set generation in social networks. *Proc. Internat. Conf. on Comput. and Network Comm.* (IEEE, New York), 112–117.

Dinh TN, Shen Y, Nguyen DT, Thai MT (2014) On the approximability of positive influence dominating set in social networks. *J. Combinatorial Optim.* 27(3):487–503.

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

Ghayour-Baghbani F, Asadpour M, Faili H (2021) MLPR: Efficient influence maximization in linear threshold propagation model using linear programming. *Soc. Network Anal. Mining* 11(1):1–10.

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

Güney E (2019a) An efficient linear programming based method for the influence maximization problem in social networks. *Inform. Sci.* 503:589–605.

Güney E (2019b) On the optimal solution of budgeted influence maximization problem in social networks. *Oper. Res.* 19(3): 817–831.

Güney E, Leitner M, Ruthmair M, Sinnl M (2021) Large-scale influence maximization via maximal covering location. *Eur. J. Oper. Res.* 289(1):144–164.

Günneç D, Raghavan S, Zhang R (2020a) A branch-and-cut approach for the least cost influence problem on social networks. *Networks* 76(1):84–105.

Günneç D, Raghavan S, Zhang R (2020b) Least-cost influence maximization on social networks. *INFORMS J. Comput.* 32(2): 289–302.

Haynes T, Hedetniemi S, Slater P (1998a) *Domination in Graphs: Advanced Topics* (CRC Press, Boca Raton, FL).

Haynes T, Hedetniemi S, Slater P (1998b) *Fundamentals of Domination in Graphs* (CRC Press, Boca Raton, FL).

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

Kempe D, Kleinberg J, Tardos E (2015) Maximizing the spread of influence through a social network. *Theory Comput.* 11(4):105–147.

Khomami MMD, Rezvanian A, Bagherpour N, Meybodi MR (2018) Minimum positive influence dominating set and its application in influence maximization: A learning automata approach. *Appl. Intelligence* 48(3):570–593.

Kunegis J (2017) KONECT network data set. Accessed December 5, 2020, http://konect.cc.

Leitner M, Ljubić I, Sinnl M (2015) A computational study of exact approaches for the bi-objective prize-collecting Steiner tree problem. *INFORMS J. Comput.* 27(1):118–134.

Leitner M, Ljubić I, Riedler M, Ruthmair M (2019) Exact approaches for network design problems with relays. *INFORMS J. Comput.* 31(1):171–192.

Leskovec J, Krevl A (2014) SNAP Datasets: Stanford large network data set collection. Accessed December 5, 2020, http://snap.stanford.edu/data.

Lesser O, Tenenboim-Chekina L, Rokach L, Elovici Y (2013) Intruder or welcome friend: Inferring group membership in online social networks. *Social Computing, Behavioral-Cultural Modeling and Prediction* (Springer, New York), 368–376.

Li Y, Fan J, Wang Y, Tan K-L (2018) Influence maximization on social graphs: A survey. *IEEE Trans. Knowledge Data Engrg.* 30(10): 1852–1872.

Lin G, Guan J, Feng H (2018) An ILP based memetic algorithm for finding minimum positive influence dominating sets in social networks. *Phys. A* 500:199–209.

Matz SC, Kosinski M, Nave G, Stillwell DJ (2017) Psychological targeting as an effective approach to digital mass persuasion. *Proc. National Acad. Sci. USA* 114(48):12714–12719.

Pajouh FM, Balasundaram B, Hicks IV (2016) On the 2-club polytope of graphs. *Oper. Res.* 64(6):1466–1481.

Raghavan S, Zhang R (2019) A branch-and-cut approach for the weighted target set selection problem on social networks. *INFORMS J. Optim.* 1(4):304–322.

Raghavan S, Zhang R (2021a) Influence maximization with latency requirements on social networks. *INFORMS J. Comput.*, ePub ahead of print November 9, https://doi.org/10.1287/ijoc.2021.1095.

Raghavan S, Zhang R (2021b) Weighted target set selection on trees and cycles. *Networks* 77(4):587–609.

Rossi RA, Ahmed NK (2015) The network data repository with interactive graph analytics and visualization. Bonet B, Koenig S, eds. *Proc. 29th AAAI Conf. on Artificial Intelligence* (AAAI Press, Palo Alto, CA), 4292–4293.

Saxena A (2004) On the dominating set polytope of a cycle. Working paper, Carnegie Mellon University, Pittsburgh, PA.

Schmidt M (2019) Calculating the true size of the influencer marketing industry. Accessed December 5, 2020, https://www.forbes.com/sites/forbestechcouncil/2019/02/13/calculating-the-true-size-of-the-influencer-marketing-industry/#28b76073658d.

Schmidt D, Zey B, Margot F (2021) Stronger MIP formulations for the steiner forest problem. *Math. Programming* 186(1): 373–407.

Schomer A (2020) Influencer marketing: State of the social media influencer market in 2020. Accessed December 5, 2020, https://www.businessinsider.com/influencer-marketing-report.

Shearer E, Matsa KE (2018). News use across social media platforms 2018. Technical report, Pew Research Center. Accessed December 5, 2020, https://www.pewresearch.org/journalism/2018/09/10/news-use-across-social-media-platforms-2018/.

Stein C, Cormen T, Rivest R, Leiserson C (2009) *Introduction to Algorithms* (MIT Press, Cambridge, MA).

Tong G, Wu W, Pardalos PM, Du D-Z (2017) On positive-influence target-domination. *Optim. Lett.* 11(2):419–427.

Valente TW (2012) Network interventions. *Science* 337(6090): 49–53.

Verma A, Buchanan A, Butenko S (2015) Solving the maximum clique and vertex coloring problems on very large sparse networks. *INFORMS J. Comput.* 27(1):164–177.

Walteros JL, Buchanan A (2020) Why is maximum clique often easy in practice? *Oper. Res.* 68(6):1866–1895.

Wang F, Camacho E, Xu K (2009) Positive influence dominating set in online social networks. *Proc. Internat. Conf. on Combinatorial Optim. and Applications* (Springer, New York), 313–321.

Wang F, Du H, Camacho E, Xu K, Lee W, Shi Y, Shan S (2011) On positive influence dominating sets in social networks. *Theoretical Comput. Sci.* 412(3):265–269.

Wolsey LA (1998) *Integer Programming* (Wiley, New York).

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

Zhang R (2016) Mathematical programming models for influence maximization on social networks. PhD thesis, University of Maryland, College Park.

Zhang B, Pavlou P, Krishnan R (2018) On direct vs. indirect peer influence in large social networks. *Inform. Systems Res.* 29(2):292–314.

Zhu X, Yu J, Lee W, Kim D, Shan S, Du D-Z (2010) New dominating sets in social networks. *J. Global Optim.* 48(4): 633–642.

Zou F, Zhang Z, Wu W (2009). Latency-bounded minimum influential node selection in social networks. *Proc. Internat. Conf. on Wireless Algorithms, Systems, and Applications* (Springer, New York), 519–526.

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 $b_i^1 = b_i$ and $b_i^2 = 0$ for all i in L.
4: **for** each star subproblem **do**
5: StarHandling
6: **end for**
7: SolutionBacktrack

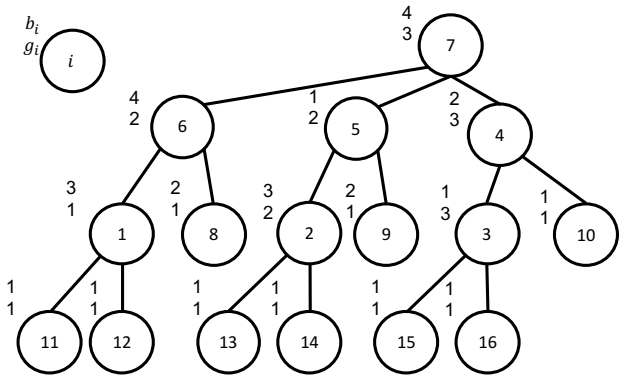


Figure EC.1 A PIDS problem instance.

X_c^{NPS} represent the set of nodes selected in the solution to star c, and let C_c^{NPS} 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_c^{PS} representing the set of nodes selected in the solution to star c and C_c^{PS} 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 $X_c^{NPS} = L(c)$ and $C_c^{NPS} = \sum_{j \in L(c)} b_j$. Otherwise, the central node c of the bottom star is selected with $X_c^{NPS} = \{c\}$ and $C_c^{NPS} = b_c$. 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 $X_c^{PS} = L(c)$ and $C_c^{PS} = \sum_{j \in L(c)} b_j$. Otherwise, the central node c is selected with $X_c^{PS} = \{c\}$ and $C_c^{PS} = b_c$.

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 $X_1^{NPS} = \{11, 12\}$ and $C_1^{NPS} = 2$. When node 1's parent (node 6) is selected, we make the same comparison. Thus, the solution is $X_1^{PS} = \{11, 12\}$ and $C_1^{NPS} = 2$.

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

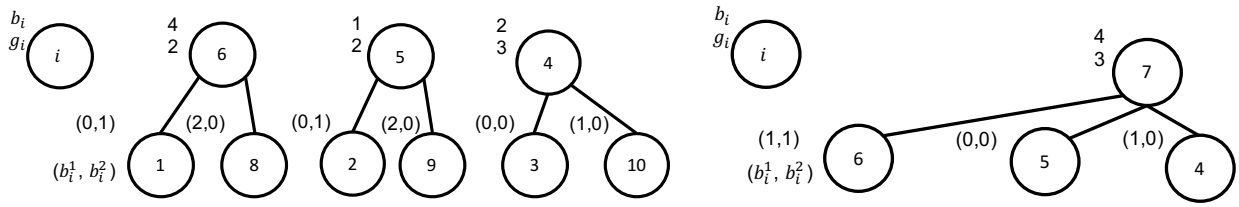


Figure EC.2 **(a) Star 4, 5 and 6.** **(b) The last star.**

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 $b_c^1 = C_c^{NPS} - C_c^{PS}$ and $b_c^2 = b_c - C_c^{NPS}$ are computed and used to solve the next star subproblem. Thus, $b_1^1 = 0$ and $b_1^2 = 1$. 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 $b_2^1 = 0$, $b_2^2 = 1$. In star 3, we have $X_3^{PS} = X_3^{NPS} = \{3\}$, $C_3^{PS} = C_3^{NPS} = 1$ and $b_3^1 = 0$, $b_3^2 = 0$. 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_i^1 and b_i^2 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, $b_8^1 = 2$, and $b_8^2 = 0$. Thus, $b_i^1 = b_i$, and $b_i^2 = 0$ 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 $B_c^{NPS} = \sum_{i \in L(c)} b_i^1$, since all of the children $i \in L(c)$ must at least incur

the cost of the solution X_i^{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_i^2 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** ($|L(c)| \geq g_c$): We select the first g_c nodes in $L(c)$ in ascending order of their b_i^2 value (we denote this set as S_{g_c}) and total cost of $B_c^{NPS} + \sum_{i \in S_{g_c}} b_i^2$. 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 $B_6^{NPS} = \sum_{i \in L(6)} b_i^1 = 2$. For NoParent-Selected, $X_6^{NPS} = \{1, 8\}$, and $C_6^{NPS} = 3$. For Parent-Selected, $X_6^{PS} = \{8\}$, and $C_6^{PS} = 2$. Thus, $TC = 7$. Contracting star 6 gives $b_6^1 = 1$ and $b_6^2 = 1$. For star 5, $|L(5)| = 2$. First, $b_5 = 1 < 2 = B_5^{NPS}$. Thus, we select node 5. Then, $X_5^{NPS} = X_5^{PS} = \{5\}$, and $C_5^{NPS} = C_5^{PS} = 1$. Thus, $TC = 8$. Contracting star 5 gives $b_5^1 = 0$ and $b_5^2 = 0$. For star 4, $|L(4)| = 2$. First, $b_4 = 2 > 1 = B_1^{NPS}$. Thus, we consider two cases. For the case where no parent is selected, we select node 4. The cost is 2. Thus, $X_4^{NPS} = \{4\}$, and $C_4^{NPS} = 2$. 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, $X_4^{PS} = \{3, 10\}$, and $C_4^{PS} = 1$. Thus, $TC = 9$. Contracting star 7 gives $b_4^1 = 1$ and $b_4^2 = 0$. Now, we only have one star left, as shown in Figure EC.2(b). Star 7 has $|L(7)| = 3$ and $B_7^{NPS} = 2 < 4$. 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, $X_7^{NPS} = \{4, 5, 6\}$,and $C_7^{NPS} = 3$. 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.
$(X_c^{NPS}, C_c^{NPS}) \leftarrow$ SolveStar(star c, NoParent-Selected).
if star c is the last star **then**
  $TC = TC + C_c^{NPS}$
else
  $(X_c^{PS}, C_c^{PS}) \leftarrow$ SolveStar(star c, Parent-Selected).
  The contracted node has $b_c^1 = C_c^{NPS} - C_c^{PS}$ and $b_c^2 = b_c - C_c^{NPS}$.
  $TC = TC + C_c^{PS}$
end if
function SolveStar(a star c, Flag)
  if Flag == Parent-Selected **then**
    $g_c = g_c - 1$.
  end if
  $B_c^{NPS} = \sum_{i \in L(c)} b_i^1$ and let S_{g_c} be the set of the cheapest g_c nodes in $L(c)$ by b^2.
  if $|L(c)| < g_c$ **then**
    $C = b_c$
  else
    $C = \min\left\{b_c,\ B_c^{NPS} + \sum_{i \in S_{g_c}} b_i^2\right\}$.
  end if
  If C is b_c, let $X \leftarrow b$.
  If C is $B_c^{NPS} + \sum_{i \in S_{g_c}} b_i^2$, let $X \leftarrow S_{g_c}$
  return X, C.
end function

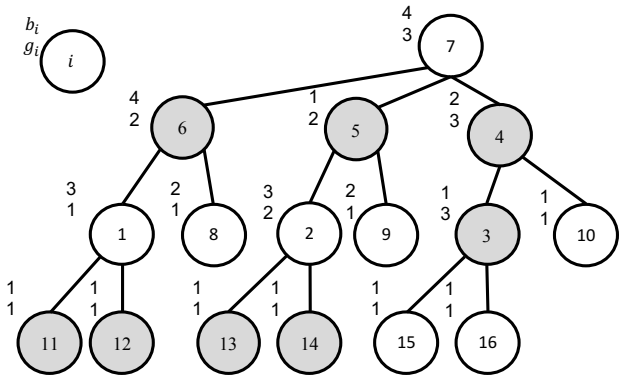


Figure EC.3 The solution obtained by our DP algorithm. Shaded nodes are selected in the PIDS.

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 $\mathbf{x}^*$ 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

Let $\mathbf{x}^* = \mathbf{0}$. Then, call PIECING($r$, $\mathbf{x}^*$, NoParent-Selected) for the root node r.
function PIECING(c, $\mathbf{x}$, $Flag$)
  If Flag == ParentSelected, $X^{'} \leftarrow X_c^{PS}$. **Otherwise**, $X^{'} \leftarrow X_c^{NPS}$.
  $X \leftarrow X \cup X^{'}$ and $x_i = 1 \, \forall i \in X^{'}$.
  if $c \in X^{'}$ **then**
    $\forall j \in \{L(c) \setminus L\}$ call PIECING(j, $\mathbf{x}$, Parent-Selected).
  else
    $\forall j \in \{L(c) \setminus (X^{'} \cup L)\}$ call PIECING(j, $\mathbf{x}$, NoParent-Selected).
    for $i \in \{X^{'} \setminus L\}$ **do**
      $\forall j \in \{L(i) \setminus L\}$ call PIECING(j, $\mathbf{x}$, Parent-Selected).
    end for
  end if
  return x.
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_cth 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. Technical Proofs

EC.2.1. Proof of Theorem 2

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

$$\text{(LP3)} \quad \text{Minimize} \quad \sum_{i \in V} b_i x_i \tag{EC.1}$$

$$\text{Subject to: } (t_{ij}) \quad x_i - y_{dj} \geq 0 \qquad \forall i \in V,\, j \in n(i) \tag{EC.2}$$

$$(u_{id}) \quad y_{id} - x_i \geq 0 \qquad \forall i \in V, d \in a(i) \tag{EC.3}$$

$$(v_{id}) \quad -y_{id} - y_{di} = -1 \qquad \forall \{i,d\} \in E_t \tag{EC.4}$$

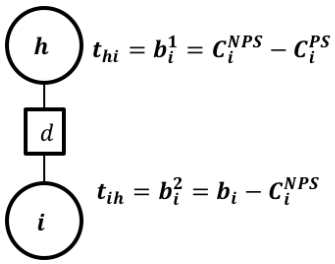
$$(w_i) \quad \sum_{d \in a(i)} y_{di} + g_i x_i \geq g_i \quad \forall i \in V \tag{EC.5}$$

$$x_i \geq 0 \qquad \forall i \in V \tag{EC.6}$$

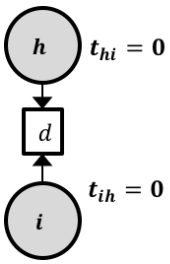
$$y_{id}, y_{di} \geq 0 \qquad \forall \{i,d\} \in E_t \tag{EC.7}$$

The dual of LP3 is as follows:

$$\text{(DLP3) Maximize} \quad \sum_{i \in V} g_i w_i - \sum_{\{i,d\} \in E_t} v_{id} \tag{EC.8}$$



(a) **t variables.**



(b) **Condition (EC.16) when** $x_i \neq y_{dh}$.

Figure EC.4

$$\text{Subject to: } (x_i)\ \textstyle\sum_{j\in n(i)} t_{ij} - \sum_{d\in a(i)} u_{id} + g_i w_i \le b_i \quad \forall i \in V \tag{EC.9}$$

$$(y_{id}) \qquad u_{id} - v_{id} \le 0 \qquad \forall i \in V, d \in a(i) \tag{EC.10}$$

$$(y_{di}) \qquad -t_{ji} - v_{id} + w_i \le 0 \qquad \forall d \in D, i \in a(d) \tag{EC.11}$$

$$t_{ij} \ge 0 \qquad \forall i \in V,\ j \in n(i) \tag{EC.12}$$

$$u_{id} \ge 0 \qquad \forall \{i,d\} \in E_t \tag{EC.13}$$

$$w_i \ge 0 \qquad \forall i \in V \tag{EC.14}$$

We have $\mathbf{t}$, $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{w}$ 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:

$$(u_{id} - v_{id}) y_{id} = 0 \qquad \forall i \in V, d \in a(i) \tag{EC.15}$$

$$(x_i - y_{dj}) t_{ij} = 0 \qquad \forall i \in V,\ j \in n(i) \tag{EC.16}$$

$$(y_{id} - x_i) u_{id} = 0 \qquad \forall i \in V, d \in a(i) \tag{EC.17}$$

$$(g_i - \textstyle\sum_{d\in n(i)} y_{di} - g_i x_i) w_i = 0 \qquad \forall i \in V \tag{EC.18}$$

$$(b_i - \textstyle\sum_{j\in n(i)} t_{ij} + \sum_{d\in a(i)} u_{id} - g_i w_i) x_i = 0 \quad \forall i \in V \tag{EC.19}$$

$$(-t_{ji} - v_{id} + w_i) y_{di} = 0 \qquad \forall d \in D, i \in a(d) \tag{EC.20}$$

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

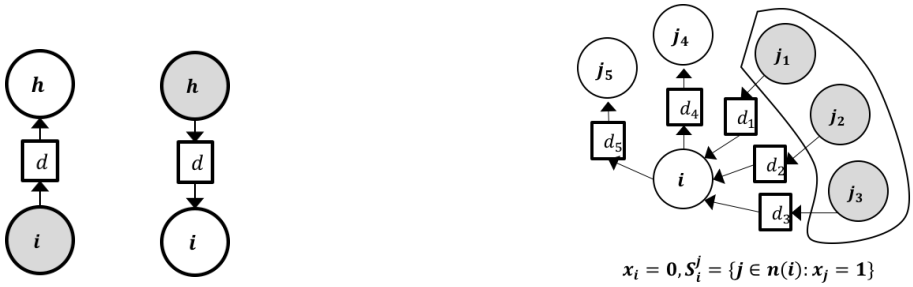


Figure EC.5 (*a*) **Case 1: A leaf node** i **has** $x_i = y_{id}$. (*b*) **Case 2:** $x_i = 0$.

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_i^1, b_i^2, X_i^{NPS}, C_i^{NPS}, X_i^{PS}, and C_i^{PS} are obtained in the DP in Section EC.1. We set $t_{ih} = b_i^2$ and $t_{hi} = b_i^1$, 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, $b_i^1 = b_i^2 = 0$. 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 $b_i^1 = b_i$ and $b_i^2 = 0$. 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 $S_i^j = \{j \in n(i) : x_j = 1\}$, which denotes the set of nodes that are selected and adjacent to node i in the original graph, and $S_i^d = \{d \in a(i) \cap a(j) : j \in S_i^j\}$, which denotes the set of dummy nodes adjacent to node i and the nodes in S_i^j. Then, let $w_i = \max\{t_{ji} : j \in S_i^j\}$, which is the biggest t_{ji} value among the nodes in S_i^j. Then, let $u_{id} = v_{id} = w_i - t_{ji}$ for all j in S_i^j 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^j. When $|S_i^j| > g_i$, nodes with

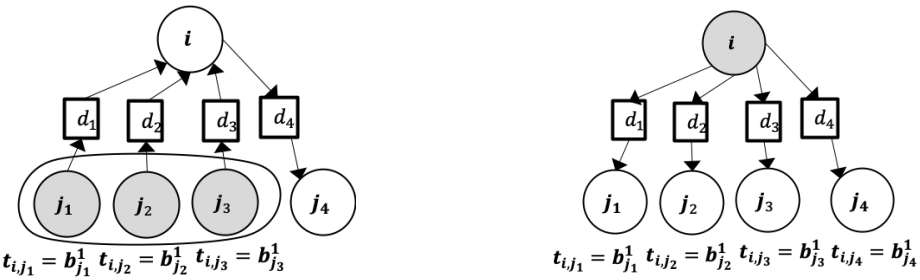


Figure EC.6 **Case 3:** (*a*) $X_i^{PS} \neq \{i\}$ **and** $|X_i^{PS}| = g_i - 1$. (*b*) $X_i^{PS} = \{i\}$.

positive t_{ji} can be removed from S_i^j 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:

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

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

$$= \begin{cases} b_i & \text{(Note: } |S_i^j| = g_i \text{ or } w_i = 0 \text{ when } |S_i^j| > g_i) \\ b_i & \text{(Note: } b_i = C_i^{PS} + b_i^1 + b_i^2 \text{ based on the definitions of } b_i^1 \text{ and } b_i^2) \end{cases} = b_i.$$

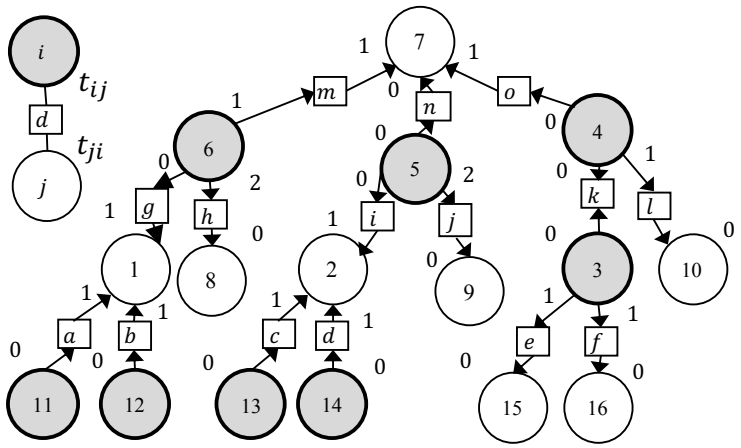
Thus, condition (EC.19) is satisfied because constraint (EC.9) is respected and $x_i = 0$. Condition (EC.20) is satisfied because constraint (EC.11) is binding for all d in S_i^d and $y_{di} = 0$ for all d in $a(i) \setminus S_i^d$.

Case 3: Suppose that node i is not a leaf node in G and $x_i = 1$. It means that $y_{id} = 1$ 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^{NPS} is $\{i\}$ or has $|X_i^{NPS}| = g_i$. Set w_i as the g_ith smallest t_{ji} value for all j in $n(i)$. Then, set $u_{id} = v_{id} = \max\{w_i - t_{ji}, 0\}$ for all d in $a(i)$. Thus,

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

Then,

$$LHS = \begin{cases} B_i^{NPS} + b_i^2 + \sum_{j \in X_i^{NPS}} b_j^2 & \text{if } X_i^{NPS} \neq \{i\} \\ B_i^{NPS} + b_i^2 + \sum_{j \in X_i^{PS}} b_j^2 + b_i^1 & \text{if } X_i^{NPS} = \{i\} \end{cases} = C_i^{NPS} + b_i^2 = b_i$$



$t_{1,10}=1$	$t_{6,7}=1$	$u_{3,e}=1$
$t_{1,11}=1$	$t_{7,6}=1$	$w_4=1$
$t_{1,6}=1$	$t_{7,4}=1$	$u_{4,k}=1$
$t_{2,13}=1$	$w_8=2$	$u_{4.l}=1$
$t_{2,14}=1$	$w_9=2$	$u_{5,i}=1$
$t_{2,5}=1$	$w_{10}=1$	$w_6=1$
$t_{3,15}=1$	$w_{11}=1$	$u_{6,h}=1$
$t_{3,16}=1$	$w_{12}=1$	$w_7=1$
$t_{3,4}=1$	$w_{13}=1$	$u_{7,n}=1$
$t_{4,10}=1$	$w_{14}=1$	$u_{7,o}=1$
$t_{5,9}=2$	$w_{15}=1$	
$t_{6,8}=2$	$w_{16}=1$	

Figure EC.7 (a) A PIDS problem instance for Theorem 2. (b) Non-zero dual variable values except v.

The second to the last equality holds because $C_i^{NPS}=B_i^{NPS}+\sum_{j\in X_i^{NPS}} b_j^2$ when $|X_i^{NPS}|=g_i$ and $C_i^{PS}=B_i^{NPS}+\sum_{j\in X_i^{PS}} b_j^2$ and $C_i^{NPS}=C_i^{PS}+b_i^1$ when $X_i^{NPS}=\{i\}$. The last equality follows from the definition of b_i^2.

Second, suppose that node i is selected in the Parent-Selected solution ($X_i^{PS}=\{i\}$, as shown in Figure EC.6(b). It implies that $X_i^{NPS}=X_i^{PS}$. If $LHS<b_i$, let $w_i=\max\{t_{ji}:j\in S_{g_i-1}\}$. Set $u_{id}=v_{id}=w_i-t_{ji}=w_i-b_j^2$ for all j in S_{g_i-1} and d in $a(i)\cap a(j)$. Thus, $LHS=B_i^{NPS}+\sum_{j\in S_{g_i-1}} b_j^2+w_i>b_i$, as node i is selected in X_i^{PS} by the DP algorithm. At this point, we ensure that $LHS\geq b_i$. Then, if $LHS>b_i$, select a $\tilde{d}$ in $a(i)$ and increase its $u_{i\tilde{d}}$ and $v_{i\tilde{d}}$ by $LHS-b_i$, i.e., $u_{i\tilde{d}}=v_{i\tilde{d}}=u_{i\tilde{d}}+LHS-b_i$. Thus, conditions (EC.19) is satisfied because constraint (EC.9) is binding by construction. Conditions (EC.20) is satisfied because $y_{di}=0$ and constraint (EC.11) is respected.

□

Figure EC.7(a) is the transformed graph and its solution based on the instance in Figure EC.3. On the right part of Figure EC.7, it has non-zero dual variables, except for v variables because $u_{id}=v_{id}$. First of all, we set up all t variables. They are shown in the first two columns in Figure EC.7(b). 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, $|S_1^j|=3>1=g_1$. Thus, $w_1=0$. For node 2, $|S_2^j|=3>2=g_2$. Thus, $w_2=0$. For node 7, $|S_7^j|=3=g_7$. 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 $w_4=1=t_{7,4}$, $u_{4,k}=1-t_{3,4}=1$ and $u_{4,l}=1-t_{10,4}=1$. 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 $w_6=1$ and $u_{6,h}=1$. The total objective value is 12, which is exactly the same as that of the solution obtained by the DP.

EC.2.2. Proof of Lemma 1

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. $\square$

EC.2.3. Proof of Lemma 2

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_{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. $\square$

EC.2.4. Proof of Lemma 3

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 ($deg(j) - |n(j) \cap S| \geq g_j$ for all j in S) and Lemma 2 ($|S_{\text{all}}| \leq a_i$). $\square$

EC.2.5. Proof of Lemma 4

Equation (29) minus the jth 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 $C_{S \setminus S_{\text{all}}}^{a_i - |S_{\text{all}}|}$, equation (31) minus the jth equation from the equation set (32) yields $u_j = 0$ for all j in $\{V \setminus (S^+ \cup N_{\text{all}} \cup H_C)\}$. If a node $j \in H_1$ does not belong to H_C for some $C \in C_{S \setminus S_{\text{all}}}^{a_i - |S_{\text{all}}|}$, then $j \in V \setminus (S^+ \cup N_{\text{all}} \cup H_C)$ 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 $C \in C_{S \setminus S_{\text{all}}}^{a_i - |S_{\text{all}}|}$, then $\gamma_j \leq a_i$. Otherwise, if $\gamma_j > a_i$ it cannot be removed from any $\mathbf{x}_0^C$ because we can only select a_i nodes from S if $x_i = 0$ (i.e., j is in all sets H_C for $C \in C_{S \setminus S_{\text{all}}}^{a_i - |S_{\text{all}}|}$). $\square$

EC.2.6. Proof of Lemma 5

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

$$\mu_i = \mu_0 \tag{EC.21}$$

$$\sum_{j \in S_{\text{all}}} \mu_j + \sum_{j \in C} \mu_j = \mu_0 \ \forall C \in C_{S \setminus S_{\text{all}}}^{a_i - |S_{\text{all}}|} \tag{EC.22}$$

We know that $C_{S\setminus S_{\text{all}}}^{a_i-|S_{\text{all}}|}$ is the set of all combinations for choosing $a_i - |S_{\text{all}}|$ nodes from the set $S \setminus S_{\text{all}}$. Thus, $\mu_j = \frac{\mu_0}{a_i}$ for all j in S is a solution to (EC.21) and (EC.22). However, it is not the unique solution. Depending on the value of $\sum_{j\in S_{\text{all}}} \mu_j$, we can have infinitely many solutions by setting $u_j = \frac{\mu_0 - \sum_{j'\in S_{\text{all}}} \mu_{j'}}{a_i - |S_{\text{all}}|}$ for all j in $S \setminus S_{\text{all}}$. For the uniqueness of the desired $(\pi_0, \boldsymbol{\pi})$, we need two things. First, $S_{\text{all}} = \emptyset$. Thus, the solution does not depend on the value of $\sum_{j\in S_{\text{all}}} \mu_j$ anymore. Second, when $|S| \geq 2$, we need $a_i \leq |S| - 1$. Then, from $C_{S\setminus S_{\text{all}}}^{a_i-|S_{\text{all}}|}$, 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., $C_1 \setminus j_1 = C_2 \setminus j_2$, $C_1 \setminus C_2 = j_1$ and $C_2 \setminus C_1 = j_2$). Take their corresponding (EC.22)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, $u_j = \frac{\mu_0}{a_i}$ for all j in S. Thus, $\pi_i = \pi_0 = a_i$ and $\pi_j = 1$ for all j in S. $\square$