Influence Maximization with Latency Requirements on Social Networks

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

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

Targeted marketing strategies are of significant interest in the smartapp economy. Typically, one seeks to identify individuals to strategically target in a social network so that the network is influenced at a minimal cost. In many practical settings the effects of direct influence predominate, leading to the Positive Influence Dominating Set with Partial Payments (PIDS-PP) problem that we discuss in this paper. The PIDS-PP problem is NP-complete, as it generalizes the dominating set problem. We discuss several mixed integer programming formulations for the PIDS-PP problem. First, we describe two compact formulations on the payment space. We then develop a stronger compact extended formulation. We show that when the underlying graph is a tree, this compact extended formulation provides integral solutions for the node selection variables. In conjunction, we describe a polynomial-time dynamic programming algorithm for the PIDS-PP problem on trees. We project the compact extended formulation onto the payment space, providing an equivalently strong formulation that has exponentially many constraints. We present a polynomial time algorithm to solve the associated separation problem. Our computational experience on a test-bed of 100 real-world graph instances (with up to approximately 465,000 nodes and 835,000 edges) demonstrates the efficacy of our strongest payment space formulation. It finds solutions that are on average 0.4% from optimality, and solves 80 out of the 100 instances to optimality.

Key words: social networks, influence maximization, latency constraints, mixed integer programming, strong models

1. Introduction

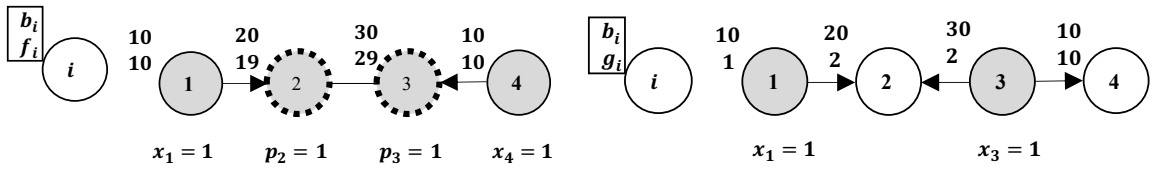
Mobile applications and social networks play an important role in the spread of influence, information, and ideas in people's daily lives. Although researchers have acknowledged social influence as a component in decision making for a long time (see for example Bourne 1957, Brown and Reingen 1987, Granovetter 1978), online social media help trace the dynamics easily (see Valente 2012). Indeed they provide an unprecedented platform for understanding and reaping the benefits of direct targeting. This has resulted in the social media influencer marketing segment being a thriving industry with an estimated worth somewhere between $5 billion to $16 billion in 2020 (Schmidt 2019), and predictions of value up to $15 billion by 2022 (Schomer 2020).

In direct targeting the goal is to identify a set of nodes to target in a network, with the objective that these directly targeted nodes influence the rest of the network. Although influence can

theoretically propagate through a network, there is a significant difference between the magnitude of the effects of "direct" and "indirect" influence in practice. "Direct influence" refers to influence received from a node that has been selected for targeting, and "indirect influence" refers to influence received from a node that has not been selected for targeting (i.e., it has been influenced by its neighbors and subsequently influences its uninfluenced neighbors). Indeed, Goel et al. (2015) investigated the diffusion of nearly a billion news stories, videos, pictures, and petitions on Twitter and showed that almost all of the diffusion (over 99%) consisted of direct influence. Furthermore, Zhang et al. (2018) studied the diffusion of technology adoption in the context of caller ringback tones (CRBT) on a data set of 200 million calls among 1.4 million users, and found that the adoption of CRBT is consistently predicted solely by direct influence (indicating that the magnitude of direct influence is statistically much larger than that of indirect influence). The predominance of direct influence is seen in many different settings including medical technology diffusion (Coleman et al. 1966) and also in settings where behavioral change is desired for medical reasons (Greaves et al. 2010) within a target population.

Motivated by the importance of direct influence in targeted marketing, in this paper we focus our attention on the *Positive Influence Dominating Set with Partial Payments* (PIDS-PP) problem. Given a social network represented as an undirected graph $G = (V, E)$, each node $i \in V$ has a threshold b_i (representing the cost to the marketer to directly target the node) and an influence factor f_i (representing the amount of influence each of its directly targeted neighbors exerts on it). A node i that is not directly targeted can be influenced by the payment of an amount $p_i = \max(0, b_i - |B_i|f_i)$, where B_i represents the set of neighbors of node i that have been directly targeted by the marketer. The goal is to select a subset $T \in V$ (a PIDS-PP) that is directly targeted, so that the sum of the threshold (or full) payments to the selected nodes (T) and the partial payments to the remaining nodes ($V \backslash T$) is minimized. Notice that when $b_i = f_i = 1$ for all $i \in V$, we obtain the classic dominating set problem (see Haynes et al. 1998a,b, for a survey). Thus, the PIDS-PP problem generalizes the dominating set problem, and is NP-hard.

The influence factor f_i implicitly assumes that the identity of a neighbor has no impact on its influence (i.e., all neighbors of a node have equal influence on it). This is motivated by the fact that privacy concerns are a significant issue on social networks (and has received attention from regulators globally). Note however, although a node is equally influenced by each neighbor, this influence factor may be different for each node in the network. We also note that a node that receives a partial payment needs to receive influence from at least one or more of its neighbors in order to be influenced. Thus nodes that receive partial payments do not belong to the class of directly targeted nodes, and any influence they propagate is indirect influence. Since the PIDS-PP



(a) PIDS-PP Problem Instance (b) Equivalent PIDS Problem Instance

Figure 1 Comparing the PIDS-PP and the PIDS Problem

problem is focused solely on direct influence (given the orders of magnitude difference in direct and indirect influence observed in practice), these indirect influence effects are not modeled.

The PIDS-PP problem is closely related to two problems previously studied in the literature. When partial payments are not permitted (i.e., $p_i = 0$ for all $V \backslash T$), we obtain the Positive Influence Dominating Set (PIDS) problem (see Wang et al. 2009), which requires that a node i that is not directly targeted receives influence from $g_i = \lceil b_i / f_i \rceil$ directly targeted neighbors. Figure 1(a) shows a PIDS-PP instance. A shaded node with solid outline denotes that a node is directly target. A shaded node with dashed outline denotes that a node receives partial payment. A directed arc shows the influence direction. An optimal solution has $x_1 = 1, p_2 = 1, p_3 = 1, x_4 = 1$ with an objective value of 22. By calculating g_i for each node, we converting this PIDS-PP instance into an equivalent PIDS instance in Figure 1(b). Given that partial payments are not allowed, an optimal solution is to select nodes 1 and 3 ($x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 0$) with an objective value of 40. Comparing to the PIDS-PP instance, the total cost increases by over 81%. It demonstrates the potential cost savings by considering partial payments in the PIDS-PP problem. Further, when indirect influence plays a larger and equivalent role in influence propagation, influence can propagate along paths in the network through nodes that have been influenced but were not directly targeted. In this case, we obtain the Least Cost Influence Problem (LCIP)(see Günneç et al. 2020). Thus, in a certain sense, one can also view the PIDS-PP problem as an influence maximization problem (i.e., the LCIP) with a latency requirement (of one time-period).

1.1. Our Contributions and Organization of the Paper

We review the related literature in Section 2 and describe the relationship between the PIDS-PP problem and other influence maximization problems. Section 3 describes three mixed-integer programming (MIP) formulations for the PIDS-PP problem. The first formulation MIP1, on the payment space, is closely related to one for the dominating set problem. Next, we strengthen MIP1, by adding a set of valid inequalities that are based on the observation for a node that is not directly targeted and receives no partial payment; one of its directly targeted neighbors gives it "lower" (L) influence. Adding these L-type constraints to MIP1, we obtain a stronger MIP formulation MIP1L. Then, we add arc variables to MIP1L and apply an edge-splitting idea that strengthens

the formulation, to obtain our third MIP formulation MIP2. We show that the linear relaxation of MIP2 provides integral solutions for the node selection variables on trees (thus proving that it is the strongest possible formulation for the PIDS-PP problem on trees). To complement this result, we present a polynomial-time dynamic programming algorithm in Appendix EC.2 of the electronic companion for the PIDS-PP problem on trees. Section 4 projects the compact extended formulation MIP2 onto the space of the payment variables, giving rise to a formulation MIP3 on the payment space with two exponentially sized sets of valid inequalities whose polynomial-time separation is discussed. Section 5 reports on a computational study regarding the efficacy of the proposed MIP formulations. The projected formulation (on the payment space), MIP3, finds high-quality solutions (within a 0.4% gap) for very large instances with up to approximately 465,000 nodes and 835,000 edges. Section 6 provides concluding remarks.

2. Related Literature

Social network analytics have introduced several interesting combinatorial optimization problems to the research community. These include 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). Influence maximization on social networks has stimulated a stream of recent research work (Fischetti et al. 2018, Wu and Küçükyavuz 2018, Li et al. 2019, Günneç et al. 2020, Raghavan and Zhang 2019, Nannicini et al. 2020).

Kempe et al. (2003) were the first researchers to consider the influence maximization problem (IMP) in an operational framework, employing models from mathematical sociology (Granovetter 1978) that explicitly represent the step-by-step dynamics of influence propagation. 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) and showed that it is NP-hard in order to find the optimal target set. This seminal work led to a flurry of follow-up studies on the problem and its variants. The comprehensive reviews by Chen et al. (2013), Li et al. (2018), and Banerjee et al. (2020) nicely summarize the most relevant work on this topic in a probabilistic setting.

From the computational perspective, while Kempe et al. (2003) showed that a simple greedy algorithm has an $(1 - 1/e - \epsilon)$-approximation ratio for the IMP, it requires a costly simulation in each step. Consequently, a significant amount of effort has been made on speeding up the procedure. The first major breakthrough is the Cost Effective Lazy Forward (CELF) algorithm in Leskovec et al. (2007). In the CELF algorithm and its improvement CELF++ (Goyal et al. 2011), after the selection of the first node, the costly simulation step is restricted to the top candidate whenever it is possible. Thus, it speeds up the procedure by orders of magnitude. Recently, a series

of algorithms based on Reverse Reachable Sets (RRS) has been developed for the IMP (Borgs et al. 2014, Tang et al. 2015, Nguyen et al. 2016). The key idea of RRS is to intelligently generate a set of influence propagation realizations a priori based on the live-edge graphs in Kempe et al. (2003). Then, the solution of the IMP is obtained by solving a maximum coverage problem in one shot. This leads to a tremendous advantage in terms of running time. All these heuristics rely on stylized technical assumptions in the IMP that result in a submodularity property, that is not possessed in a deterministic setting including the APX-hard PIDS-PP problem (the PIDS-PP problem is APX-hard since the dominating set problem is APX-hard, Chlebík and Chlebíková 2008). For exact methods, Wu and Küçükyavuz (2018) studied the IMP problem in a two-stage stochastic programming framework. Nannicini et al. (2020) considered a robust version of the IMP and presented a novel mixed integer programming formulation, derived from a bilevel formulation of the problem.

Initiated by Chen (2009), another stream of follow-up work has focused on the problem in a deterministic setting with a cost minimization aspect (i.e., instead of the marketer being given a budget k, the desire is to find the minimum number of nodes to target in the network so that the entire network is influenced). In the Target Set Selection (TSS) problem, given a connected undirected graph, each node has associated with it a critical value g_i, which takes values between 1 and the degree of the node, denoted by $deg(i)$. All nodes are inactive initially. A selected subset of nodes, the target set, are activated (i.e., switched to an active state). Next, the states of the nodes are updated step by step with respect to the following rule: an inactive node i becomes active if at least g_i of its neighbors are active in the previous step. The goal is to find the minimum size target set while ensuring that all nodes are active by the end of this diffusion process. Chen (2009) showed that the TSS problem is hard to approximate within a polylogarithmic factor. Raghavan and Zhang (2019) introduced the *weighted* TSS (WTSS) problem. In the WTSS problem, for each node $i \in V$, there is a weight, denoted by b_i, which models the fact that different nodes require differing levels of effort to become active.

Günneç and Raghavan (2017) described 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. Günneç et al. (2020) addressed the complexity of the LCIP for a variety of special cases. When the entire network must be influenced and the neighbors of a node exert equal influence, they showed that the LCIP on trees is polynomially solvable by describing two polynomial time algorithms as well as a totally unimodular (TU) formulation. Building upon the TU formulation, Günneç et al. (2020) proposed a branch-and-cut approach for solving the LCIP on arbitrary graphs. 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.

Wang et al. (2009) proposed the Positive Influence Dominating Set problem. In their variant (and many of the other papers in the literature), it is assumed that a node i in the graph not selected in the PIDS needs at least half of its neighbors in the PIDS (i.e., if a node is not selected for direct targeting, then it needs at least half of its neighbors to be directly targeted). They also presented and tested an iterative greedy selection algorithm for the PIDS problem with a real-world online social network data set. Zhu et al. (2010) 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 and heuristics have been proposed for the PIDS problem by Wang et al. (2011), Dhawan and Rink (2015), Khomami et al. (2018), and Lin et al. (2018). 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). Raghavan and Zhang (2017) further generalized the PIDS problem. They introduced the weighted version, and each node can require any positive number of neighbors to be in the PIDS (as opposed to a fixed number or a fixed proportion of neighbors that is the same for all nodes in the graph). They presented an integer programming formulation for the PIDS problem, and showed that (i) it contains a set of facet defining inequalities, and (ii) provides the PIDS polytope on trees. They described their computational experience with this formulation in a branch-and-cut setting on large real-world graphs.

3. A Strong and Compact Extended Formulation

In this section, we describe three MIP formulations for the PIDS-PP problem. Typically, a combinatorial optimization can be formulated in many different ways. One common way (see Conforti et al. 2014, Nemhauser and Wolsey 1988) to compare these different formulations for the same problem is to solve their LP relaxations. Then, these formulations are evaluated by the optimal objective values of their LP relaxations. Given two different formulations (A and B) of a given minimization problem, let z_A^{LP} and z_B^{LP} be the optimal objective values of their LP relaxations, respectively. We say that formulation A is stronger (or tighter) than formulation B if $z_A^{LP} > z_B^{LP}$. A stronger formulation is often more computationally efficient (Barnhart et al. 1993).

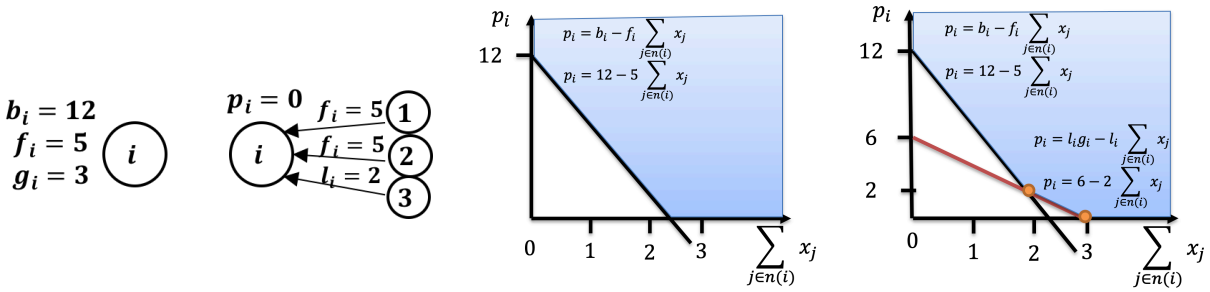


Figure 2 (*a*) **A PIDS-PP Problem Instance** (*b*) **No L-type** (*c*) **With L-Type**

Our first MIP formulation, for the PIDS-PP problem uses a binary variable x_i (for each $i \in V$) to denote whether node i is selected for direct targeting by receiving full payment (b_i). The non-negative variable p_i represents the partial payment that node i receives if it is not selected for targeting. Let $n(i)$ denote the set of node i's neighbors. The formulation MIP1 is as follows:

$$\text{(MIP1)} \qquad \text{Min} \quad \sum_{i \in V} p_i + \sum_{i \in V} b_i x_i \tag{1}$$

$$\text{Subject to} \quad p_i + b_i x_i + f_i \sum_{j \in n(i)} x_j \geq b_i, \qquad \forall i \in V, \tag{2}$$

$$x_i \in \{0,1\},\ p_i \geq 0, \qquad \forall i \in V. \tag{3}$$

The objective function (1) is used to minimize the total cost over the network. Constraint set (2) models the diffusion process—either a node is selected for direct targeting (i.e., paid b_i), or the sum of the partial payment (p_i) and the total influence coming from its neighbors that have been directly targeted is at least b_i. Although p_i has an upper bound of $b_i - f_i$, it is not necessary to enforce it. If $b_i - f_i < p_i < b_i$, p_i can be reduced to $b_i - f_i$, resulting in a lower cost solution, and if $p_i = b_i$, we can construct a feasible solution with an equal or lower cost by setting $x_i = 1$ and $p_i = 0$ (the cost can be lower because the node i is now selected for direct targeting and can influence its neighbors). Thus, we note that in an optimal solution, at most one of x_i and p_i takes a positive value for any node i in V. Notice that because the x variables are binary, when the data are integral, the p variables are automatically an integer. With a slight abuse of notation, from here on, we refer to the $(\mathbf{p}, \mathbf{x})$ space as the payment variable space.

As noted earlier, when $b_i = f_i = 1$ we obtain the dominating set problem. Our discussion shows that for the dominating set problem, $0 \leq p_i \leq b_i - f_i = 0$. In other words, the p variables can be dropped from MIP1 for the dominating set problem, yielding a formulation described by Saxena (2004). He showed that this formulation is integral on trees (i.e., its LP relaxation provides integral solutions). Although MIP1 is a valid formulation for the PIDS-PP problem, as we will demonstrate in Section 5 this formulation is weak. Our goal is to build a stronger formulation for the PIDS-PP problem. We start by making the following observation.

Observation 1: Suppose that a node i in V does not receive any payment. Then, it has at least g_i neighbors that are directly targeted, where $g_i = \left\lceil \frac{b_i}{f_i} \right\rceil$. Among these directly targeted neighbors, $(g_i - 1)$ of them give it a total of $f_i(g_i - 1)$ units of influence, with each of these directly targeted neighbors providing it with f_i units of influence. Furthermore, there is one directly targeted neighbor that provides it with l_i units of influence, where $l_i = b_i - f_i(g_i - 1)$.

Consider the example in Figure 2(a) for illustration. This node has $b_i = 12$ and $f_i = 5$. Thus, its $g_i = 3$. Using constraint (2), Figure 2(b) plots (in two dimensions) p_i as a function of $\sum_{j \in n(i)} x_j$ under the assumption that node i node is not directly targeted (i.e., $x_i = 0$). The feasible values of p_i and $\sum_{j \in n(i)} x_j$ are shaded in this two dimensional plot (under the assumption that they are both non-negative). Notice that $\sum_{j \in n(i)} x_j$ can be fractional in constraint (2), with a value of 2.4 instead of the correct value of 3 to make $p_i = 0$ (in this two dimensional illustration we have a fractional extreme point in the $(p_i, \sum_{j \in n(i)} x_j)$ space). In order to strengthen this formulation, we add the following set of inequalities, which we refer to as L-type constraints:

$$\text{(L-type constraints)} \quad p_i + l_i g_i x_i + l_i \textstyle\sum_{j \in n(i)} x_j \geq l_i g_i, \quad \forall i \in V. \tag{4}$$

PROPOSITION 1. *L-type constraints are valid for the PIDS-PP problem.*

Proof: Given a solution $(\hat{x}, \hat{p})$ of the PIDS-PP problem in terms of MIP1, it must satisfy constraint (2). Then, for a node i, if $\hat{x}_i = 1$, its L-type constraint is satisfied. If $\hat{x}_i = 0$, $\hat{p}_i \geq b_i - f_i \sum_{j \in n(i)} \hat{x}_j = l_i + f_i(g_i - 1 - \sum_{j \in n(i)} \hat{x}_j) \geq l_i + l_i(g_i - 1 - \sum_{j \in n(i)} \hat{x}_j) = l_i g_i - l_i \sum_{j \in n(i)} \hat{x}_j$ because $b_i = l_i + f_i(g_i - 1)$ and $f_i \geq l_i > 0$. $\square$

Figure 2(c) demonstrates the effect of the L-type constraint, when viewed in two dimensions in the $(p_i, \sum_{j \in n(i)} x_j)$ space. It cuts off the fractional extreme point, where $\sum_{j \in n(i)} \hat{x}_j = 2.4$, and introduces two integer extreme points, where $\sum_{j \in n(i)} \hat{x}_j = 2$ or $\sum_{j \in n(i)} \hat{x}_j = 3$. Adding L-type constraints to MIP1, we obtain a stronger formulation, which we refer to as MIP1L:

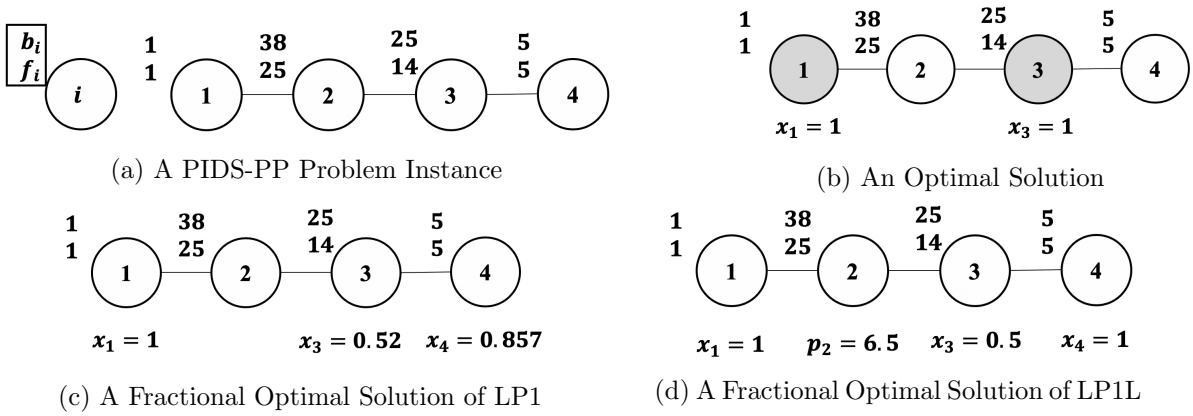
$$\text{(MIP1L)} \quad \text{Min} \quad \textstyle\sum_{i \in V} p_i + \sum_{i \in V} b_i x_i \tag{5}$$

$$\text{Subject to} \quad p_i + b_i x_i + f_i \textstyle\sum_{j \in n(i)} x_j \geq b_i, \quad \forall i \in V, \tag{6}$$

$$p_i + l_i g_i x_i + l_i \textstyle\sum_{j \in n(i)} x_j \geq l_i g_i, \quad \forall i \in V, \tag{7}$$

$$p_i \geq 0,\ x_i \in \{0, 1\}, \quad \forall i \in V. \tag{8}$$

Figure 3(a) provides a PIDS-PP problem instance whose underlying graph is a tree, Figure 3(b) shows that nodes 1 and 3 must be directly targeted in the optimal solution that has an objective value of 26. Figure 3(c) describes a fractional optimal solution to the LP relaxation of MIP1,



(a) A PIDS-PP Problem Instance

(b) An Optimal Solution

(c) A Fractional Optimal Solution of LP1

(d) A Fractional Optimal Solution of LP1L

Figure 3 A PIDS-PP Problem Example for Comparing the LP Relaxations

denoted by LP1. It has $x_1 = 1$, $x_3 = 0.52$, and $x_4 = 0.857$; all other decision variables are zero, and the objective value is 18.286. Figure 3(d) describes a fractional optimal solution to the LP relaxation of MIP1L, denoted by LP1L. It has $x_1 = 1$, $p_2 = 6.5$, $x_3 = 0.5$, and $x_4 = 1$; all other decision variables are zero, and the objective value is 25. Although LP1L is stronger than LP1 for this instance, neither of them provide integral solutions for this tree instance. (Recall that LP1 without the p variables is integral on trees for the dominating set problem.)

We now create an extended formulation by adding arc variables to the formulation. For each edge $\{i, j\}$ in E, we define y_{ij} and y_{ji} as binary variables representing whether (when equal to 1) the directly targeted node i influences its neighbor j, and whether the directly targeted node j influences its neighbor i, respectively. We can add the inequality:

$$x_i \geq y_{ij}, \quad \forall i \in V, j \in n(i). \tag{9}$$

Additionally, since y_{ji} tells us whether a directly targeted neighbor of i influences it, we can replace $\sum_{j \in n(i)} x_j$ in MIP1 and MIP1L by $\sum_{j \in n(i)} y_{ji}$.

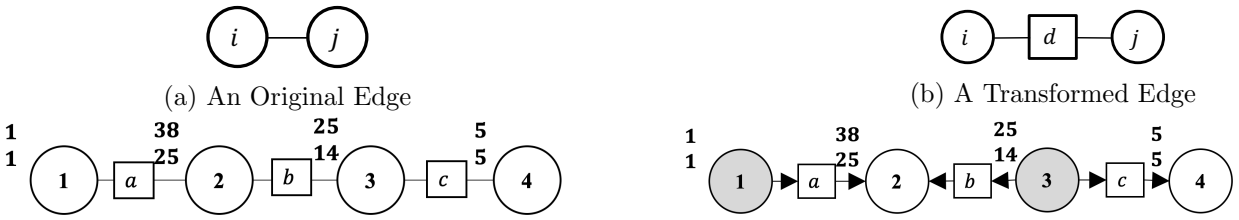
Observation 2: In a feasible solution to the PIDS-PP problem, neighboring nodes do not influence each other. Therefore, we can add the following set of inequalities in the extended formulation:

$$y_{ij} + y_{ji} \leq 1, \quad \forall \{i, j\} \in E. \tag{10}$$

Observation 3: Ideally, if a node i is directly targeted (i.e., $x_i = 1$), it should send influence to all of its neighbors that are not directly influenced (i.e., $y_{ij} = 1$ for all j in $n(i)$ with $x_j = 0$). One might try to accomplish this by adding the following set of inequalities:

$$x_i \leq y_{ij}, \quad \forall i \in V, j \in n(i). \tag{11}$$

Unfortunately, constraint (11) is *not* valid for the PIDS-PP problem when a directly targeted node has a directly targeted node as a neighbor (as it conflicts with constraint 10).



(a) An Original Edge (b) A Transformed Edge

(c) The Transformed Graph for the Instance in Figure 3 (d) The Directed Arcs in the Transformed Graph

Figure 4 Illustration of the Edge-Splitting Idea

To work around this, and obtain the strengthening benefit of constraint (11) without violating constraint (10), we use the following edge-splitting idea. From the input graph G, we create a new graph G_t by adding one dummy node d to each edge $\{i, j\}$ in G. Let D denote the set of dummy nodes. Since the dummy nodes have effectively split each edge into two, we replace each of the original edges $\{i, j\} \in E$ by two edges $\{i, d\}$ and $\{d, j\}$ in the new graph G_t. The procedure is shown in Figure 4(a) and Figure 4(b). Let E_t denote the set of edges in G_t ($G_t = (V \cup D, E_t)$). Notice that G_t is bipartite, and E_t only contains edges between the nodes in V and D. Figure 4(c) shows the transformed graph of the example in Figure 3 based on this procedure. Dummy nodes are represented by rectangles. Figure 4(d) shows the influence direction by directed arcs in the transformed graph of the optimal solution shown in Figure 3(b).

By splitting an edge, the role of an outgoing arc from node i is played by arc (i, d), and the role of an incoming arc to node j is now played by arc (d, j). Specifically, applying this to constraint (11), $x_i \leq y_{ij}$, yields constraint (14), $x_i \leq y_{id}$. Similarly, constraint (9), $x_i \geq y_{ij}$, yields constraint (13), $x_i \geq y_{dj}$. Additionally, $\sum_{j \in n(i)} y_{ji}$ becomes $\sum_{d \in a(i)} y_{di}$, where $a(i)$ denotes the set of node i's neighbors in the transformed G_t, giving rise to constraints (15) and (16) from constraints (6) and (7) in MIP1L. From Observation 2, we can write $y_{id} + y_{di} = 1$, i.e., constraint (17), ensuring that on each edge influence is propagated in exactly one direction. We write it in the equality form (instead of the inequality form) because it is possible to explicitly direct every edge in G_t, without violating any other constraint. A second benefit of the equality form (as we will see later) is that we can remove constraint (17) and substitute out all y_{id} by $1 - y_{di}$ to improve the computing efficiency by reducing $|E_t|$ constraints and $|E_t|$ variables in the computational experiment. Putting all of the above observations together, we obtain the compact extended formulation MIP2.

$$\text{(MIP2)} \quad \text{Min} \quad \sum_{i \in V} p_i + \sum_{i \in V} b_i x_i \tag{12}$$

$$\text{Subject to} \quad x_i \geq y_{dj}, \qquad \forall i \in V, j \in n(i), \tag{13}$$

$$x_i \leq y_{id}, \qquad \forall i \in V, d \in a(i), \tag{14}$$

$$p_i + b_i x_i + f_i \sum_{d \in a(i)} y_{di} \geq b_i, \qquad \forall i \in V, \tag{15}$$

$$p_i + l_i g_i x_i + l_i \sum_{d \in a(i)} y_{di} \geq l_i g_i, \quad \forall i \in V, \tag{16}$$

$$y_{id} + y_{di} = 1, \quad \forall \{i,d\} \in E_t, \tag{17}$$

$$p_i \geq 0,\ x_i \in \{0,1\}, \quad \forall i \in V, \tag{18}$$

$$y_{id},\ y_{di} \in \{0,1\}, \quad \forall \{i,d\} \in E_t. \tag{19}$$

MIP2 results in a stronger formulation. Its LP relaxation obtains integral solutions for the example in Figure 3. In fact, MIP2 is the strongest possible formulation for the PIDS-PP problem on trees. Theorem 1 proves this result. First, we note in Proposition 2 that on trees the PIDS-PP problem can be solved in linear time via dynamic programming (DP). The proof to Theorem 1 uses the integral primal feasible solution to LP2 obtained by the DP algorithm in Appendix EC.2 of the electronic companion. Then, it constructs a dual feasible solution for the dual problem to LP2 and shows that this pair of primal and dual solutions satisfies the complementary slackness (CS) conditions.

PROPOSITION 2. *The PIDS-PP problem on trees can be solved in $O(|V|)$ time.*

Proof: See Appendix EC.2 of the electronic companion. □

THEOREM 1. *For trees, LP2 provides optimal solutions with x variables binary.*

Proof: See Appendix EC.3 of the electronic companion. □

4. A Strong Formulation on the Payment Space via Projection

In this section, the feasible region of the extended formulation LP2 is projected onto the space of the payment (i.e., p and x) variables by projecting out all arc (i.e., y) variables. This allows us to derive a strong formulation with p and x variables only. As will be evident in our computational experiments, this formulation has great advantages in computational efficiency (in terms of scaling up), compared to MIP2.

Using $y_{id} + y_{di} = 1$ in LP2, we first project out all y_{id} variables, setting them by $1 - y_{di}$, and obtain the following formulation, referred to as LP_d and whose feasible region is denoted as P_d.

$$(\text{LP}_d) \quad \text{Min} \quad \sum_{i \in V} p_i + \sum_{i \in V} b_i x_i \tag{20}$$

$$\text{Subject to} \quad x_j - y_{di} \geq 0, \quad \forall j \in V, i \in n(j), \tag{21}$$

$$-x_i - y_{di} \geq -1, \quad \forall i \in V, d \in a(i), \tag{22}$$

$$p_i + b_i x_i + \sum_{d \in a(i)} f_i y_{di} \geq b_i, \quad \forall i \in V, \tag{23}$$

$$p_i + l_i g_i x_i + \sum_{d \in a(i)} l_i y_{di} \geq l_i g_i, \quad \forall i \in V, \tag{24}$$

$$p_i,\ x_i \geq 0, \quad \forall i \in V, \tag{25}$$

$$y_{di} \geq 0, \quad \forall \{i,d\} \in E_t. \tag{26}$$

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

$$-t_{di} - u_{id} + f_i v_i + l_i w_i \leq 0, \quad \forall i \in V, d \in a(i), \tag{27}$$

$$t_{di},\, u_{id},\, v_i,\, w_i \geq 0, \quad \forall i \in V, d \in a(i). \tag{28}$$

Here, t_{di}, u_{id}, v_i and w_i are dual multipliers corresponding to constraints (21), (22), (23) and (24), respectively. If P_d is written in matrix notation as $\{(\mathbf{x}, \mathbf{p}, \mathbf{y}) : A\mathbf{x} + B\mathbf{p} + G\mathbf{y} \geq \mathbf{b}, (\mathbf{x}, \mathbf{p}, \mathbf{y}) \geq \mathbf{0}\}$, based on Balas and Pulleyblank (1983), then any feasible vector $(\mathbf{t}, \mathbf{u}, \mathbf{v}, \mathbf{w})$ to W, defines a valid inequality: $(\mathbf{t}, \mathbf{u}, \mathbf{v}, \mathbf{w})^T(A\mathbf{x} + B\mathbf{p}) \geq (\mathbf{t}, \mathbf{u}, \mathbf{v}, \mathbf{w})^T\mathbf{b}$ to the projection of P_d (in the space of the payment (p and x) variables). Furthermore, the projection of P_d is defined by the valid inequalities projected by the extreme rays of W. In Theorem 2, we identify the extreme rays of W. First, we 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 non-negative 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 that $\mathbf{r}^1$, $\mathbf{r}^2 \in R(\mathbf{r})$. A ray $R(\mathbf{r})$ is extreme if its direction vector $\mathbf{r}$ is extreme.

THEOREM 2. *The vector $\mathbf{r} = (\mathbf{t}, \mathbf{w}, \mathbf{u}, \mathbf{v}) \in W$ is extreme if and only if there exists a positive α such that one of the following four cases holds:*

1. *$t_{di} = \alpha$ for one $\{i, d\} \in E_t$. All other t, w, u, v are 0.*
2. *$u_{id} = \alpha$ for one $\{i, d\} \in E_t$. All other t, w, u, v are 0.*
3. *$v_i = \alpha$ for one $i \in V$. Then for $d \in a(i)$, either $t_{di} = f_i\alpha$ or $u_{id} = f_i\alpha$. All other t, w, u, v are 0.*
4. *$w_i = \alpha$ for one $i \in V$. Then for $d \in a(i)$, either $t_{di} = l_i\alpha$ or $u_{id} = l_i\alpha$. All other t, w, u, v are 0.*

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 t^1_{di} and t^2_{di}, all of the other components are 0. Then, $\mathbf{r^1}$, $\mathbf{r^2}$ are in $R(\mathbf{r})$. Thus, 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 components of $\mathbf{r}$ with a value of 0, its corresponding components in $\mathbf{r^1}$ and $\mathbf{r^2}$ are also 0. Given i and d, let q^k_{id}, $k = 1, 2$, represent the positive element between t^k_{di} and u^k_{id}, $k = 1, 2$, (since only one of the two can be positive in the four cases). Then, we have $v^1_i + v^2_i = 2\alpha$ and $q^1_{id} + q^2_{id} = 2f_i\alpha$, for all $d \in a(i)$. For a pair d_1 and d_2, we have $q^1_{id_1} > q^1_{id_2}$ if and only if $q^2_{id_1} < q^2_{id_2}$. However, constraint (27) stipulates that $f_i v^k_i \leq \min\{q^k_{id_1}, q^k_{id_2}\}$ for $k = 1, 2$. Thus, $q^k_{id_1} = q^k_{id_2} = f_i\alpha_k$, $k = 1, 2$, for all d_1, $d_2 \in a(i)$. Otherwise, either constraint (27) would be violated or we could only have $v^1_i + v^2_i < 2\alpha$. Therefore, $\mathbf{r^1}$, $\mathbf{r^2}$ are in $R(\mathbf{r})$. Thus, $\mathbf{r}$ is extreme.

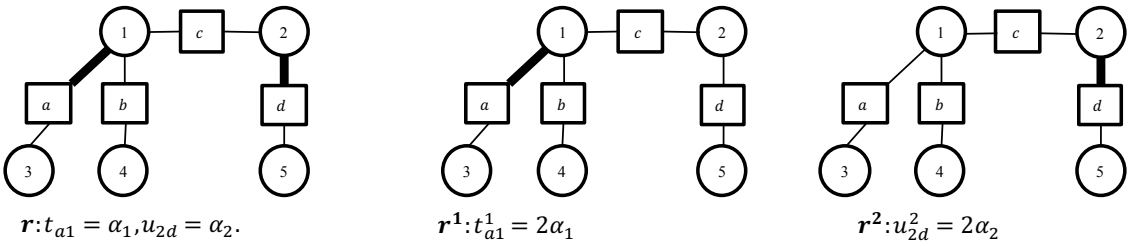


Figure 5 $S^v \cup S^w = \emptyset$ **and** $|S^t| + |S^u| > 1$**. Figure Shows that If** $S^v \cup S^w = \emptyset$**,** $\mathbf{r}$ **Must Satisfy Case 1 or Case 2.**

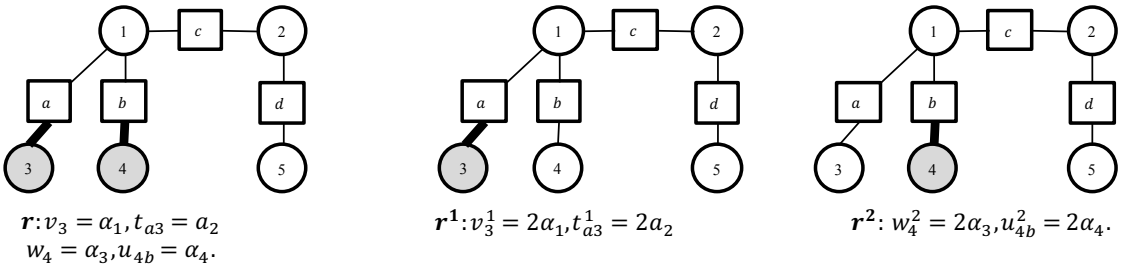


Figure 6 **When** $|S^v \cup S^w| > 1$**,** $\mathbf{r}$ **Is Not Extreme.**

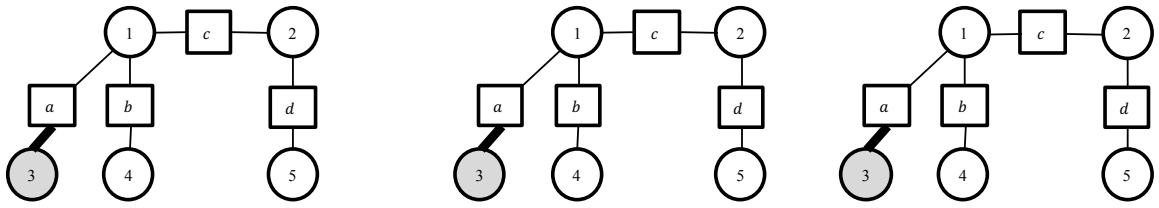
Case 4 is similar to Case 3.

Necessity. Let $\mathbf{r}$ be an extreme vector of W. Let $S^t = \{\{i,d\} \in E_t : t_{di} > 0\}$, $S^u = \{\{i,d\} \in E_t : v_{id} > 0\}$, $S^v = \{i \in V : v_i > 0\}$, and $S^w = \{i \in V : w_i > 0\}$ based on this $\mathbf{r}$. In the following proof, to prove that a given ray $\mathbf{r}$ is not extreme, 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 $\mathbf{2r - r^1}$. Then, $\mathbf{r} = \frac{1}{2}(\mathbf{r^1} + \mathbf{r^2})$ by design.

First, we consider extreme rays, where $\mathbf{v} = \mathbf{w} = \mathbf{0}$, meaning that $S^v \cup S^w = \emptyset$. If $|S^t| + |S^u| > 1$, let $\mathbf{r^1}$ contain all but one positive component in $S^t \cup S^u$ with their values doubled (then $\mathbf{r^2}$ contains the one positive component omitted by $\mathbf{r^1}$, with its value doubled). Thus, if $|S^t| + |S^u| > 1$, $\mathbf{r}$ is not extreme, contrary to the assumption. We conclude that if $S^v \cup S^w = \emptyset$, then $|S^t| + |S^u| = 1$ and thus, $\mathbf{r}$ must be of form Case 1 or Case 2. Figure 5 illustrates this situation. The bold line represents the positive t and u components in a vector r and the positive components are shown below the pictures.

Now, we consider extreme rays with positive $\mathbf{v}$ and $\mathbf{w}$. First consider the case where $S^v \cup S^w \neq \emptyset$. If $|S^v \cup S^w| > 1$, without loss of generality, let $i \in S^v \cup S^w$. Then, $\mathbf{r^1}$ has the value of $v^1_i = 2v_i$, $w^1_i = 2w_i$, $t^1_{di} = 2t_{di}$, $u^1_{id} = 2u_{id}$ for all $d \in a(i)$ and 0s for the other components. Thus, if $|S^v \cup S^w| > 1$, $\mathbf{r}$ is not extreme. Figure 6 illustrates this situation. The shaded nodes represent the positive v and w components in a vector $\mathbf{r}$.

Now consider the case where $|S^v \cup S^w| = 1$. If $S^v \cap S^w \neq \emptyset$, it means that there is a single node i in both S^v and S^w. Then, $\mathbf{r^1}$ can be set with a value of $v^1_i = 2v_i$, $t^1_{di} = 2\min\{t_{di}, f_i v_i\}$, $u^1_{id} = 2f_i v_i - t^1_{di}$

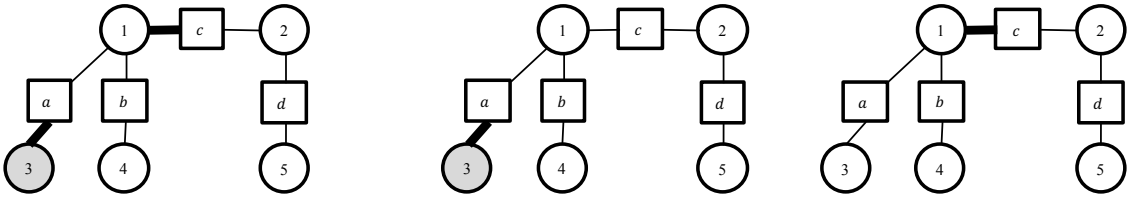


$\boldsymbol{r}$: $v_3 = \alpha_1, t_{a3} = a_2$
$w_3 = \alpha_3, u_{3a} = \alpha_4$.

$\boldsymbol{r^1}$: $v_3^1 = 2\alpha_1, t_{a3}^1 = 2\min\{a_2, f_3\alpha_1\}$.
$u_{3a}^1 = 2f_3\alpha_1 - t_{a3}^1$

$\boldsymbol{r^2}$: $w_3^2 = 2\alpha_3$,
$t_{a3}^2 = 2\alpha_2 - 2\min\{a_2, f_3\alpha_1\}$,
$u_{3a}^2 = 2\alpha_4 - (2f_3\alpha_1 - t_{a3}^1)$.

Figure 7 **When $|S^v \cup S^w| = 1$ and $S^v \cap S^w \neq \emptyset$ (Both v_i and w_i Have Positive Values for Some Node i), $\mathbf{r}$ Is Not Extreme.**



$\boldsymbol{r}$: $w_3 = \alpha_1, t_{a3} = a_2, u_{3a} = \alpha_3$,
$u_{1c} = \alpha_4$. $S_q = \{\{1, c\}\}$.

$\boldsymbol{r^1}$: $w_3^1 = 2\alpha_1, t_{a3}^1 = 2a_2, u_{3a}^1 = 2\alpha_3$,

$\boldsymbol{r^2}$: $u_{1c}^2 = 2a_4$.

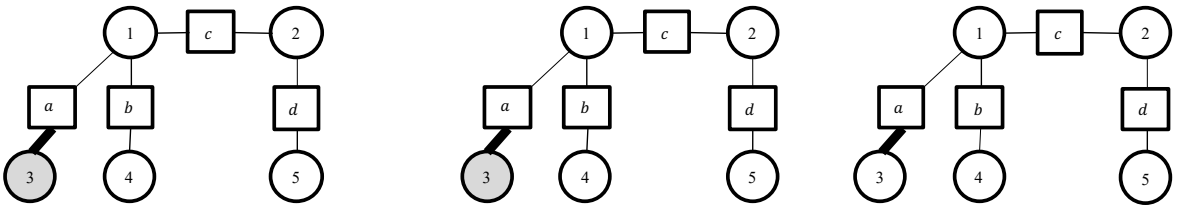
Figure 8 **When $|S^v \cup S^w| = 1$, $|S^v \cap S^w| = 0$ and $S_q \neq \emptyset$ (Some t's and u's Not Adjacent to Node i Have Positive Values), $\mathbf{r}$ Is Not Extreme.**

for all $d \in a(i)$ and 0s for the other components. Thus, if $|S^v \cup S^w| = 1$ and $S^v \cap S^w \neq \emptyset$, $\mathbf{r}$ is not extreme. Figure 7 illustrates this situation.

Suppose that $|S^v \cup S^w| = 1$, $|S^v \cap S^w| = 0$ and $i \in S^v \cup S^w$ (i.e., there is a single node i with either v_i or w_i positive); define $S_q = \{\{j, d\} \in E_t : q_{jd} > 0 \,\&\, j \in V \setminus i\}$. Thus S_q contains edges with positive t and u components that are not adjacent to node i. If $S_q \neq \emptyset$, let $\mathbf{r^1}$ have $v_i^1 = 2v_i$, $w_i^1 = 2w_i$ and $t_{di}^1 = 2t_{di}$, $u_{id}^1 = 2u_{id}$ for all $d \in a(i)$ and 0s in the other components. Thus, if $|S^v \cup S^w| = 1$, $|S^v \cap S^w| = 0$, and $S_q \neq \emptyset$, $\mathbf{r}$ is not extreme. Figure 8 illustrates this situation with $S_q = \{\{1, c\}\}$.

Suppose that $|S^v \cup S^w| = 1$ and $|S^v \cap S^w| = 0$, without loss of generality, let $i \in S^v$ and $S^w = \emptyset$; define $S_1 = \{\{i, d\} \in E_t : t_{di} > 0 \oplus u_{id} > 0\}$ (where only one of the t and u variables associated with an edge $\{i, d\}$ is positive) and $S_2 = \{\{i, d\} \in E_t : t_{di} > 0 \,\&\, u_{id} > 0\}$ (where both t and u variables associated with an edge $\{i, d\}$ are positive). If $S_2 \neq \emptyset$ and let $\{i, d\} \in S_2$, then, we define $\gamma = 2\min\{v_i, \frac{t_{di}}{f_i}, \frac{u_{id}}{f_i} : \{i, d\} \in S_2\}$ and make $\mathbf{r^1}$ have $v_i^1 = \gamma$. For $\{i, d\} \in S_2$, we have $t_{di}^1 = f_i\gamma$. Also, for $\{i, d\} \in S_1$, if $t_{di} > 0$, we have $t_{di}^1 = f_i\gamma$. Otherwise, we have $u_{id}^1 = f_i\gamma$. The remaining components are 0s. Thus, if $|S^v \cup S^w| = 1$, $|S^v \cap S^w| = 0$ and $S_2 \neq \emptyset$, $\mathbf{r}$ is not extreme. Furthermore, $|S_1| = |a(i)|$ because constraint (27) must be satisfied for all d in $a(i)$. Figure 9 illustrates this situation with $S_2 = \{\{3, a\}\}$.

Finally, suppose that $|S^v \cup S^w| = 1$, $|S^v \cap S^w| = 0$, $|S_1| = |a(i)|$, where $i \in S^v \cup S^w$ and $S_2 = \emptyset$. Without loss of generality, let $i \in S^v$ and $S^w = \emptyset$ (the case $i \in S^w$ and $S^v = \emptyset$ can be done

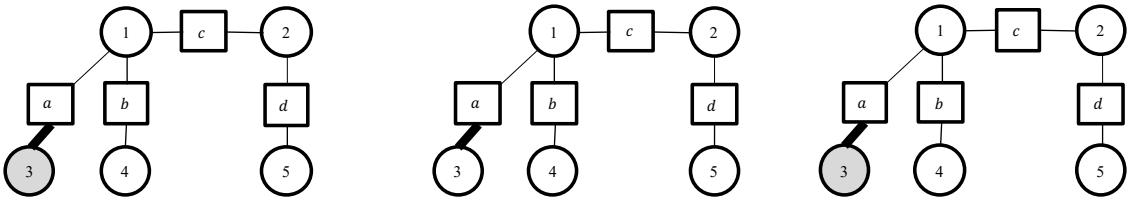


$\boldsymbol{r}: v_3 = \alpha_1, t_{a3} = \alpha_2, u_{3a} = \alpha_3.$
$S_2 = \{\{3, a\}\}.$

$\boldsymbol{r^1}: v_3^1 = 2\min\left\{\alpha_1, \frac{\alpha_2}{f_3}, \frac{\alpha_3}{f_3}\right\} = \gamma,$
$t_{a3}^1 = f_3\gamma.$

$\boldsymbol{r^2}: v_3^2 = 2\alpha_1 - \gamma,$
$t_{a3}^2 = 2\alpha_2 - t_{a3}^1, u_{3a}^2 = 2\alpha_3.$

Figure 9 **When $|S^v \cup S^w| = 1$, $|S^v \cap S^w| = 0$ and $S_2 \neq \emptyset$ (Both t_{di} and u_{id} Have Positive Values for Some $\{i, d\} \in E_t$), $\mathbf{r}$ Is Not Extreme.**



$\boldsymbol{r}: v_3 = \alpha, u_{3a} = f_3\alpha + \beta.$
$S^+ = \{\{3, a\}\}.$

$\boldsymbol{r^1}: u_{3a}^1 = 2\beta.$

$\boldsymbol{r^2}: v_3^2 = 2\alpha, u_{3a}^2 = 2f_3\alpha.$

Figure 10 **When $|S^v \cup S^w| = 1$, $|S^v \cap S^w| = 0$ and $S^+ \neq \emptyset$ (Either t_{di} or u_{id} Is Greater Than α for Some $\{i, d\} \in E_t$), $\mathbf{r}$ Is Not Extreme.**

analogically), let $v_i = \alpha$ and define $S^+ = \{\{i, d\} : q_{id} > f_i\alpha\}$, which has t and u variables whose values are strictly larger than $f_i\alpha$. When $S^+ \neq \emptyset$, without loss of generality, let $\{i, d\} \in S^+$ and $u_{id} > 0$, we can make $\mathbf{r^1}$ have $u_{id}^1 = 2(u_{id} - f_i\alpha)$ and 0s in the other components. Thus, if $|S^v \cup S^w| = 1$, $|S^v \cap S^w| = 0$, and $S^+ \neq \emptyset$, $\mathbf{r}$ is not extreme. Figure 10 illustrates this situation, where $S^+ = \{\{3, a\}\}$ and β is a positive value.

Therefore, if $S^v \cup S^w \neq \emptyset$, we must have $|S^v \cup S^w| = 1$, $|S^v \cap S^w| = 0$, $|S_1| = |a(S^v \cup S^w)|$, and $S_2 = S_q = S^+ = \emptyset$. Thus, $\mathbf{r}$ is either in Case 3 or in Case 4. $\square$

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

$$p_i + (b_i - kf_i)x_i + f_i \sum\nolimits_{j \in S} x_j \geq b_i - kf_i, \quad \forall i \in V,\ k = 0, 1, 2, \ldots, deg(i),\ S \in C_i^{deg(i)-k}. \tag{29}$$

Here, we use $C_i^{deg(i)-k}$ to denote the set of all combinations with $deg(i) - k$ elements from node i's neighbors and S is one combination picked from $C_i^{deg(i)-k}$.

Similarly, Case 4's extreme directions generate the following valid inequality in the original graph G:

$$p_i + (l_ig_i - kl_i)x_i + l_i \sum\nolimits_{j \in S} x_j \geq l_ig_i - kl_i, \quad \forall i \in V,\ k = 0, 1, 2, \ldots, deg(i),\ S \in C_i^{deg(i)-k}. \tag{30}$$

For a given i, if $k \geq g_i$, the constraints generated by Case 3 and Case 4's extreme directions are redundant. Thus, the projection of $\mathrm{P_d}$ onto $(\mathbf{p}, \mathbf{x})$ space is the following:

$$p_i + b_i x_i + f_i \sum_{j \in n(i)} x_j \geq b_i, \qquad \forall i \in V, \tag{31}$$

$$p_i + (b_i - k f_i) x_i + f_i \sum_{j \in S} x_j \geq b_i - k f_i, \qquad \forall i \in V,\ k = 1, 2, \ldots, g_i - 1,\ S \in C_i^{deg(i)-k}, \tag{32}$$

$$p_i + l_i g_i x_i + l_i \sum_{j \in n(i)} x_j \geq l_i g_i, \qquad \forall i \in V, \tag{33}$$

$$p_i + (l_i g_i - k l_i) x_i + l_i \sum_{j \in S} x_j \geq l_i g_i - k l_i, \qquad \forall i \in V,\ k = 1, 2, \ldots, g_i - 1,\ S \in C_i^{deg(i)-k}, \tag{34}$$

$$0 \leq x_i \leq 1,\ p_i \geq 0, \qquad \forall i \in V. \tag{35}$$

Constraints (31) and (33) are obtained from constraints (29) and (30) when $k = 0$, respectively. We list them separately to emphasize that constraint (31) is identical to constraint (6), and constraint (33) is equivalent to constraint (7). Constraints (32) and (34) represent the new inequalities obtained from the projection. For the fractional solution in Figure 3(d), it violates $p_3 + 11x_3 + 14x_2 \geq 11$ (node 3's constraint (32) with $k = 1$ and $S = \{2\}$) and $p_3 + 11x_3 + 11x_2 \geq 11$ (node 3's constraint (34) with $k = 1$ and $S = \{2\}$). After adding these two constraints to LP1L, the resulting LP yields the integral optimal solution shown in Figure 3(b).

Therefore, based on the projection, we obtain a valid formulation (MIP3) with x and p variables only for the PIDS-PP problem. We note that for trees, as the x variables are integral in LP2, constraint (37) can be replaced by its relaxation (35).

$$\text{(MIP3)} \qquad \text{Min} \quad \sum_{i \in V} p_i + \sum_{i \in V} b_i x_i \tag{36}$$

$$\text{Subject to (31), (32), (33) and (34),}$$

$$x_i \in \{0, 1\},\ p_i \geq 0, \qquad \forall i \in V. \tag{37}$$

Although constraints (32) and (34) are exponentially-sized, Proposition 3 shows that they can be separated in polynomial time. Let Δ be the maximum degree number among all nodes (i.e., $\Delta = \max\{deg(i) : i \in V\}$).

PROPOSITION 3. *The valid inequalities (32) and (34) can be separated in $O(|V|\Delta \log \Delta)$ time.*

Proof: Given a fractional solution $(\mathbf{p}^*, \mathbf{x}^*)$, a node i in V and a specific k, where $k = 1, 2, \ldots, g_i - 1$, the corresponding separation procedure of inequality (32) can be formulated as the following optimization problem:

$$\text{Minimize } p_i^* + (b_i - k f_i) x_i^* + f_i \sum_{j \in n(i)} x_j^* z_j \tag{38}$$

$$\text{Subject to} \qquad \sum_{j \in n(i)} z_j = deg(i) - k, \tag{39}$$

$$z_j \in \{0, 1\}, \qquad \forall j \in n(i). \tag{40}$$

Algorithm 1 Separation Algorithm for Inequality Set (32)

Input: A solution x^* and a PIDS-PP problem instance.

for $i \in V$ **do**
  Let $S \leftarrow n(i)$.
  for $k = 1, 2, \ldots, g_i - 1$ **do**
    let $m_k = \arg\max\{x_j^* : j \in S\}$ and $S \leftarrow S \setminus m_k$.
    if $p_i^* + (b_i - kf_i)x_i^* + f_i \sum_{j \in S} x_j^* < b_i - kf_i$ **then**
      Add $p_i + (b_i - kf_i)x_i + f_i \sum_{j \in S} x_j \geq b_i - kf_i$.
    end if
  end for
end for

For each node j in $n(i)$, the binary variable z_j is 1 if node j is included in the combination S. Otherwise, it is 0. If the objective value is smaller than $b_i - kf_i$, we have a violated constraint. Otherwise, we either change the value of k or move to another node i. This optimization problem has one constraint and can be solved by taking the $deg(i) - k$ smallest values of x_j^* among node i's neighbors ($j \in n(i)$). We can use Algorithm 1 to separate the whole inequality set (32). For each node, we sort its neighbors, which takes at most $O(\Delta \log \Delta)$ steps and makes at most Δ comparisons. The process is repeated for $|V|$ nodes. Thus, the overall time complexity is $O(|V|\Delta \log \Delta)$.

One can similarly separate inequalities (34). The difference is that the objective function of the optimization problem should be

$$\text{Minimize } p_i^* + (l_i g_i - k l_i)x_i^* + l_i \sum_{j \in n(i)} x_j^* z_j. \tag{41}$$

If the objective value is smaller than $l_i g_i - k l_i$, we have a violated constraint. □

Before concluding this section, we compare the four aforementioned formulations in terms of their LP relaxations. Let LP1, LP1L, LP2 and LP3 denote the LP relaxations of MIP1, MIP1L, MIP2, and MIP3, respectively. Also, let z_{LP1}, z_{LP1L}, z_{LP2}, and z_{LP3} be the objective values of LP1, LP1L, LP2, and LP3, respectively.

PROPOSITION 4. $z_{LP3} = z_{LP2} \geq z_{LP1L} \geq z_{LP1}$.

Proof: LP3 is obtained by projecting out y variables in LP2. Thus, they are equivalent, $z_{LP3} = z_{LP2}$. Compared to LP1L, LP3 has constraints (32) and (34) in addition to constraints (2) and (4). Therefore, LP3 is at least as strong as LP1L, $z_{LP3} \geq z_{LP1L}$. Finally, LP1L is obtained by adding L-type constraints to LP1. Thus, LP1L is at least as strong as LP1, $z_{LP1L} \geq z_{LP1}$. Notice that the instance in Figure 3 shows that the relationship among these LP relaxations can hold in a strict sense, i.e., $z_{LP3} = z_{LP2} > z_{LP1L} > z_{LP1}$. □

	Source	# of Node	# of Edge
Gnutella	SNAP	10876	39994
Ning	BGU	9727	40570
Hamsterster	KONECT	1788	12476
Escorts	KONECT	10106	39016
Anybeat	N.R.	12645	49132
Advogato	N.R.	5042	39277
Facebook	BGU	39439	50222
Douban	N.R.	154908	327162
Epinions	SNAP	75879	508837
Twitter	N.R.	465017	835423

Table 1 Structure and Sources of the Social Networks.

5. Computational Study

In this section, we discuss our computational experience with these four formulations on (large) real-world social networks. Our computational experiments have several goals. We examine the strength of the formulations empirically (by comparing their LP relaxations), and their effectiveness in solving the PIDS-PP problem. We make these evaluations on seven large social networks first, before embarking on solving instances on three very large social networks with up to approximately 465,000 nodes and 835,000 edges. Finally, we evaluate the benefit of partial payments, compared to a setting where partial payments are not allowed. Our computational experiments are conducted on a machine with the following specifications: Intel Xeon E5-2630V4, 64 GB ram and Ubuntu. For our implementation, we use CPLEX 12.9 with the Python API.

5.1. Descriptions of Social Networks

We obtained ten social networks—Gnutella, Ning, Hamsterster, Escorts, Anybeat, Advogato, Facebook, Douban, Epinions, and Twitter—from four online data repositories. In all of our graphs, nodes represent users and edges are connections between users. We first transform all graphs to 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. Then, we use the biggest connected component in our computational experiments if multiple connected components exist in a graph. Table 1 lists each graph with its source and the number of nodes and edges. The sources SNAP, BGU, KONECT, and N.R., refer to the Stanford Large Network Dataset Collection (SNAP, see Leskovec and Krevl 2014), the BGU Social Networks Security Research Group (BGU, see Lesser et al. 2013), the Koblenz Network Collection (KONECT, see Kunegis 2017), and the Network Repository (N.R., see Rossi and Ahmed 2015), respectively.

Gnutella is a large file sharing peer-to-peer network (the first decentralized peer-to-peer network of its kind). We considered one snapshot of the Gnutella network collected on August 4th, 2002. The Gnutella network on August 4, 2002, has 10,876 nodes and 39,994 edges. *Ning* is an online platform for people and organizations to create custom social networks. Snapshots of the friendship

and group affiliation networks from Ning were harvested during September 2012. It has 9,727 nodes and 40,570 edges. *Hamsterster* contains friendships between users of the website hamsterster.com. It has 1,788 nodes and 12,476 edges. *Escorts* is a bipartite network consisting of 10,106 nodes and 39,106 edges. Nodes in this bipartite network are buyers and their escorts. An edge denotes a transaction between a buyer and an escort. *Anybeat* is an online community, a public gathering place, where one can interact with people from around one's neighborhood or across the world. The data are a friendship network that has 12,645 nodes and 49,132 edges. *Advogato* is based on the friendship network of Advogato.org. It has 5,042 nodes and 39,277 nodes. *Facebook* is a compound Facebook social network of all participants in the LetsDoIt system, a group decision support system prototype for leisure actives. It contains 39,439 nodes and 50,222 edges.

We also have three very large social networks. The first one is based on Douban.com. This website, launched on March 6, 2005, is a Chinese Web 2.0 website providing user interactions for sharing opinions on movies, books, and music. It is one of the largest online communities in China. The graph *Douban* contains the friendship network crawled in December 2010. It has 154,908 nodes and 327,162 edges. *Epinions* is a who-trust-whom online social network of the general consumer review site Epinions.com. Members of the site can decide whether to "trust" one another. Nodes represent users in the Epinions network topology and edges represent trust relationships among Epinions users. The Epinions network has 75,879 nodes and 508,837 edges. *Twitter* is a portion of the social network of Twitter, and contains information about who follows whom with 465,017 nodes and 835,423 edges. These three graphs are not used until our last experiment.

We created a PIDS-PP problem instance by first randomly generating the node type g_i from a discrete uniform distribution between $[1, deg(i)]$ and the influence factor f_i from a discrete uniform distribution between $[1, 50]$. Then, the threshold for a node i was calculated as $b_i = d_i \times (g_i - 1) + s$, where s is generated from a discrete uniform distribution between $[1, f_i]$. In this way, we ensure that if all neighbors of a node are directly targeted, the node becomes influenced. Then, for each social network, ten instances are generated. Thus, there are 100 instances in total given that we have ten social networks. The URL http://dx.doi.org/10.17632/wybgt3wj5x.1 provides these 100 instances.

5.2. Investigating the Strength of the LP Relaxations

We have already shown the relationship between MIP1, MIP1L, MIP2 and MIP3 in terms of their LP relaxations in Proposition 4. Now, we empirically evaluate how much stronger MIP3, MIP2, and MIP1L are, compared to MIP1. In our implementation of MIP2, we remove the constraint $y_{id} + y_{di} = 1$ and use only the variable y_{di} (thus, 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. Next, for our implementation

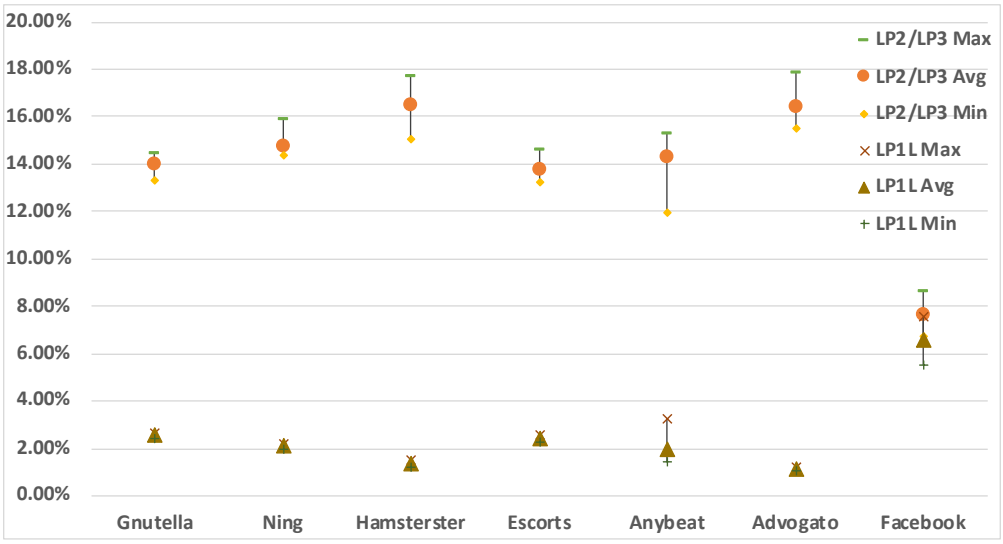


Figure 11 **Relative Improvement of the LP Relaxation of LP1L and LP2/LP3 over That of LP1 on the 70 Instances.**

	LP1			LP1L			LP2			LP3		
	Avg	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg	Min	Max
Gnutella	4.7	4.0	5.7	8.5	7.9	9.4	455.7	374.2	569.4	227.5	165.1	290.0
Ning	2.4	1.8	2.7	4.9	3.9	5.8	391.9	304.9	472.3	195.4	74.8	339.5
Hamsterster	0.1	0.1	0.1	0.2	0.2	0.3	45.3	31.8	66.6	22.8	13.2	39.7
Escorts	3.8	2.7	4.5	7.3	5.8	9.8	496.9	348.4	725.1	207.8	118.7	405.8
Anybeat	3.1	1.9	4.4	5.3	4.0	6.7	647.4	465.8	816.1	286.3	146.1	420.1
Advogato	1.0	0.7	1.4	2.2	1.9	2.7	593.2	409.3	852.9	171.5	77.6	379.7
Facebook	7.1	4.7	10.3	4.4	3.0	6.3	23.9	17.9	38.3	23.5	10.6	34.3

Table 2 **Running Time in Seconds of LP1, LP1L, LP2 and LP3 on the 70 Instances.**

of MIP3, we start with constraints (31) and (33) and use CPLEX's callbacks to add the violated constraints (32) and (34) dynamically, given that constraints (32) and (34) are exponentially-sized.

We use 70 instances, based on Gnutella, Ning, Hamsterster, Escorts, Anybeat, Advogato, and Facebook, to compare the strength of the LP relaxations of the four formulations. Figure 11 plots the average, minimum and maximum value of the relative improvement of the LP Relaxation. Here, we present the relative improvement of LP1L over LP1 and that of LP2/LP3 over LP1. They are calculated as $\frac{z_{LP1L}-z_{LP1}}{z_{LP1}} \times 100$ and $\frac{z_{LP2}-z_{LP1}}{z_{LP1}} \times 100$, respectively. On average, L-type constraints can improve the LP relaxation by 2.6%. However, the average improvement of LP2/LP3 is about 14%. Thus, constraints (32) and (34) contribute to more than a 11% relative improvement on average, in addition to the L-type constraints. The biggest improvement is about 18% for Advogato, and the smallest about 7% for Facebook. Therefore, we can say that LP2 and LP3 are able to significantly improve the quality of the LP relaxation, compared to LP1.

However, the improvement does come at some cost. Table 2 reports the running times of LP1, LP1L, LP2, and LP3 in seconds. For LP3, the running time includes both the separation and

	MIP1 Optimality Gap			MIP1L Optimality Gap		
	Avg	Min	Max	Avg	Min	Max
Gnutella	10.79%	10.25%	11.21%	9.14%	8.70%	9.53%
Ning	10.04%	9.73%	10.43%	9.07%	8.74%	9.49%
Hamsterster	8.82%	7.55%	10.28%	8.21%	6.78%	9.46%
Escorts	10.98%	10.35%	11.49%	9.45%	8.90%	9.96%
Anybeat	8.90%	7.98%	10.07%	8.55%	7.64%	9.59%
Advogato	11.21%	10.55%	12.14%	10.79%	10.16%	11.90%
Facebook	0.67%	0.44%	0.85%	0.46%	0.28%	0.58%
	MIP2 Running Time (s)			MIP3 Running Time (s)		
	Avg	Min	Max	Avg	Min	Max
Gnutella	996.0	276.3	2226.6	380.0	124.3	761.6
Ning	475.6	199.1	982.4	308.4	97.0	1080.8
Hamsterster	90.3	33.4	293.0	71.8	16.4	328.8
Escorts	515.9	202.0	1219.4	250.2	100.9	589.3
Anybeat	767.8	380.4	1311.5	573.3	191.9	1609.5
Advogato	995.8	329.7	2427.9	485.7	92.1	2268.0
Facebook	58.3	49.3	78.9	27.3	11.1	39.5

Table 3 Results of MIP1, MIP1L, MIP2 and MIP3 on the 70 Instances. (The Optimality Gap Is Reported for MIP1 and MIP1L Because None of the Instances Is Solved Optimally. Running Times in Seconds Are Reported for MIP2 and MIP3 Because all 70 Instances Are Solved Optimally.)

solving time. Considering the average running time, LP1L usually needs twice the time of LP1, while LP2 and LP3 need two orders of magnitude more time than LP1. However, LP3 is generally twice as fast as LP2 in our experiment. While LP1, LP1L, and LP2 are compact formulations, LP2 usually needs much more running time than LP1 and LP1L. This is caused by the much bigger size of LP2, compared to LP1 and LP1L. LP1 has $2|V|$ variables and $|V|$ constraints; LP1L has $2|V|$ variables and $2|V|$ constraints; and LP2 has $2|V|+2|E|$ variables and $4|E|+2|V|$ constraints. Overall, LP2 and LP3 provide stronger LP bounds than LP1 and LP1L. In the next experiment, we will evaluate their performance as MIPs.

Before concluding this section, we note that LP1L shows the benefit of the L-type constraints (2.6% over LP1), and LP2 shows the benefit of combining both the L-type constraints and the edge-splitting idea (14% over LP1). For completeness, we ran LP2 without the L-type constraints (constraint (16)) to explicitly quantify the impact of the edge-splitting idea. We found that, on average, the edge-splitting idea can improve LP relaxation by 12% compared to LP1. Generally, but not always, the edge-splitting idea generates a bigger improvement than the L-type constraints. However, combining these two ideas together leads to the largest improvement as demonstrated in LP2.

5.3. Testing the Efficacy of MIP1, MIP1L, MIP2, and MIP3

In this section, we test the performance of MIP1, MIP1L, MIP2, and MIP3. Given that the graphs in our experiments have hundreds of thousands of nodes and edges, a large number of violated constraints (32) and (34) in MIP3 could be found and added in the branch and bound search

	The LP to IP Gap		
	Avg	Min	Max
Gnutella	0.0211%	0.0050%	0.0496%
Ning	0.0198%	0.0130%	0.0351%
Hamsterster	0.1062%	0.0155%	0.2950%
Escorts	0.0183%	0.0000%	0.0726%
Anybeat	0.0187%	0.0082%	0.0284%
Advogato	0.0412%	0.0065%	0.0935%
Facebook	0.0047%	0.0000%	0.0150%

Table 4 The LP to IP Gap of LP2/LP3 on the 70 Instances

process. This can slow down the solution time (because over time many of the cuts that are added may not be of benefit elsewhere in the branch and bound tree) as the size of the model increases. To address this, we use the "Filter" option for the violated constraints that we add through callbacks. In this way, CPLEX manages these dynamically added cuts as the cuts created by CPLEX itself. Then, the cuts that become loose (i.e., that are no longer binding at the optimal solution at a given branch and bound node) will be removed to maintain the model at a reasonable size. Lastly, in order to focus on the effect of various formulations, we turn off CPLEX's own cuts and allow only one thread. Other than that, we keep the default setting for CPLEX. For each instance, the running time is capped at 3600 seconds (1 hour), unless stated otherwise.

We test MIP1, MIP1L, MIP2, and MIP3 on the 70 instances, based on Gnutella, Ning, Hamsterster, Escorts, Anybeat, Advogato, and Facebook. The running time and optimality gap are reported for solved and unsolved instances within the time limit, respectively. Let z_{BFS} and lb be the objective values 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 $\frac{z_{BFS}-lb}{lb} \times 100$. Table 3 shows the results of the four formulations across all seven graphs. We present the optimality gap in the columns "MIP1 Optimality Gap" and "MIP1L Optimality Gap" for MIP1 and MIP1L, and the running times in seconds in columns "MIP2 Running Time" and "MIP3 Running Time" for MIP2 and MIP3, respectively. The average optimality gap is 8.77% and 7.97% for MIP1 and MIP1L, respectively. Not only does MIP1L improve the overall average optimality gap by 0.80%, it also has a smaller optimality gap than MIP1 for 69 out of the 70 instances. Although MIP1L is a stronger formulation than MIP1 (as shown in Section 5.2), it is unable to solve any instance to optimality. However, both MIP2 and MIP3 can solve all 70 instances to optimality, indicating that the L-type constraints together with the edge-splitting idea result in particularly strong formulations for the PIDS-PP problem.

Table 4 presents the gap between the LP relaxations of MIP2 and MIP3 against the optimal integer solutions, which is referred to as the LP-to-IP gap. Let z^* be the objective value of the optimal MIP solution. The LP-to-IP gap is calculated as $(1 - \frac{z_{LP2}}{z^*}) \times 100$. Among all of the 70

	MIP1L+(32)								MIP1L+(34)			
	Solved Instances Running Time (S)				Unsolved Instances Optimality Gap				Unsolved Instances Optimality Gap			
	#	Avg	Min	Max	#	Avg	Min	Max	#	Avg	Min	Max
Gnutella	1	2720.5	2720.5	2720.5	9	0.04%	0.02%	0.08%	10	4.85%	4.54%	5.13%
Ning	8	1108.8	828.3	1716.5	2	0.10%	0.02%	0.17%	10	5.94%	4.99%	8.34%
Hamsterster	10	223.1	20.3	1123.6	0	N.A.	N.A.	N.A.	10	5.88%	4.21%	7.53%
Escorts	8	1940.4	1136.0	3507.4	2	0.06%	0.01%	0.10%	10	6.04%	4.59%	9.18%
Anybeat	8	1849.3	388.8	3221.6	2	0.02%	0.01%	0.02%	10	6.36%	4.43%	8.93%
Advogato	8	1484.0	385.6	3154.1	2	0.78%	0.03%	2.25%	10	7.46%	6.10%	9.74%
Facebook	10	13.8	7.8	20.7	0	N.A.	N.A.	N.A.	10	0.15%	0.04%	0.25%

Table 5 Results of MIP1L+(32) and MIP1L+(34) on the 70 Instances.

instances, the average LP-to-IP gap is 0.0329%, and the maximum LP to IP gap is 0.2950%. Thus, it shows that MIP2 and MIP3 are extremely strong in terms of their LP relaxations. Further, it echoes the important finding that a strong formulation is critical to the computational tractability (Barnhart et al. 1993, Vielma 2015).

Next, we take a closer look at MIP2 and MIP3. On average, MIP3 is about two times faster than MIP2 over the 70 instances solved by both MIP2 and MIP3. Thus, as we pointed out earlier, the much larger size of MIP2 could deteriorate in performance when the size of the instances becomes larger. The difference becomes rather apparent on the very large scale social networks later. It is worth noting that MIP2 consumes much more memory than MIP3. We observe that MIP2 can use more than 30GB memory in the search process, which often is between 10 to 50 times more than the amount required by MIP3. Further, the number of nodes processed by MIP2 is on average about 40% fewer than that of MIP3 in the branch and bound search process. Thus, the larger memory usage in MIP2 is mainly due to the much larger size of MIP2.

Before running tests on the very large scale social networks, given that adding the two sets of constraints (32) and (34) to MIP1 gives the model MIP3 that yielded such strong results, we want to evaluate their benefits separately. To that end, we run two settings: MIP1L+(32), obtained by adding constraint (32) to MIP1L, and MIP1L+(34), obtained by adding constraint (34) to MIP1L. Table 5 describes the results on the 70 test instances. For MIP1L+(32), the results are broken out by solved and unsolved instances for each graph (recall there are 10 instances for each graph). The number of solved instances are provided along with the average, minimum, and maximum running time in seconds. The number of unsolved instances are also provided, along with their average, minimum, and maximum optimality gap. Since MIP1L+(34) does not solve any instances to optimality the results are broken out as unsolved instances and reported in a similar manner to the unsolved instances for MIP1L+(32). While MIP3 solves all of the 70 instances to optimality, MIP1L+(32) solves 53 out of the 70 instances optimally, and MIP1L+(34) doesn't solve any of them to optimality. Thus, although constraint (32) has a stronger performance than constraint (34)

	Optimality Gap			Running Time			
	Avg	Min	Max	Solved #	Avg	Min	Max
Douban	0.00%	0.00%	0.00%	10	4189.2	3427.1	4955.1
Epinions	3.63%	1.70%	5.58%	0	T.L.	T.L.	T.L.
Twitter	0.31%	0.21%	0.48%	0	T.L.	T.L.	T.L.

Table 6 MIP3 on the 30 Very Large Instances. (T.L.: Time Limit 3 Hours)

	Influence Greedy w/o Post Processing			Threshold Greedy w/o Post Processing			Influence Greedy			Threshold Greedy		
	Avg	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg	Min	Max
Gnutella	36%	34%	38%	33%	32%	35%	32%	30%	34%	28%	26%	30%
Ning	33%	31%	37%	31%	28%	35%	30%	28%	33%	26%	24%	29%
Hamsterster	30%	25%	33%	27%	22%	32%	28%	23%	31%	23%	18%	27%
Escorts	37%	33%	38%	33%	30%	35%	33%	30%	34%	28%	25%	30%
Anybeat	34%	26%	38%	30%	25%	35%	29%	23%	33%	26%	21%	31%
Advogato	30%	27%	32%	28%	26%	30%	28%	26%	30%	23%	21%	25%
Facebook	48%	41%	57%	45%	32%	60%	39%	34%	51%	45%	32%	59%
Douban	41%	41%	43%	37%	36%	38%	34%	33%	35%	34%	33%	35%
Epinion	32%	29%	34%	29%	27%	31%	29%	26%	30%	24%	22%	26%
Twitter	43%	41%	45%	40%	39%	43%	35%	33%	38%	39%	37%	42%

Table 7 Relative Gaps of the Cost of Influence Greedy Solution to the Cost of Best Feasible Solution Obtained by MIP3 and that of the Threshold Greedy Solution

computationally, these two constraints complement each other. Combining them together provides the best and strongest outcome!

In the next experiment, we focus on the 30 instances based on three very large social networks, Douban, Epinions and Twitter. Based on our results earlier, only MIP2 and MIP3 are applied to the 30 instances. Also, the time limit was set as 3 hours for each of the 30 instances. When MIP2 is applied to these instances, it **cannot** solve the LP relaxation at the root node for any instance. Needless to say, when the LP relaxation of an MIP formulation cannot be solved, the model is not viable because the search process is predicated on solving the initial LP relaxation. However, MIP3 is able to find and prove good-quality solutions for these instances. The results are shown under the column "Optimality Gap" in Table 6. MIP3 solved 10 Douban instances. For the remaining Epinions and Twitter instances, the optimality gap is less than 2% on average. The running time on the solved instances are also reported in Table 6. The sub-column "Solved #" presents the number of solved instances. Based on the above experiments, we conclude that among the four formulations presented in this paper, MIP3 is the best. MIP3 has $2|V|$ variables, which is the same as MIP1. It has both theoretical desirable properties, and is also capable of finding and proving good-quality solutions for instances with up to 465,000 nodes and 835,000 edges.

5.4. Evaluating the Performance of Greedy Heuristics

Greedy heuristics have served as important benchmarks for influence maximization problems. Recall, Kempe et al. (2003) show that a simple greedy algorithm is an $(1-1/e-\epsilon)$-approximation

ratio for the IMP. Günneç et al. (2020) also describe two greedy heuristics (called influence greedy threshold greedy) for the LCIP, and find that, on average, influence greedy has a 6% gap, and threshold greedy has a 14% gap to the optimal solutions. To this end, we evaluate the cost of the best feasible solution (BFS) obtained by MIP3 against two greedy heuristics for the PIDS-PP problem on the 100 instances that we now describe.

Recall that b_i denotes the threshold for each node i in V. Let t_i be the current partial payment required to influence node i (i.e., initially $t_i = b_i$, and if r neighbors of node i are currently directly targeted then the current partial payment $t_i = \max(0, b_i - rf_i)$), and let $\delta(i)$ denote the neighbors of node i that have yet to be influenced. Influence greedy selects the node (say k) with the smallest influence factor among the nodes that have not yet been influenced. It then weighs the tradeoff between paying b_k to directly target it, or t_k as a partial payment. Compared to paying t_k to node k, paying b_k reduces the partial payment t_j of each node j in $\delta(k)$ by its influence factor f_j, if $t_j > f_j$, or reduces its current threshold t_j to 0, if $t_j \leq f_j$. Therefore, if $b_k - t_k > \sum_{j\in\delta(k)} \min\{f_j, t_j\}$, we pay the current partial payment t_k, and mark node k as being influenced. If $b_k - t_k \leq \sum_{j\in\delta(k)} \min\{f_j, t_j\}$, we pay the threshold b_k, mark node k as being influenced and directly targeted, and propagate influence to nodes in $\delta(k)$. Then, we select the next node. Threshold greedy is similar to influence greedy except that it selects the node (say k) with the smallest value of t_k among the nodes that have not yet been influenced. For both greedy heuristics, we apply a post processing step to lower the partial payments made to nodes that are not directly targeted (this could happen for example to a node k that is given a partial payment t_k in either influence or threshold greedy, if one of node k's neighbor is selected for direct targeting later in the greedy algorithm thereby reducing the partial payment necessary) if possible.

We use z_{IG}, z_{TH} and z_{MIP3} to denote the cost of influence greedy, that of threshold greedy and that of the best feasible solution obtained by MIP3, respectively. Table 7 shows the relative gaps, which are calculated as $(\frac{z_{IG}}{z_{MIP3}} - 1) \times 100$ and $(\frac{z_{TG}}{z_{MIP3}} - 1) \times 100$ for influence greedy and threshold greedy, respectively. The first two columns show the gaps without the post processing step, and the last two columns show the gaps with the post processing step. On average, without the post processing step, the cost of influence greedy and that of threshold greedy are 36% and 33% larger than that of the solution obtained by MIP3, respectively. Invoking the post processing step reduces the gaps to 32% and 30%, respectively. Thus, this post processing step is able to improve the solution quality by reducing the gap between 3% to 4% on average. Overall, threshold greedy obtains better solutions for 75 out of the 100 instances, while influence greedy is better for 25 of them. Although threshold greedy performs better than influence greedy, neither of them is able to provide outcomes that are as good as the solutions obtained by MIP3. This finding emphasizes the benefit of well-designed optimization methods for the PIDS-PP problem.

	Avg	Min	Max
Gnutella	25.02%	23.34%	26.46%
Ning	22.81%	21.22%	25.34%
Hamsterster	19.54%	13.50%	22.10%
Escorts	24.72%	23.61%	26.49%
Anybeat	21.62%	18.03%	24.81%
Advogato	19.78%	16.06%	21.63%
Facebook	14.43%	11.58%	16.09%
Douban	20.80%	20.09%	21.39%
Epinion	8.41%	6.45%	10.93%
Twitter	17.84%	17.30%	18.41%

Table 8 Relative Extra-Cost without Partial Payment.

5.5. Evaluating the Impacts of Partial Payment

In the next experiment, we evaluate the impacts of partial payment. To do so, we evaluate the cost and the solution structure of a PIDS-PP problem instance against a setting, where no partial payment is allowed (i.e., either a node is paid b_i or 0). In other words, we compare the cost and the solution structure of a PIDS-PP problem instance against the cost and the solution structure of its PIDS problem counterpart. Recall that in the PIDS problem, nodes must either be directly targeted (and paid b_i) or have at least g_i neighbors that are directly targeted.

We convert our instances into their corresponding PIDS problem instances by calculating g_i (this is the number of directly targeted neighbors that a node needs in order to be influenced) for all nodes i in V. We obtain a lower bound on the optimal solution of the corresponding PIDS problem by solving the BIP4 formulation described by Raghavan and Zhang (2017). The time limit for each PIDS problem instance is set to be the same as its PIDS-PP problem counterpart. Let lb_{pids}, and $z_{pids\text{-}pp}$ be the lower bound of the PIDS problem instance and the objective value of the best feasible solution of the PIDS-PP problem instance, respectively. The relative extra-cost is at least $\frac{lb_{pids} - z_{pids\text{-}pp}}{z_{pids\text{-}pp}} \times 100$. Table 8 shows that the relative extra-cost is around 20% on average across all instances. The least amount is 6.45% for Epinions and the largest amount is 26.46% for Gnutella.

We also look at the impact of partial payment on the solution structure. Taking the best feasible solutions of the PIDS problem and the PIDS-PP problem instances, Table 9 compares the average fraction of nodes being directly targeted (DT), the average fraction of nodes receiving partial payment (PP), and the average fraction of nodes receiving no payment (Rest) in the solutions. Thus, compared to the PIDS problem solutions, fewer nodes (approximately 3% fewer nodes in our experiments) are directly targeted generally in the PIDS-PP problem. On the other hand, 26% fewer nodes receive no payments in the PIDS-PP compared to the PIDS problem. Overall, partial payments are able to significantly reduce the overall targeting cost, by slightly reducing the number of nodes being directly targeted and instead providing partial payments to a fraction (between 10% and 20% in our experiments) of the nodes in the graph.

	PIDS		PIDS-PP		
	DT	Rest	DT	PP	Rest
Gnutella	51.73%	48.27%	48.13%	18.74%	33.13%
Ning	47.48%	52.52%	46.29%	16.28%	37.43%
Hamsterster	51.48%	48.52%	49.47%	20.77%	29.77%
Escorts	50.35%	49.65%	47.64%	18.41%	33.95%
Anybeat	40.24%	59.76%	44.81%	13.32%	41.86%
Advogato	50.10%	49.90%	49.04%	19.53%	31.43%
Facebook	35.27%	64.73%	31.67%	10.47%	57.86%
Douban	40.29%	59.71%	39.88%	11.66%	48.45%
Epinions	43.96%	56.04%	43.37%	15.66%	40.97%
Twitter	41.00%	59.00%	35.99%	9.75%	54.26%

Table 9 Comparison of the Solution Structure between the PIDS Problem and the PIDS-PP Problem. (DT: The Average Fraction of Nodes Being Directly Targeted, PP: The Average Fraction of Nodes Receiving Partial Payment, Rest: the Average Fraction of Nodes Receiving No Payment)

6. Conclusions

In this paper, we study the PIDS-PP problem, an NP-hard problem that arises in the context of direct targeting on social networks, where direct influence plays the dominant role in influence propagation. In contrast to much of the previous influence maximization literature, partial payments and latency constraints (of 1) are considered simultaneously in the PIDS-PP problem. Partial payments are more closely aligned with marketing practice, where it is common to provide partial payment (e.g., coupons that reduce the price of a product) instead of receiving a product gratis. As our computational experiments show, they can also save significantly in targeting costs. Latency constraints arise in many different settings, including technology diffusion, or settings where rapid influence maximization is desired.

This paper focuses on developing strong MIP formulations for the PIDS-PP problem. We show that the PIDS-PP problem on trees is polynomially solvable via a linear-time DP algorithm. We discuss four different formulations for the PIDS-PP problem: MIP1, MIP1L, MIP2, and MIP3. MIP2 and MIP3 are equivalent as linear programs, and stronger than both MIP1 and MIP1L. MIP2 is a compact extended formulation that applies several strengthening ideas (including edge-splitting) to MIP1L. MIP3 is obtained by projecting MIP2 onto the space of the payment variables. MIP3 and MIP1L lie in the same space, but MIP3 has an exponential set of additional constraints (obtained by projecting from MIP2). We also show that the extended formulation MIP2 and its projection MIP3 are the strongest possible ones for trees (i.e., they provide integral solutions when the underlying graph is a tree). We conduct an extensive computational experiment on real-world social networks to show the efficacy of MIP3. Our results show that MIP3 finds high-quality solutions for very large graphs with approximately 465,000 nodes and 835,000 edges.

There is one natural direction for further research. In the PIDS-PP problem, influence is only allowed to propagate to the neighbors of a directly targeted node (i.e., one step), while in the LCIP,

influence is allowed to propagate indefinitely (or $|V|-1$ steps). These represent two extremes: one where only direct influence plays a role, and the other, where indefinitely long chains of indirect influence are permitted. In between lie the 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 partial payments and latency constraints where we are allowed a prespecified number of steps/time periods for the influence to propagate through the network. This is a problem of significant practical relevance, but remains a challenging next step (since the formulation for the PIDS-PP problem and the LCIP are completely different and cannot be applied directly to this variant). Our work on the PIDS-PP problem provides a strong first step along this research pathway.

References

Adcock, A. B., Sullivan, B. D., and Mahoney, M. W. (2013). Tree-like structure in large social and information networks. In *2013 IEEE 13th International Conference on Data Mining*, pages 1–10. IEEE.

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

Balasundaram, B., Butenko, S., and Hicks, I. V. (2011). Clique relaxations in social network analysis: The maximum k-plex problem. *Operations Research*, 59(1):133–142.

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

Barnhart, C., Johnson, E. L., Nemhauser, G. L., Sigismondi, G., and Vance, P. (1993). Formulating a mixed integer programming problem to improve solvability. *Operations Research*, 41(6):1013–1019.

Borgs, C., Brautbar, M., Chayes, J., and Lucier, B. (2014). Maximizing social influence in nearly optimal time. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 946–957. SIAM.

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

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

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

Chen, W., Castillo, C., and Lakshmanan, L. V. (2013). *Information and influence propagation in social networks*. Morgan & Claypool Publishers.

Chlebík, M. and Chlebíková, J. (2008). Approximation hardness of dominating set problems in bounded degree graphs. *Information and Computation*, 206(11):1264–1275.

Coleman, J., Katz, E., and Menzel, H. (1966). *Medical innovation: A diffusion study*. Bobbs-Merrill, Indianapolis.

Conforti, M., Cornuéjols, G., and Zambelli, G. (2014). *Integer programming*. Springer, New York.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to algorithms*. MIT Press Cambridge.

Dhawan, A. and Rink, M. (2015). Positive influence dominating set generation in social networks. In *2015 International Conference on Computing and Network Communications (CoCoNet)*, pages 112–117. IEEE.

Dinh, T. N., Shen, Y., Nguyen, D. T., and Thai, M. T. (2014). On the approximability of positive influence dominating set in social networks. *Journal of Combinatorial Optimization*, 27(3):487–503.

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

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

Goemans, M. X. (1994). The steiner tree polytope and related polyhedra. *Mathematical programming*, 63(1-3):157–182.

Goyal, A., Lu, W., and Lakshmanan, L. V. (2011). Celf++ optimizing the greedy algorithm for influence maximization in social networks. In *Proceedings of the 20th international conference companion on World wide web*, pages 47–48.

Granovetter, M. (1978). Threshold models of collective behavior. *American Journal of Sociology*, 83(6):1420–1443.

Greaves, C. J., Reddy, P., and Sheppard, K. (2010). Supporting behaviour change for diabetes prevention. In *Diabetes Prevention in Practice*, pages 19–29. TUMAINI Institute for Prevention Management.

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

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

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

Haynes, T., Hedetniemi, S., and Slater, P. (1998a). *Domination in graphs: Advanced topics*. CRC Press.

Haynes, T., Hedetniemi, S., and Slater, P. (1998b). *Fundamentals of domination in graphs*. CRC Press.

Jackson, M. O., Malladi, S., and McAdams, D. (2020). Learning through the grapevine: The impact of noise and the breadth and depth of social networks. Working Paper.

Kempe, D., Kleinberg, J., and Tardos, É. (2003). Maximizing the spread of influence through a social network. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 137–146.

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

Kunegis, J. (2017). KONECT network dataset. http://konect.uni-koblenz.de/networks/konect.

Kwak, H., Lee, C., Park, H., and Moon, S. (2010). What is twitter, a social network or a news media? In *Proceedings of the 19th international conference on World wide web*, pages 591–600.

Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., VanBriesen, J., and Glance, N. (2007). Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 420–429. ACM.

Leskovec, J. and Krevl, A. (2014). SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data.

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

Li, X., Smith, J. D., Dinh, T. N., and Thai, M. T. (2019). Tiptop:(almost) exact solutions for influence maximization in billion-scale networks. *IEEE/ACM Transactions on Networking*, 27(2):649–661.

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

Liben-Nowell, D. and Kleinberg, J. (2008). Tracing information flow on a global scale using internet chain-letter data. *Proceedings of the national academy of sciences*, 105(12):4633–4638.

Lin, G., Guan, J., and Feng, H. (2018). An ILP based memetic algorithm for finding minimum positive influence dominating sets in social networks. *Physica A: Statistical Mechanics and its Applications*.

Magnanti, T. L. and Wolsey, L. A. (1995). Optimal trees. *Handbooks in operations research and management science*, 7:503–615.

Nannicini, G., Sartor, G., Traversi, E., and Wolfler-Calvo, R. (2020). An exact algorithm for robust influence maximization. *Mathematical Programming*. Forthcoming.

Nemhauser, G. L. and Wolsey, L. A. (1988). *Integer and combinatorial optimization*. Wiley New York.

Nguyen, H. T., Thai, M. T., and Dinh, T. N. (2016). Stop-and-stare: Optimal sampling algorithms for viral marketing in billion-scale networks. In *Proceedings of the 2016 International Conference on Management of Data*, pages 695–710.

Pajouh, F. M., Balasundaram, B., and Hicks, I. V. (2016). On the 2-club polytope of graphs. *Operations Research*, 64(6):1466–1481.

Raghavan, S. and Zhang, R. (2017). Rapid influence maximization on social networks: The positive influence dominating set problem. Working paper, University of Maryland.

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

Rossi, R. A. and Ahmed, N. K. (2015). The network data repository with interactive graph analytics and visualization. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*.

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

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

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

Tang, Y., Shi, Y., and Xiao, X. (2015). Influence maximization in near-linear time: A martingale approach. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 1539–1554.

Valente, T. W. (2012). Network interventions. *Science*, 337(6090):49–53.

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

Vielma, J. P. (2015). Mixed integer linear programming formulation techniques. *SIAM Review*, 57(1):3–57.

Walteros, J. L. and Buchanan, A. (2020). Why is maximum clique often easy in practice? *Operations Research*. Forthcoming.

Wang, F., Camacho, E., and Xu, K. (2009). Positive influence dominating set in online social networks. In Du, D.-Z., Hu, X., and Pardalos, P. M., editors, *Combinatorial Optimization and Applications*, pages 313–321, Berlin, Heidelberg. Springer.

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

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

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

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

Electronic Companion

EC.1. Remarks on The PIDS-PP Problem on Trees

The study of the PIDS-PP problem on trees is important for several reasons. First, the information propagation paths on social networks often resemble trees, as shown by Liben-Nowell and Kleinberg (2008) on Internet chain-letter data and by Kwak et al. (2010) on Twitter data. Second, from the perspective of the decomposition, Adcock et al. (2013) empirically demonstrate real social and information networks possess large-scale tree-like structure. Third, tree graphs are commonly used in stylized models on social networks to generate insights and obtain a better understanding of the underlying problem (see Jackson et al. 2020). Fourth, deriving tight and strong models on special cases can be considered as a starting point for obtaining stronger models on the general case for many combinatorial optimization problems (see Goemans 1994, Magnanti and Wolsey 1995). Our interest in trees is due to the fact that trees are the structurally simplest non-trivial graphs. Thus, a high-level idea is to derive useful results and insights from special cases which are polynomial solvable, and then apply them to the general case which is NP-hard. The model we derive for the PIDS-PP problem on trees plays a crucial role in developing our model on arbitrary graphs. The linear-time algorithm plays an important role in proving the tightness of MIP2 on trees. The rest of this electronic companion is organized as follows. Section EC.2 presents a linear-time dynamic algorithm for the PIDS-PP problem on tree. Section EC.3 proves that MIP2 is indeed the strongest possible formulation for the PIDS-PP problem on trees.

EC.2. Algorithm for The PIDS-PP Problem on Trees

In this section, we present a dynamic programming (DP) algorithm to solve the PIDS-PP 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-PP 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 for direct targeting (for brevity, we will simply say that the node is "not selected"), whereas in the second case, the parent is selected for direct targeting (for brevity, we will simply say that the node is "selected"). Recall that when a node is selected for direct targeting it receives full payment. 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

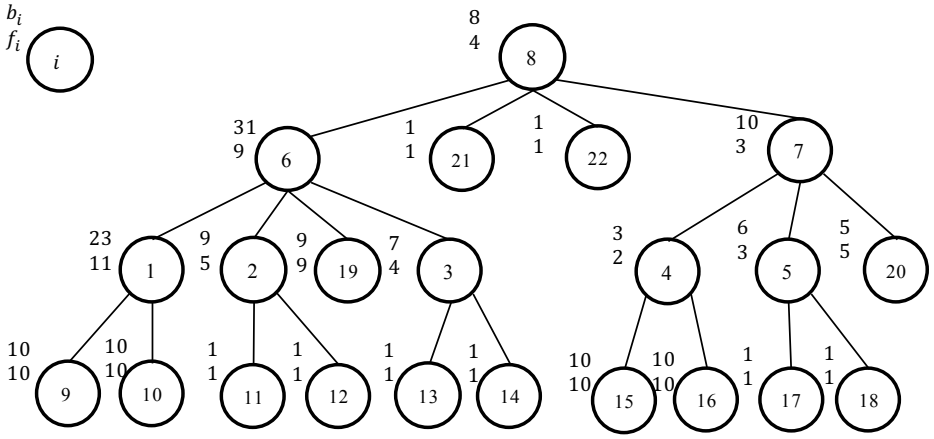


Figure EC.1 A PIDS-PP Problem Instance.

to combine the solution candidates from the star subproblems and identify a final solution (set of nodes selected for direct targeting, and the payment amounts to all nodes on 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 by 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-PP problem on a star. To better illustrate the algorithm, we consider the instance shown in Figure EC.1. Let L denote the set of all leaf nodes in the original tree. Let c denote the central node of a star, all of the other nodes in the star are children of the central node and are denoted by $L(c)$ (notice that $L(c)$ need not be a subset of L), and refer to this star as star 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 X_c^{NPS} represent the set of nodes selected in the solution to star c, let P_c^{NPS} be the payment made to the central node in the solution, and let C_c^{NPS} be the total cost of the solution for star c. Analogously, we consider the case, where the

Algorithm 2 Algorithm for the PIDS-PP problem 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: For all $i \in L$ (the set of leaf nodes in G), set $b_i^1 = b_i$, and $b_i^2 = 0$.
4: **for** each star subproblem **do**
5: StarHandling
6: **end for**
7: SolutionBacktrack

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, P_c^{PS} being the payment made to the central node in the solution, and C_c^{PS} denoting the total cost of the solution for star c.

In Figure EC.1, node 8 is selected as the root. Following the bottom-up ordering of the tree we would consider stars $1, 2, 3, 4$ and 5 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". Notice that because influence can only propagate to a neighbor of a selected node (and no further), 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 a bottom star. Either the central node of the bottom star is selected or all children in the bottom star are selected with an additional (potentially 0) partial payment to the central node. 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 $(\sum_{j\in L(c)} b_j + \max\{0, b_c - |L(c)|f_c\})$. If b_c is greater, then all of the children in the bottom star are selected and $X_c^{NPS} = L(c)$, $P_c^{NPS} = \max\{0, b_c - |L(c)|f_c\}$, and $C_c^{NPS} = (\sum_{j\in L(c)} b_j + \max\{0, b_c - |L(c)|f_c\})$. Otherwise, the central node c of the bottom star is selected and $X_c^{NPS} = \{c\}$, $P_c^{NPS} = b_c$, and $C_c^{NPS} = b_c$. At the same time, for the case where the parent of the central node c of the bottom star is selected, we compare the cost of b_c against the cost of $(\sum_{j\in L(c)} b_j + \max\{0, b_c - (|L(c)+1)|f_c\})$. If b_c is greater, then all of the children are selected and $X_c^{PS} = L(c)$, $P_c^{PS} = \max\{0, b_c - (|L(c)+1)|f_c\}$, and $C_c^{PS} = (\sum_{j\in L(c)} b_j + \max\{0, b_c - (|L(c)+1)|f_c\})$. Otherwise, the central node c is selected and $X_c^{PS} = \{c\}$, $P_c^{PS} = b_c$, 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 selecting node 1 with a payment of 23 against the cost of selecting all of the leaf nodes of star 1 (i.e., nodes 9 and 10) at a cost of 20 units and a partial payment amounting to 1 unit to node 1 (for a total cost of 21). Thus, the solution is $X_1^{NPS} = \{9, 10\}$, $P_1^{NPS} = 1$, and $C_1^{NPS} = 21$. Now consider the case, where node 1's parent (node 6) is selected. In this case, we compare the cost of selecting the central node 1 (with a payment of 23) against selecting all of the leaf nodes of the star (i.e., nodes 9 and 10) with no additional partial payment to the central node (for a total cost of 20). Thus, the solution is $X_1^{PS} = \{9, 10\}$, $P_1^{PS} = 0$, and $C_1^{PS} = 20$. Similarly, for the bottom stars 2, 3, 4, and 5, we find: In star 2, $X_2^{NPS} = X_2^{PS} = \{11, 12\}$, $P_2^{NPS} = P_2^{PS} = 0$, and $C_2^{NPS} = C_2^{PS} = 2$. In star 3, $X_3^{NPS} = X_3^{PS} = \{13, 14\}$, $P_3^{NPS} = P_3^{PS} = 0$, and $C_3^{NPS} = C_3^{PS} = 2$. In star 4, $X_4^{NPS} = X_4^{PS} = \{4\}$, $P_4^{NPS} = P_4^{PS} = 3$, and $C_4^{NPS} = C_4^{PS} = 3$. In star 5, $X_5^{NPS} = X_5^{PS} = \{17, 18\}$, $P_5^{NPS} = P_5^{PS} = 0$, and $C_5^{NPS} = C_5^{PS} = 2$.

Next, once a star's solution candidates are determined, the star is contracted into a single child node for its parent's star subproblem. It may appear that we have considered all possible solution candidates X_c^{NPS} and X_c^{PS} for a given star c in the optimal solution. However, that is not necessarily the case. Consider star 1. Here $X_1^{PS} = X_1^{NPS} = \{9, 10\}$. In both cases, the leaf nodes 9 and 10 are

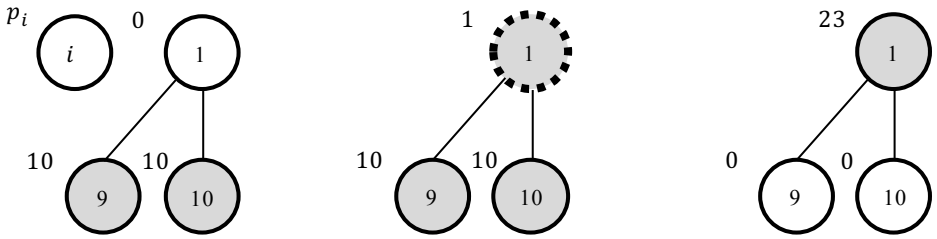


Figure EC.2 $(a) X_1^{PS} = \{9, 10\}, P_1^{PS} = 0, C_1^{PS} = 20$, $(b) X_1^{NPS} = \{9, 10\}, P_1^{NPS} = 1, C_1^{NPS} = 21$, (c) **selected node 1 and** $b_1 = 23$. **Possible Solutions for Star 1 in the Optimal Solution to the Problem.**

selected. Although star 1 does not need its central node 1 to be selected in either case, star 6 may need node 1 to be selected in order to activate its central node 6 because influence only propagates to the neighbors of the selected nodes. 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. Figure EC.2 illustrates the situations for star 1. The nodes receiving payment are shaded. Among them, the ones with solid border are selected (i.e., directly targeted and receiving full payment) and the others with a dotted border receive partial payment. Figure EC.2(a) displays the solution X_1^{PS}, Figure EC.2(b) displays the solution X_1^{NPS}, and Figure EC.2(c) displays the solution, where the central node 1 is selected. Observe that the costs of the three solutions satisfy $C_c^{PS} \leq C_c^{NPS} \leq b_c$. Therefore, we must incur a cost of at least C_c^{PS} for star c in the optimal solution. 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 are used to solve the next star subproblem.

Unlike bottom stars, we have (e.g., star 6) stars containing both contracted stars as leaf nodes (nodes 1, 2, and 3), as well as the leaf nodes in L (node 19). Therefore, we compute $b_i^1 = b_i$, and $b_i^2 = 0$ for leaf nodes $i \in L$. Consider node 19 to explain this calculation. If its parent is not selected, node 19 must be selected with a cost of 9. If its parent is selected, node 19 gets influenced and its cost is 0. Thus, $b_{19}^1 = 9$, and $b_{19}^2 = 0$. Figure EC.3 displays stars 6 and 7 after contracting stars 1, 2, 3, 4 and 5, respectively. At this point $TC = 29$ and the calculated values of (b_i^1, b_i^2) are shown for all leaf nodes.

We are now ready to discuss how to solve the PIDS-PP 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 influence the entire star (if the central node c is selected then all children $i \in L(c)$ follow the solution X_i^{PS} whose cost is already included in TC) or we select a subset of nodes in $L(c)$ as cheaply as possible that influences the entire star (the nodes i in $L(c)$ that are not selected would then follow the solution X_i^{NPS}).

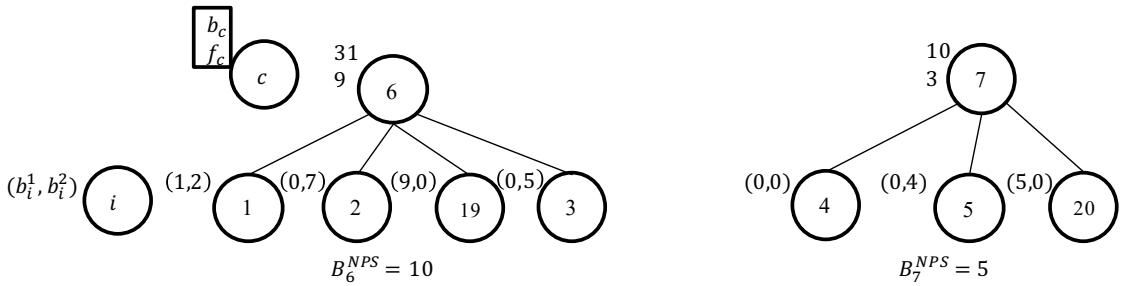


Figure EC.3 **(a) Star 6 after Contracting Stars 1, 2, and 3.** **(b) Star 7 after Contracting Stars 4 and 5.**

We need to compute the cost of the alternative, where a minimum cost subset of nodes in $L(c)$ are selected to influence the entire star. If the central node c of the star is not selected, we must at least incur the cost $B_c^{NPS} = \sum_{i \in L(c)} b_i^1$, given that all of the children $i \in L(c)$ must at least incur the cost of C_i^{NPS} when their parent is not selected. The influence factor f_c of the central node c is used as a filter. Any child $i \in L(c)$ with $b_i^2 \geq f_c$ will never be selected. When $b_i^2 \geq f_c$, paying b_i^2 units to a child node i sends f_c influence to node c, thus, the decrease in the threshold of the central node is less than what we spend. We could be better off by paying the central node directly and using the X_i^{NPS} solution for such children. Therefore, we collect the nodes $i \in L(c)$ with $b_i^2 < f_c$ in set S_c and sort them in ascending order of their b_i^2 values. The set $S_c^0 \subseteq S_c$ denotes the set of nodes $i \in S_c$ with $b_i^2 = 0$ (i.e., these nodes are essentially free to select). The cost of the solution depends on the size of the set S_c. Let $g_c = \lceil \frac{b_c}{f_c} \rceil$ and $l_c = b_c - (g_c - 1) f_c$.

Case 1 ($|S_c| < g_c$): All nodes in the set S_c are selected, and the payment to the central node c is determined as $l_c + f_c(g_c - 1 - |S_c|)$ for a total cost of $B_c^{NPS} + l_c + f_c(g_c - 1 - |S_c|) + \sum_{i \in S_c} b_i^2$.

Case 2 ($|S_c| \geq g_c$ and $|S_c^0| \geq g_c$): All nodes in the set S_c^0 are selected, and no additional payment to the central node is necessary, resulting in a total cost of B_c^{NPS}.

Case 3 ($|S_c| \geq g_c$ and $|S_c^0| < g_c$): We select the first g_c nodes in S_c in ascending order of their b_i^2 value (we denote this set as S_{g_c}) if the threshold for the g_c-th node is less than or equal to l_c for a total cost of $B_c^{NPS} + \sum_{i \in S_{g_c}} b_i^2$. Otherwise, we select the first $(g_c - 1)$ nodes in S_c in ascending order of their b_i^2 values (we denote this set as $S_{g_c - 1}$) and pay l_c to node c, resulting in a total cost of $B_c^{NPS} + l_c + \sum_{i \in S_{g_c - 1}} b_i^2$.

Comparing b_c, the cost of selecting the central node c against the cost of the solution just obtained from one of the three above cases (and selecting the one with a lower cost) provides us with the solution to the PIDS-PP problem 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 influence the entire star, or we select a subset of nodes in $L(c)$ as cheaply as possible that influences the entire star. The cost of the alternative, where a minimum cost subset of the nodes in $L(c)$

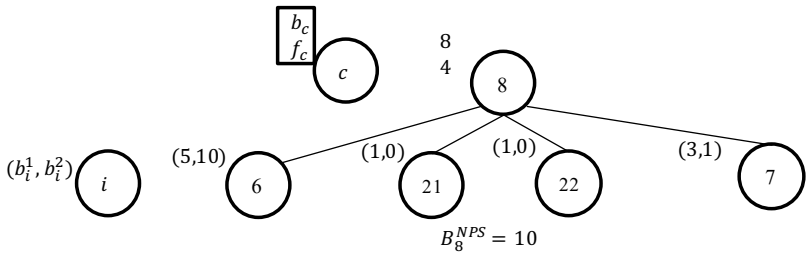


Figure EC.4 The Last Star after Contracting Stars 6 and 7.

are selected to influence the entire star, is calculated identically as the above three cases with the change 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, which finds the optimal solution for a given star. When the procedure is applied to star 6, we start with $S_6 = \{19, 1, 3, 2\}$, $S_6^0 = \{19\}$ and $B_6^{NPS} = 10$. We get $X_6^{NPS} = \{19, 1, 3\}$, $P_6^{NPS} = 4$, and $C_6^{NPS} = B_6^{NPS} + P_6^{NPS} + \sum_{i \in X_6^{NPS}} b_i^2 = 21$; and $X_6^{PS} = \{19, 1\}$, $P_6^{PS} = 4$, and $C_6^{PS} = B_6^{NPS} + P_6^{PS} + \sum_{i \in X_6^{PS}} b_i^2 = 16$. Contracting star 6 gives $b_6^1 = 5$ and $b_6^2 = 10$. Similarly for star 7, we start with $S_7 = \{4, 20\}$, $S_7^0 = \{4, 20\}$ and $B_7^{NPS} = 5$ (notice that node 5 is not in S_7 because $b_5^2 = 4 > f_7 = 3$). We get $X_7^{NPS} = \{4, 20\}$, $P_7^{NPS} = 4$ and $C_7^{NPS} = 9$; and $X_7^{PS} = \{4, 20\}$, $P_7^{PS} = 1$, and $C_7^{PS} = 6$. Contracting star 7 gives $b_7^1 = 3$ and $b_7^2 = 1$. Now, TC is 51. After contracting stars 6 and 7, we only have one star left, as shown in Figure EC.4. Here, $S_8 = \{21, 22, 7\}$, $S_8^0 = \{21, 22\}$ and $B_8^{NFS} = 10$. We find that it is cheapest to select central node 8 (because its cost of 8 is less than $B_8^{NFS} = 10$) than any of its children. Hence, $X_8^{NPS} = \{8\}$, $P_8^{NPS} = 8$ and $C_8^{NPS} = 8$. Thus, TC is 59.

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 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), $\mathbf{x}^*$ be a binary vector indicating the nodes selected with a value of 1s, and $\mathbf{p}^*$ be the vector of payment. Note that this $\mathbf{p}^*$ payment vector is slightly different from the p variables in the MIP formulations, as it

Algorithm 3 StarHandling

Input: star c.

$(X_c^{NPS}, P_c^{NPS}, C_c^{NPS}) \leftarrow$ SolveStar(star c, NoParent-Selected).
if star c is the last star **then**
  $TC = TC + C_c^{NPS}$
else
  $(X_c^{PS}, P_c^{PS}, C_c^{PS}) \leftarrow$ 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)
  $B_c^{NPS} = \sum_{i \in L(c)} b_i^1$, $g_c = \left\lceil \frac{b_c}{f_c} \right\rceil$, $l_c = b_c - (g_c - 1) f_c$, $S_c = \{i \in L(c) : b_i^2 < f_c\}$ and $S_c^0 = \{i \in L(c) : b_i^2 = 0\}$.
  if Flag == Parent-Selected **then**
    $g_c = g_c - 1$.
  end if
  if $|S_c| < g_c$ **then**
    $C = \min\left\{b_c,\ B_c^{NPS} + \sum_{i \in S_c} b_i^2 + l_c + f_c(g_c - 1 - |S_c|)\right\}$
  else
    if $|S_c^0| \geq g_c$ **then**
      $C = \min\{b_c,\ B_c^{NPS}\}$
    else
      Let S_{g_c} and $S_{g_c - 1}$ be the sets of the cheapest g_c and $(g_c - 1)$ nodes in S_c by b^2, respectively.
      $C = \min\left\{b_c,\ B_c^{NPS} + \sum_{i \in S_{g_c}} b_i^2,\ B_c^{NPS} + \sum_{i \in S_{g_c - 1}} b_i^2 + l_c\right\}$
    end if
  end if
  $X \leftarrow c$ and $P = b_c$ **if** C is b_c.
  $X \leftarrow S_c$ and $P = l_c + f_c(g_c - 1 - |S_c|)$ **if** C is $B_c^{NPS} + \sum_{i \in S_c} b_i^2 + l_c + f_c(g_c - 1 - |S_c|)$.
  $X \leftarrow S_c^0$ and $P = 0$ **if** C is B_c^{NPS}.
  $X \leftarrow S_{g_c}$ and $P = 0$ **if** C is $B_c^{NPS} + \sum_{i \in S_{g_c}} b_i^2$.
  $X \leftarrow S_{g_c - 1}$ and $P = l_c$ **if** C is $B_c^{NPS} + \sum_{i \in S_{g_c - 1}} b_i^2 + l_c$.
  return X, P, C.
end function

includes both full and partial payments. Figure EC.5 shows the final solution. The nodes receiving payment are shaded. Among them, the ones with a solid border are selected (i.e., directly targeted and receiving full payment), and the others with a dotted border receive partial payment.

Proof of Proposition 2: The 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. For each star c, we need to find g_c cheapest children that takes $O(deg(c))$ time. Finding the g_cth order statistics can be done in $O(deg(c))$ time by the Quickselect method in Chapter 9 of Cormen et al. (2009); thus, it takes $O(deg(c))$ time to go through the list to collect the g_c cheapest normal children. For the whole tree, this is bounded by $O(|V|)$ time. In the backtracking procedure, we pick the final solution for each node, which takes $O(|V|)$ time over the tree. Therefore, the running time for the dynamic algorithm is linear on the number of nodes. □

Algorithm 4 SolutionBacktrack

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

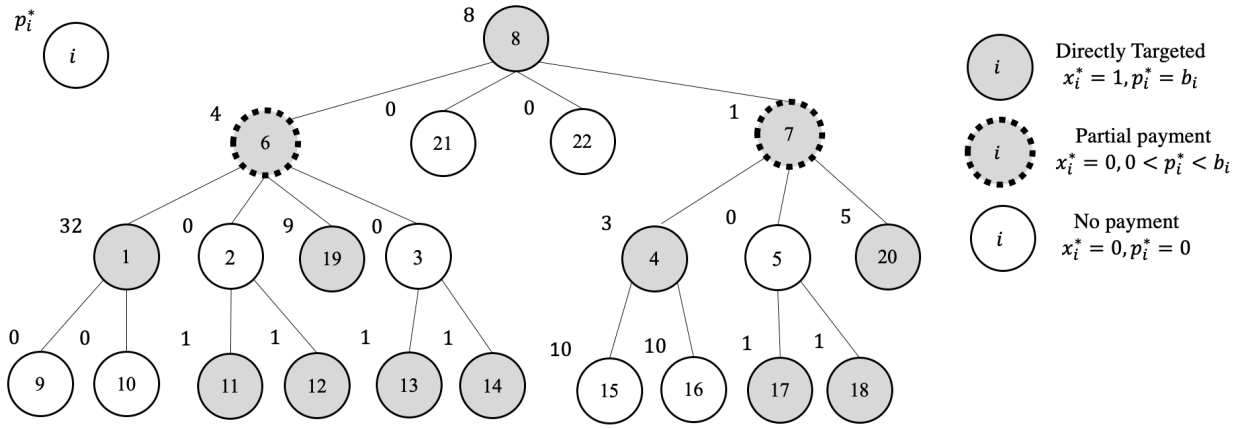


Figure EC.5 **The Solution Obtained by Our DP Algorithm. Nodes 1, 4, 7, 8 11, 12, 13, 14 17, 18, 19, and 20 are Selected (i.e., Directly Targeted and Receive Full Payment). Nodes 6 and 7 Receive Partial Payment.**

EC.3. Proof of Theorem 1: MIP2 is Tight for The PIDS-PP Problem on Trees

The linear relaxation of MIP2 is the following LP problem:

$$\text{(LP2)} \quad \text{Min} \quad \sum_{i \in V} p_i + \sum_{i \in V} b_i x_i \tag{EC.1}$$

$$\text{Subject to } (s_{id}) \quad -y_{id} - y_{di} = -1, \qquad \forall \{i,d\} \in E_t, \tag{EC.2}$$

$$(t_{ij}) \quad x_i - y_{dj} \geq 0, \qquad \forall i \in V, j \in n(i), \tag{EC.3}$$

$$(u_{id}) \quad y_{id} - x_i \geq 0, \qquad \forall i \in V, d \in a(i), \tag{EC.4}$$

$$(v_i) \quad p_i + b_i x_i + \sum_{d \in a(i)} f_i y_{di} \geq b_i, \quad \forall i \in V, \tag{EC.5}$$

$$(w_i) \quad p_i + l_i g_i x_i + \sum_{d \in a(i)} l_i y_{di} \geq l_i g_i, \ \forall i \in V, \tag{EC.6}$$

$$p_i,\ x_i \geq 0, \qquad \forall i \in V, \tag{EC.7}$$

$$y_{id},\ y_{di} \geq 0, \qquad \forall \{i,d\} \in E_t. \tag{EC.8}$$

We refer to this LP problem as LP2. We have s_{id}, t_{ij}, u_{id}, v_i, and w_i as dual variables for the constraints (EC.2), (EC.3), (EC.4), (EC.5), and (EC.6) respectively. The dual to LP2, which is referred to as DLP2 is as follows:

$$\text{(DLP2) Max} \quad \textstyle\sum_{i\in V} b_i v_i + \sum_{i\in V} l_i g_i w_i - \sum_{\{i,d\}\in E_t} s_{id} \tag{EC.9}$$

$$\text{Subject to } (y_{id}) \qquad -s_{id} + u_{id} \leq 0, \qquad \forall i \in V, d \in a(i), \tag{EC.10}$$

$$(y_{di}) \qquad -s_{id} - t_{ji} + f_i v_i + l_i w_i \leq 0, \qquad \forall d \in D, i \in a(d), \tag{EC.11}$$

$$(x_i) \quad \textstyle\sum_{j\in n(i)} t_{ij} - \sum_{d\in a(i)} u_{id} + b_i v_i + l_i g_i w_i \leq b_i, \ \forall i \in V, \tag{EC.12}$$

$$(p_i) \qquad v_i + w_i \leq 1, \qquad \forall i \in V, \tag{EC.13}$$

$$s_{id} \text{ free}, \qquad \forall \{i,d\} \in E_t, \tag{EC.14}$$

$$t_{ij} \geq 0, \qquad \forall i \in V, j \in n(i), \tag{EC.15}$$

$$u_{id} \geq 0, \qquad \forall i \in V, d \in a(i), \tag{EC.16}$$

$$v_i,\ w_i \geq 0, \qquad \forall i \in V. \tag{EC.17}$$

First, based on this pair of primal and dual problems, we have the complementary slackness (CS) conditions:

$$(-s_{id} + u_{id}) y_{id} = 0, \qquad \forall i \in V, d \in a(i), \tag{EC.18}$$

$$(x_i - y_{dj}) t_{ij} = 0, \qquad \forall i \in V, j \in n(i), \tag{EC.19}$$

$$(y_{id} - x_i) u_{id} = 0, \qquad \forall i \in V, d \in a(i), \tag{EC.20}$$

$$(p_i + b_i x_i + \textstyle\sum_{d\in a(i)} f_i y_{di} - b_i) v_i = 0, \qquad \forall i \in V, \tag{EC.21}$$

$$(p_i + l_i g_i x_i + \textstyle\sum_{d\in a(i)} l_i y_{di} - l_i g_i) w_i = 0, \qquad \forall i \in V, \tag{EC.22}$$

$$(-s_{id} - t_{ji} + f_i v_i + l_i w_i) y_{di} = 0, \qquad \forall d \in D, i \in a(d), \tag{EC.23}$$

$$(\textstyle\sum_{j\in n(i)} t_{ij} - \sum_{d\in a(i)} u_{id} + b_i v_i + l_i g_i w_i - b_i) x_i = 0, \ \forall i \in V, \tag{EC.24}$$

$$(v_i + w_i - 1) p_i = 0, \qquad \forall i \in V. \tag{EC.25}$$

The solution vector $\mathbf{x}^*$, $\mathbf{p}^*$ obtained in the dynamic programming algorithm (Algorithm 2 in Appendix EC.2) can be transferred into a feasible solution for LP2. For any node i that is selected node set $x_i = 1$ and $p_i = 0$. Then, set $y_{di} = 0$ and $y_{id} = 1$ for all d in $a(i)$. For all other nodes i that are not selected, set $x_i = 0$ and $p_i = p_i^*$. Next, let $S_i^d = \{d \in a(i) \cap a(j) : x_j = 1 \ \forall j \in n(i)\}$. For all $d \in S_i^d$, set $y_{di} = 1$ and $y_{id} = 0$. For all $d \in a(i) \setminus S_i^d$, set $y_{di} = 0$ and $y_{id} = 1$.

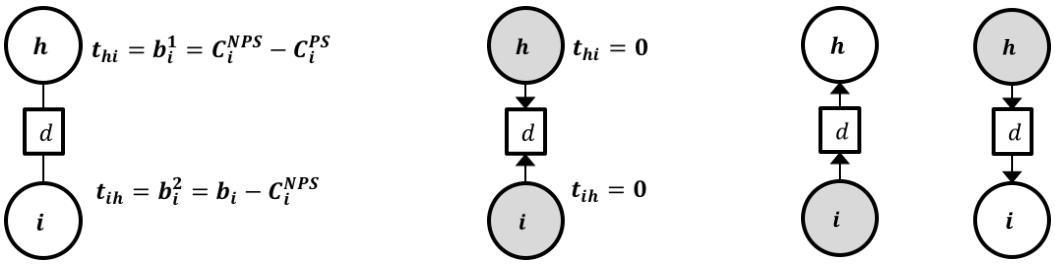


Figure EC.6 (a) **t Variables.** (b) **Condition (EC.19) when** $x_i \neq y_{dh}$. (c) **Case 1: A Leaf Node** i $p_i = 0$ **and** $x_i = y_{id}$.

With this primal feasible solution in hand, a dual feasible solution will be constructed in the following proof, and we show that the CS conditions are satisfied by this pair of primal and dual solutions. Throughout the proof, we always have $u_{id} = s_{id}$. Thus, the CS condition (EC.18) is satisfied. We can focus on the remaining CS conditions.

In DLP2, only t variables interact between two nodes in V. If their values are fixed, we can isolate each node in V and assign values to s, u, v and w variables. Following the bottom-up order as defined in Algorithm 2, we start with the bottom level of the original 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, S_i, X_i^{NPS}, P_i^{NPS}, X_i^{PS}, and P_i^{PS} are obtained in Algorithm 2. We set $t_{ih} = b_i^2$ and $t_{hi} = b_i^1$, as shown in Figure EC.6(a). For condition (EC.19), it requires $t_{ih} = 0$ when $x_i = 1$ and $y_{dh} = 0$. This means that the corresponding $x_h = 1$. Given that node h and node i are both selected, it implies that $C_i^{PS} = b_i$. Thus, $b_i^1 = b_i^2 = 0$ because $C_i^{PS} \leq C_i^{NPS} \leq b_i$. Thus, $t_{ih} = 0$ and $t_{hi} = 0$, as shown in Figure EC.6(b). For other situations, we have $x_i = y_{dh}$. Thus, condition (EC.19) is satisfied.

Now, three cases are considered to assign associated dual variables for a node i in V. All s, 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. When node i is selected, $p_i = 0$, $x_i = 1$, and $y_{id} = 1$. Otherwise, $p_i = 0$, $x_i = 0$ and $y_{id} = 0$. This means that $p_i = 0$, and $x_i = y_{id}$, as shown in Figure EC.6(c). Also, $t_{ih} = 0$ and $t_{hi} = b_i$ because $b_i^1 = b_i$ and $b_i^2 = 0$. Set $v_i = 1$. All primal and dual constraints are binding for conditions (EC.20), (EC.21), (EC.22) (EC.23), (EC.24), and (EC.25) given that $b_i = f_i$. 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 $x_i = 0$. Let $S_i^j = \{j \in n(i) : x_j = 1\}$, a subset of node i's neighbors in the original tree G receiving full payment. We have two situations. First, when $p_i = 0$, it means that node i has g_i or more incoming arcs, as shown in Figure EC.7(a). Let $\delta_i = \max\{t_{ji} : j \in S_i^j\}$. It is the biggest t_{ji} value among the nodes in S_i^j. Now, if S_i^j only contains the child nodes of node i, then, each of them must have $b_j^2 = t_{ji} \leq l_i$, given that $p_i = 0$. If S_i^j contains node i's parent (node h), it implies that $|X_i^{PS}| \geq |g_i| - 1$, and X_i^{PS} only contains the

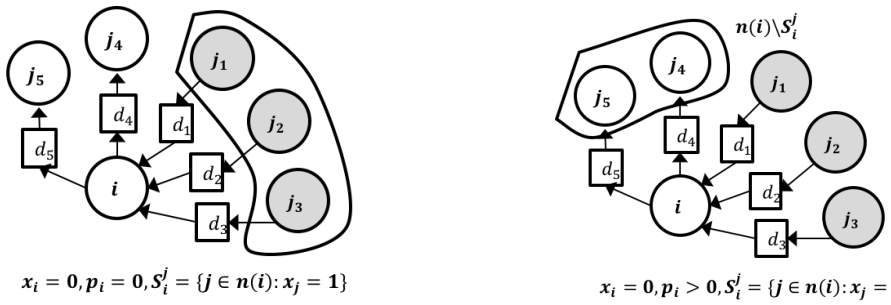


Figure EC.7 **Case 2:** (a) $|S_i^j| \geq g_i$ **and** $\delta_i = \max\{t_{ji} : j \in S_i^j\}$. (b) $|S_i^j| < g_i$ **and** $\gamma_i = \min\{f_i, \min\{t_{ji} : j \in n(i) \setminus S_i^j\}\}$.

child nodes of node i. We can infer the solution for node i when its parent is not selected. When $|X_i^{PS}| > |g_i| - 1$, no extra payment is needed ($P_i^{NPS} = 0$). When $|X_i^{PS}| = |g_i| - 1$, either node i is paid l_i ($P_i^{NPS} = l_i$), or a child node j with $b_j^2 \leq l_i$ is selected in X_i^{NPS}, compared to X_i^{PS}. Thus, $b_i^1 = C_i^{NPS} - C_i^{PS} \leq l_i$. As a result, we have $\delta_i \leq l_i$. Set $w_i = \frac{\delta_i}{l_i}$. Then, set $s_{id} = u_{id} = \delta_i - t_{ji}$ for all j in S_i^j and d in S_i^d. Condition (EC.20) is satisfied because $y_{id} = x_i = 0$ for all d in S_i^d, and $u_{id} = 0$ for all d in $a(i) \setminus S_i^d$. Condition (EC.21) is satisfied because $v_i = 0$. Condition (EC.22) is satisfied because constraint (EC.6) is binding when there are exactly g_i incoming arcs, and $w_i = \delta_i = 0$ when there are more than g_i incoming arcs. Condition (EC.23) is satisfied because constraint (EC.11) is binding for all d in S_i^d, and $y_{di} = 0$ for all d in $a(i) \setminus S_i^d$. Constraint (EC.12) is satisfied because its left-hand side is

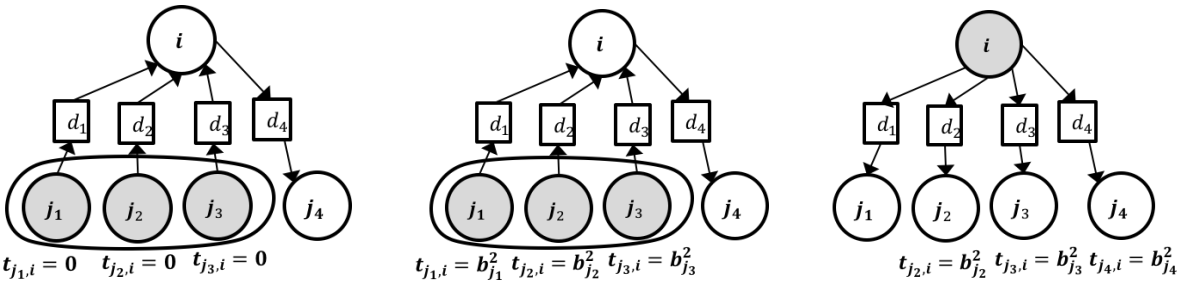
$$
\begin{aligned}
& B_i^{NPS} + b_i^2 - \sum_{j \in S_i^j} (\delta_i - t_{ji}) + g_i \delta_i && \text{(Note: } \sum_{j \in n(i)} t_{ij} = B_i^{NPS} + b_i^2, u_{id} = \delta_i - t_{ji}, w_i = \frac{\delta_i}{l_i}\text{)} \\
= & \begin{cases} B_i^{NPS} + \sum_{j \in X_i^{NPS}} b_j^2 + b_i^2 - (|S_i^j| - g_i)\delta_i & \text{if } h \notin S_i^j \\ B_i^{NPS} + \sum_{j \in X_i^{PS}} b_j^2 + b_i^1 + b_i^2 - (|S_i^j| - g_i)\delta_i & \text{if } h \in S_i^j \end{cases} \\
= & \ C_i^{NPS} + b_i^2 - (|S_i^j| - g_i)\delta_i \leq b_i. && \text{(Note: } |S_i^j| \geq g_i, \delta_i \geq 0\text{)}
\end{aligned}
$$

Constraint (EC.13) is respected because $w_i = \frac{\delta_i}{l_i} \leq 1$. Thus, conditions (EC.24) and (EC.25) are satisfied because $x_i = p_i = 0$.

Second, when $p_i > 0$, it means that node i has less than g_i incoming arcs, as shown in Figure EC.7(b). Let $\gamma_i = \min\{f_i, \min\{t_{ji} : j \in n(i) \setminus S_i^j\}\}$. We set v_i and w_i by solving the following equations:

$$
\begin{aligned}
f_i v_i + l_i w_i &= \gamma_i, \\
v_i + w_i &= 1.
\end{aligned}
$$

Then, set $s_{id} = u_{id} = \gamma_i - t_{ji}$ for all j in S_i^j and all d in S_i^d. Condition (EC.20) is satisfied because $y_{id} = x_i = 0$ for all d in S_i^d, and $u_{id} = 0$ for all d in $a(i) \setminus S_i^d$. When $p_i \geq l_i + f_i$, $|S_i^j| < g_i - 1$, constraint (EC.5) is binding and constraint (EC.6) is satisfied. It has $\gamma_i = f_i$, $v_i = 1$ and $w_i = 0$. When



(a) $X_i^{PS}\neq\{i\}$ and $|X_i^{PS}|\geq g_i$ (b) $X_i^{PS}\neq\{i\}$ and $|X_i^{PS}|\leq g_i-1$ (c) $X_i^{PS}=\{i\}$

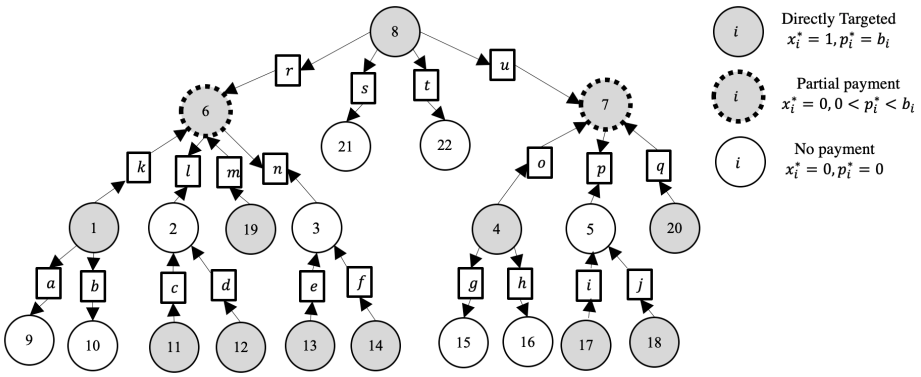
Figure EC.8 Case 3.

$p_i=l_i$, $|S_i^j|=g_i-1$ and both constraints (EC.5) and (EC.6) are binding. Thus, conditions (EC.21) and (EC.22) are satisfied. Condition (EC.23) is satisfied because constraint (EC.11) is binding for all d in S_i^d and $y_{di}=0$ for all d in $a(i)\setminus S_i^d$. Analogous to the previous situation, constraint (EC.12) is binding. The left-hand side of constraint (EC.12) is:

$$
\begin{aligned}
& B_i^{NPS}+b_i^2-\sum_{j\in S_i^j}(\gamma_i-t_{ji})+b_iv_i+l_ig_iw_i \\
= & \begin{cases}
B_i^{NPS}+\sum_{j\in X_i^{NPS}}b_j^2+b_i-|X_i^{NPS}|f_i+b_i^2 & \text{if } h\notin S_i^j \text{ and } p_i\geq l_i+f_i \\
B_i^{NPS}+\sum_{j\in X_i^{PS}}b_j^2+b_i-(|X_i^{PS}|+1)f_i+b_i^1+b_i^2 & \text{if } h\in S_i^j \text{ and } p_i\geq l_i+f_i \\
B_i^{NPS}+\sum_{j\in X_i^{NPS}}b_j^2+b_iv_i+l_ig_iw_i-(g_i-1)\gamma_i+b_i^2 & \text{if } h\notin S_i^j \text{ and } p_i=l_i \\
B_i^{NPS}+\sum_{j\in X_i^{PS}}b_j^2+b_iv_i+l_ig_iw_i-(g_i-1)\gamma_i+b_i^1+b_i^2 & \text{if } h\in S_i^j \text{ and } p_i=l_i
\end{cases} \\
= & C_i^{NPS}+b_i^2=b_i.
\end{aligned}
$$

Thus, Condition (EC.24) is satisfied because $x_i=0$. Condition (EC.25) is satisfied because constraint (EC.13) is binding.

Case 3: Suppose that node i is not a leaf node in G and $x_i=1$. This means that $p_i=0$, $y_{id}=1$ and $y_{di}=0$ for all d in $a(i)$. Then, CS conditions (EC.20), (EC.21), and (EC.22) are satisfied because those corresponding primal constraints are satisfied and binding. Constraints (EC.11) and (EC.13) are satisfied. Thus, conditionss (EC.23) and (EC.25) are satisfied because $y_{di}=0$ and $p_i=0$. Because $x_i=1$, constraint (EC.12) must be binding. Let LHS denote the value of the left-hand side of constraint (EC.12). 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 first be constructed to ensure that $LHS\geq b_i$ and then it can be adjusted to have $LHS=b_i$. Recall that X_i^{PS} is a set of nodes selected in the Parent-Selected case for node i in the DP. Based on X_i^{PS}, we consider three situations. First, suppose that $X_i^{PS}\neq\{i\}$ and $|X_i^{PS}|\geq g_i$, as shown in Figure EC.8(a). This means that the size of node i's child nodes with a zero b^2 value is at least g_i ($|S_i^0|\geq g_i$) and $X_i^{PS}=X_i^{NPS}$. Thus, $LHS=\sum_{j\in n(i)}t_{ij}=B_i^{NPS}+b_i^2=C_i^{PS}+b_i^1+b_i^2=b_i$ because $b_i^1=0$ and $B_i^{NPS}=C_i^{PS}$.



$t_{1,9} = 10$	$t_{7,20} = 5$	$v_{18} = 1$
$t_{1,10} = 10$	$t_{6,8} = 10$	$v_{19} = 1$
$t_{2,11} = 1$	$t_{8,6} = 5$	$v_{20} = 1$
$t_{2,12} = 1$	$t_{8,21} = 1$	$v_{21} = 1$
$t_{3,13} = 1$	$t_{8,22} = 1$	$v_{22} = 1$
$t_{3,14} = 1$	$t_{7,8} = 1$	$w_1 = 1$
$t_{4,15} = 10$	$t_{8,7} = 3$	$s_{1,a} = 1$
$t_{4,16} = 10$	$v_9 = 1$	$s_{1,b} = 1$
$t_{5,17} = 1$	$v_{10} = 1$	$s_{4,g} = 17$
$t_{5,18} = 1$	$v_{11} = 1$	$s_{8,r} = 2$
$t_{1,6} = 2$	$v_{12} = 1$	$v_6 = 0.2$
$t_{6,1} = 1$	$v_{13} = 1$	$w_6 = 0.8$
$t_{2,6} = 7$	$v_{14} = 1$	$s_{6,k} = 3$
$t_{6,19} = 9$	$v_{15} = 1$	$s_{6,m} = 5$
$t_{3,6} = 5$	$v_{16} = 1$	$v_7 = 1$
$t_{5,7} = 4$	$v_{17} = 1$	$s_{7,o} = 3$
		$s_{7,q} = 3$

Figure EC.9 **(a) Primal Solution for the PIDS-PP Problem Instance from Figure EC.1. (b) Non-Zero Dual Variable Values Except for u.**

Second, suppose that $X_i^{PS} \neq \{i\}$ and $|X_i^{PS}| \leq g_i - 1$, as shown in Figure EC.8(b). This implies that $X_i^{PS} \neq X_i^{NPS}$. Set $w_i = 1$ if $|X_i^{PS}| = g_i - 1$. Set $v_i = 1$ if $|X_i^{PS}| < g_i - 1$. Then, set $u_{id} = s_{id} = f_i v_i + l_i w_i - t_{ji} = f_i v_i + l_i w_i - b_j^2$ for all j in X_i^{PS} and d in $a(i) \cap a(j)$. Then,

$$\begin{aligned} LHS &= \sum_{j \in n(i)} t_{ij} - \sum_{d \in a(i)} u_{id} + b_i v_i + l_i g_i w_i \\ &= \sum_{j \in n(i)} t_{ij} + \sum_{j \in X_i^{PS}} b_j^2 - |X_i^{PS}|(f_i v_i + l_i w_i) + b_i v_i + l_i g_i w_i \\ &= \begin{cases} B_i^{NPS} + b_i^2 + \sum_{j \in X_i^{PS}} b_j^2 + l_i & \text{if } w_i = 1 \text{ and } |X_i^{PS}| = g_i - 1 \\ B_i^{NPS} + b_i^2 + \sum_{j \in X_i^{PS}} b_j^2 + (g_i - 2 - |X_i^{PS}|) f_i + l_i + f_i & \text{if } v_i = 1 \text{ and } |X_i^{PS}| < g_i - 1 \end{cases} \\ &\geq C_i^{PS} + b_i^1 + b_i^2 = b_i \end{aligned}$$

The last inequality holds because $C_i^{PS} = B_i^{NPS} + \sum_{j \in X_i^{PS}} b_j^2$ and $b_i^1 \leq l_i$ when $|X_i^{PS}| = g_i - 1$, and $b_i = (g_i - 1) f_i + l_i$, $C_i^{PS} = B_i^{NPS} + \sum_{j \in X_i^{PS}} b_j^2 + (g_i - 2 - |X_i^{PS}|) f_i + l_i$ and $b_i^1 \leq f_i$ when $|X_i^{PS}| < g_i - 1$.

Lastly, suppose that node i is selected in the Parent-Selected solution ($X_i^{PS} = \{i\}$), as shown in Figure EC.8(c). This implies that $X_i^{NPS} = X_i^{PS}$. Let $DS_i = S_i$ initially. If $|S_i| \geq g_i$, let $DS_i = S_{g_i - 1}$. Set $v_i = 1$ and $u_{id} = s_{id} = f_i - t_{ji} = f_i - b_j^2$ for all j in DS_i and d in $a(i) \cap a(j)$. Thus, $LHS = B_i^{NPS} + \sum_{j \in DS_i} b_j^2 - |DS_i| f_i + b_i = B_i^{NPS} + \sum_{j \in DS_i} b_j^2 + (g_i - 1 - |DS_i|) f_i + l_i > b_i$ as $b_i = (g_i - 1) f_i + l_i$, $b_i^2 = 0$ and node i is selected by the DP algorithm. At this point, we ensure that $LHS \geq b_i$. Then, if $LHS > b_i$, set $u_{i\tilde{d}} = s_{i\tilde{d}} = u_{i\tilde{d}} + LHS - b_i$ for a $\tilde{d}$ in $a(i)$. Thus, condition (EC.24) is satisfied because constraint (EC.12) is binding by construction. □

Figure EC.9(a) is the transformed graph, and its solution is based on the instance in Figure EC.1. We use it to illustrate the procedure for setting the dual variables. The non-zeros dual variables are displayed in Figure EC.9(b), except for u variables, which have the same value as s variables.

For clarity, we use a comma to separate the two subscripts in u_{ij}, t_{ij}, and s_{ij} (i.e., we write them as $u_{i,j}$, $t_{i,j}$, and $s_{i,j}$).

First, we assign values for t variables. Their values are in the first two columns of the table in Figure EC.9(b) and are displayed in Figure EC.9(a) as well. For Case 1, we have nodes 9, 10, 11, 12, 13, 15, 16, 17, 18, 19, 20, 21, and 22. Set their associated v as 1. For Case 2, we have nodes 2, 3, 5, 6, and 7. For nodes 2, 3, and 5, $\delta_2 = \delta_3 = \delta_5 = 0$. Thus, no variables need to be changed. For node 6, it has $S_6^j = \{1, 8, 19\}$, $p_6 = 4$, and $\gamma_6 = \min\{9, \min\{5, 7\}\} = 5$. Thus, $v_6 = 0.2$, $w_6 = 0.8$, $s_{6,k} = u_{6,k} = 5 - 2 = 3$, and $s_{6,m} = u_{6,m} = 5$. For node 7, it has $S_7^j = \{4, 8, 20\}$, $p_7 = 1$, and $\gamma_7 = \min\{3, \min\{4\}\} = 3$. Thus, $v_7 = 1$, $s_{7,o} = u_{7,o} = 3$, and $s_{7,q} = u_{7,q} = 3$. For Case 3, we have nodes 1, 4, and 8. For node 1, set $w_1 = 1$, $s_{1,a} = u_{1,a} = 1$, and $s_{1,b} = u_{1,b} = 1$ because $X_1^{PS} = \{9, 10\}$. For nodes 4 and 8, set $s_{4,g} = u_{4,g} = 20 - 3 = 17$ and $s_{8,r} = u_{8,r} = 10 - 8 = 2$. The objective value is 59, which is exactly the same as that of the solution obtained by Algorithm 2 in Appendix EC.2.