Received: 12 October 2019 | Accepted: 2 July 2020 | Published on: 10 September 2020

DOI: 10.1002/net.21972

SPECIAL ISSUE ARTICLE

Weighted target set selection on trees and cycles

S. Raghavan[1] | Rui Zhang[2]

[1]Robert H. Smith School of Business and Institute for Systems Research, University of Maryland, College Park, Maryland, USA
[2]Leeds School of Business, University of Colorado, Boulder, Colorado, USA

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

Abstract

There is significant interest in understanding the dynamics of influence diffusion on a social network. The weighted target set selection (WTSS) problem is a fundamental viral marketing problem arising on social networks. In this problem, the goal is to select a set of influential nodes to target (e.g., for promoting a new product) that can influence the rest of the network. The WTSS problem is APX-hard. With the goal of generating insights to solve the WTSS problem on arbitrary graphs, we study in this paper the WTSS problem on trees and cycles. For trees, we propose a linear-time dynamic programming algorithm and present a tight and compact extended formulation. Furthermore, we project the extended formulation onto the space of the natural node variables yielding the polytope of the WTSS problem on trees. This projection leads to an exponentially sized set of valid inequalities whose polynomial-time separation is also discussed. Next, we focus on cycles: we describe a linear-time algorithm and present the complete description of the polytope for the WTSS problem on cycles. Finally, we describe how these formulations can be applied to arbitrary graphs.

KEYWORDS

cycles, influence maximization, integer programming, polytope, social networks, trees

1 | INTRODUCTION

The target set selection (TSS) problem is a fundamental problem in social network analytics. Chen [8] posed the problem to answer the following viral marketing question: suppose we want to take advantage of peer influence on a social network to promote a new product. The diffusion process can be initialized by targeting influential people. Other people start to adopt this product due to the influence they receive from these early adopters. Ideally, as the number of people who have adopted the product grows, a cascade will result, encompassing the entire network. Yet, how do we select these influential people who are targeted initially? Mathematically, in the TSS problem, we are given a connected undirected graph $G = (V, E)$, where for each node $i \in V$, there is a threshold, denoted by g_i, that is between 1 and the degree of node i, denoted by $deg(i)$. All nodes are initially inactive. We select a subset of nodes, the target set, which become active immediately. Afterwards, in each step, we update the state of the nodes with the following rule: an inactive node i becomes active if at least g_i of its neighbors are active in the previous step. The goal is to find the minimum cardinality target set, while ensuring that all nodes are active by the end of this activation process.

In this paper, we consider the *weighted TSS* (WTSS) problem. In the WTSS problem, for each node $i \in V$, there is a positive weight, denoted by b_i, which models the situation in which different nodes might require various levels of effort to become initial adopters. The goal is to minimize the total sum of b_i for the selected nodes in the target set. Chen [8] showed that the TSS problem is APX-hard, which means that it is NP-hard to approximate within a polylogarithmic factor. He also provided a polynomial-time algorithm for the TSS problem on trees. However, the algorithm relies on an observation that is not true in the weighted case, and thus does not apply to the weighted version of the problem. Chiang et al. [10] provided a linear-time algorithm for the TSS problem on block cactus graphs. They also showed that the TSS problem is polynomially solvable on chordal graphs when $g_i \leq 2$ and on Hamming graphs when $g_i = 2$. Ben-Zwi et al. [6] showed that for a graph with a treewidth bounded by ω, the TSS problem can be solved in V^{ω} time. As with Chen [8], these algorithms only apply for the unweighted case. Ackerman et al. [1] provided some combinatorial bounds for the TSS problem under majority ($g_i \geq \frac{\deg(i)}{2}$) and strict majority ($g_i > \frac{\deg(i)}{2}$) influences.

Two papers have discussed heuristics for the TSS problem. Shakarian et al. [27] presented a heuristic and tested it on various real-world graphs. Cordasco et al. [12] discussed a heuristic for the WTSS problem and showed that the heuristic provides an optimal solution for a special case of the WTSS problem on complete graphs, where $b_i = g_i$ for all nodes i in V. Mathematical programming approaches for the TSS problem seem limited. Two formulations are discussed in Ackerman et al. [1], Shakarian et al. [27], and Spencer et al. [28] for the TSS problem. However, natural adaptations of these formulations for the WTSS problem are weak in the sense that the gap between the optimal objective value of the IP formulation and that of its linear programming (LP) relaxation is large. Even for the WTSS problem on trees, which is polynomially solvable (shown later in Section 2), LP relaxations of the natural adaptations are weak and provide fractional solutions.

Our research is motivated by the desire to develop mathematical programming approaches and a better understanding of the underlying polytopes for this fundamental problem. To this end, within this paper, we focus on the WTSS problem on trees and cycles. There are several reasons to limit our focus. First, for many hard combinatorial optimization problems, developing tight formulations for polynomially solvable special cases has provided a natural direction (see e.g., [15,22]) for developing strong formulations for the general case. Trees and cycles represent (in some sense) the structurally simplest nontrivial graphs for studying the WTSS problem, which motivates our focus on them. Additionally, Adcock et al. [2] empirically demonstrate that real social and information networks have meaningful large-scale tree-like structures in terms of their decomposability. In fact, Kwak et al. [20] show that the information propagation process on Twitter can be represented as a tree.

Our discussion of the literature has solely focused on the TSS problem. There is a larger body of literature on influence maximization, largely driven by a seminal paper by Kempe et al. [19]. Chen et al.'s [9] monograph and Li et al.'s [21] survey paper nicely summarize some of the most relevant work in this area. Kempe et al. were the first to model the influence maximization problem (IMP) as an optimization problem. They employed a well-known "threshold" model from mathematical sociology (see [16]) that explicitly represents the step-by-step dynamics of adoption. They considered a budgeted version of the problem. Given a budget of k seed products, the goal is to identify k individuals to target so that the product's adoption is *maximized* in the social network. They built their model in a randomized (i.e., stochastic) setting and show that the problem is NP-hard. Based on the submodularity property of the objective function, due to the particular randomized assumption in the problem data they make, they developed a $\left(1 - \frac{1}{e}\right)$-approximation algorithm for the problem. Their research led to work on the problem that mostly focused on speeding up the algorithm (because their algorithm has a costly simulation in each step). Wu and Küçükyavuz [31] studied the IMP in a two-stage stochastic programming framework. They proposed a delayed constraint generation algorithm by taking advantage of the submodularity property in the objective function and conducted computational experiments on the stochastic version of the IMP. Although promising, their approach is only computationally viable for seed sets of size five. Recently, Nannicini et al. [23] considered a robust version of the IMP and presented a novel mixed integer programming formulation, derived from a bilevel formulation of the problem. They apply an exact branch-and-cut approach for problems with up to 100 nodes, and find that the approach is only computationally viable for small seed sets with up to five nodes.

Günneç et al. [17,18] considered a variation of the WTSS problem called the least cost influence problem (LCIP), where partial incentives are allowed. In other words, instead of having to pay the entire amount b_i to an initial adopter i, it is also possible to pay a partial incentive to intermediate or later adopters so that the sum of the incoming influences from their active neighbors and the partial incentive is greater than their threshold value. Günneç et al. [17] considered the LCIP in the setting where the neighbors of a node may exert unequal influence, and where the goal is to influence only a given proportion of the network; they showed it to be NP-hard. They also showed the LCIP where the entire network must be influenced, and the neighbors of a node exert equal influence to be APX-hard. They described a totally unimodular (TU) formulation for the LCIP on trees. Günneç et al. [18] build upon this TU formulation for the LCIP on trees, and show how to apply it to arbitrary graphs. They design a branch-and-cut procedure that is able to find solutions that are on average within 1.87% on real-world graphs, with up to 155 000 nodes and 327 000 edges. Fischetti et al. [14] considered the LCIP in the more general setting where the neighbors of a node may exert unequal influence, the entire network need not be influenced, and the influence structure can be nonlinear; they proposed a novel set covering based formulation for this version of the LCIP. Using this formulation, they develop a branch-and-cut approach, where all variables (of which there are exponentially many) are enumerated first. In this more general setting, their approach can be applied to simulated graph instances, with up to 100 nodes and an average degree of up to 16.

1.1 | Our contributions and the organization of the rest of the paper

We first show that the WTSS problem on trees can be solved by a linear-time algorithm in Section 2. Our algorithm differs significantly from Chen [8], and his algorithm can be viewed as a special case of ours. In Section 3, we study the polytope of the WTSS problem on trees. In Section 3.1, we present a tight and compact extended formulation for the WTSS problem on trees. (The linear-time algorithm discussed in Section 2 is helpful in proving its tightness.) In Section 3.2, we project the extended formulation onto the space of the natural node variables yielding the polytope of the WTSS problem on trees. This projection leads to an exponentially sized set of valid inequalities whose polynomial-time separation is also discussed. Building upon the result for trees, in Section 4 we focus on the WTSS problem on cycles. We present a linear-time algorithm in Section 4.1, and in Section 4.2 derive the complete description of the polytope for the WTSS problem on cycles. In Section 5, we discuss how the formulation we derive in this paper for the WTSS problem on trees provides a framework for developing a strong formulation for the WTSS problem on arbitrary graphs (which is APX-hard). In a parallel paper [26], this strong formulation for the WTSS problem on arbitrary graphs is embedded into a branch-and-cut approach and tested on 180 real-world graph instances, with up to approximately 155 000 nodes and 327 000 edges. The branch-and-cut approach finds solutions that are on average 0.90% away from optimality, and solves 60 out of the 180 instances to optimality. Section 5 also discusses how Fischetti et al.'s [14] set covering formulation for the LCIP can be adapted to the WTSS problem on arbitrary graphs. Finally, Section 6 provides concluding remarks and directions for future research.

2 | LINEAR-TIME ALGORITHM FOR THE WTSS PROBLEM ON TREES

We present a dynamic programming (DP) algorithm to solve the WTSS 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 leaf nodes. The DP algorithm then solves the star subproblem for two cases. Consider the link that connects the star to the rest of the tree. We refer to the node adjacent to the central node on this link as its parent. In the first case, there is influence coming into the central node from its parent (this kind of influence is referred to as external influence), whereas in the second case, there is no external influence. Next, the star is contracted into a leaf node for its parent node's star network. This process of solving star subproblems for two cases, followed by the 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 there is no external influence. After we exhaust all subproblems, a backtracking method is used to identify a final solution, which combines the solution candidates of the star subproblems for the tree.

Algorithm 1 shows the pseudocode of the proposed algorithm. To create an ordering among the subproblems considered in the algorithm, it is convenient (but not necessary) to arbitrarily pick a root node (which we denote by r). We then prioritize the subproblems in the 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 nonleaf nodes of the tree in reverse BFS order. The global variable C^* contains the cost of the optimal solution.

Algorithm 1. Algorithm for the WTSS Problem on Trees

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

For a star subproblem, consider the situation when there is no external influence. We wish to find the minimum weight solution that makes all nodes active. Observe that a leaf node only requires one active neighbor to make itself active. Therefore, in this star network, if the central node is selected, all nodes become active. However, there is another possibility. Denote the central node by c and refer to this star as star c. Sort all leaf nodes in ascending order of their weight. Then, if the total weight of the g_c smallest weight leaf nodes is less than the central node's weight, b_c, we can select the first g_c leaf nodes to activate the central node. After that, the active central node activates the remaining leaf nodes. Consequently, when the central node receives no external influence, the candidate solution is the one with the smaller weight between the central node and the first g_c leaf nodes. Note that this is a crucial difference between the TSS and the WTSS problem. For the TSS problem, where all nodes have $b_i = 1$, a leaf node of a star will never be selected in the target set. This makes it possible to find a solution for the TSS problem (as in [8]) easily in one shot (without backtracking or the contraction procedure that we discuss next). In the situation where the central node receives external influence from its parent node, we only need to select the first $(g_c - 1)$ leaf nodes to activate the central node. Hence, when the central node receives external influence, the candidate solution is the one with the smaller weight between the central node and the first $(g_c - 1)$ leaf nodes.

Algorithm 2. StarHandling

```
Input: star $c$
 1:    if $b_c < \sum_{l \in s_{(g_c-1)}} b_l$ then
 2:        $X_I^c \leftarrow c$, $X_{NI}^c \leftarrow c$, and $C^* = C^* + b_c$.
 3:        if star $c$ is not the last star                                ▷ Case 1
 4:            Contract star $c$ to a node with weight 0 and threshold 1.
 5:        end if
 6: else if $b_c > \sum_{l \in s_{g_c}} b_l$ then
 7:        $X_I^c \leftarrow s_{(g_c-1)}$, $X_{NI}^c \leftarrow s_{g_c}$, and $C^* = C^* + \sum_{l \in s_{(g_c-1)}} b_l$.
 8:        if star $c$ is not the last star                                ▷ Case 2
 9:            Contract star $c$ to a node with weight $b_{(g_c)}$ and threshold 1.
10:        else
11:            $C^* = C^* + b_{(g_c)}$.
12:        end if
13: else
14:        $X_I^c \leftarrow s_{(g_c-1)}$, $X_{NI}^c \leftarrow c$, and $C^* = C^* + \sum_{l \in s_{(g_c-1)}} b_l$.
15:        if star $c$ is not the last star                                ▷ Case 3
16:            Contract star $c$ to a node with weight $b_c - \sum_{l \in s_{(g_c-1)}} b_l$ and threshold 1.
17:        else
18:            $C^* = C^* + b_c - \sum_{l \in s_{(g_c-1)}} b_l$.
19:        end if
20: end if
```

After we determine the solution candidates for the current star subproblem, the star is contracted into a single leaf node for its parent's star subproblem. This new single node has its threshold set as 1. To determine the weight of the contracted node, observe in the optimal solution that the star (which this contracted node represents) either receives influence from its parent (with the candidate solution corresponding to when there is external influence) or sends influence to its parent (with the candidate solution corresponding to when there is no external influence). Observe, for a star, that the weight of the candidate solution obtained when there is external influence is less than or equal to the weight of the candidate solution when there is no external influence. Thus, the weight of the candidate solution obtained when there is external influence must be incurred at a minimum. The incremental amount, representing the difference in the weight between the candidate solution without external influence and the candidate solution with external influence, must be incurred if the star sends influence to its parent in the optimal solution; it is represented as the weight of the contracted node. There are three cases to consider.

Case 1. If the central node's weight is smaller than the total weight of the first $(g_c - 1)$ smallest weight leaf nodes, then both candidate solutions (when there is external influence and otherwise) select the central node for the current star. Thus, the contracted node has a weight of 0.

Case 2. If the central node's weight is bigger than the total weight of the first g_c lowest weight leaf nodes, the candidate solution for the external influence situation selects the first $(g_c - 1)$ smallest weight leaf nodes, while the candidate solution for the situation with no external influence selects the first g_c smallest weight leaf nodes. Thus, the weight of the contracted node is equal to the weight of the g_cth smallest weight leaf node, denoted as $b_{(g_c)}$.

Case 3. If the central node's weight is not in the above two cases, the candidate solution for the situation with external influence selects the first $(g_c - 1)$ smallest weight leaf nodes, while the candidate solution for the no-external influence situation selects the central node. Thus, the weight of the contracted node is equal to the difference between the central node's weight and the total weight of the first $(g_c - 1)$ cheapest leaf nodes.

We summarize the above procedure in Algorithm 2, StarHandling with the following notation: let X_I^c denote the solution candidate with external influence, X_{NI}^c denote the solution candidate without external influence, s_{g_c} denote the set of the g_c cheapest leaf nodes, and $s_{(g_c-1)}$ denote the set of the (g_c-1) cheapest leaf nodes.

Algorithm 3. SolutionBacktrack

```
Input: the last star with root node $r$ and its solution $X_{NI}^r$
 1: $X^* \leftarrow X_{NI}^r$.
 2: if $X^*$ is $r$ then
 3:     $\forall l \in L(r) \cap NL$ call EXT-INFLUENCE $(l, X^*)$.
 4: else
 5:     $\forall l \in s_{g_r} \cap NL$ call NOEXT-INFLUENCE $(l, X^*)$.
 6:     $\forall l \in (L(r) \backslash s_{g_r}) \cap NL$ call EXT-INFLUENCE $(l, X^*)$.
 7: end if
 8: return $X^*$.
 9: function EXT-INFLUENCE $(c, X)$
10:     $X \leftarrow (X \backslash c) \cup X_I^c$.
11:     $\forall l \in L(c) \cap (X_I^c \cap NL)$ call NOEXT-INFLUENCE $(l, X)$.
12:     $\forall l \in (L(c) \backslash X_I^c) \cap NL$ call EXT-INFLUENCE $(l, X)$.
13:     return $X$.
14: end function
15: function NOEXT-INFLUENCE $(c, X)$
16:     $X \leftarrow (X \backslash c) \cup X_{NI}^c$.
17:     $\forall l \in L(c) \cap (X_{NI}^c \cap NL)$ call NOEXT-INFLUENCE $(l, X)$.
18:     $\forall l \in (L(c) \backslash X_{NI}^c) \cap NL$ call EXT-INFLUENCE $(l, X)$.
19:     return $X$.
20: end function
```

After we obtain the solution of the last star, which has the root node as its central node, we invoke a backtracking procedure to obtain the final solution for this tree. For each leaf node in this star, we know whether or not there is external influence coming into it. For instance, if the central node is selected, then the central node sends out influence to all of its leaf nodes. If the g_c cheapest leaf nodes are selected, then these g_c cheapest leaf nodes do not receive external influence, but the remaining leaf nodes do. With this information, we can now proceed down the tree, incorporating the partial solution at a node based on whether or not it receives external influence (which we now know). This backtracking procedure is described in Algorithm 3, SolutionBacktrack. We use $L(c)$ to denote the set of leaf nodes in the star c, NL to denote the set of nonleaf nodes in the tree, and X^* to denote the final solution of the tree. In this algorithm, we have two recursive functions: EXT-INFLUENCE and NOEXT-INFLUENCE, corresponding to with and without external influence, respectively. These two functions choose the solution for a star c and then recursively choose solutions for stars whose central nodes are leaf nodes of the star c.

Proposition 1. *The WTSS problem on trees can be solved in* $O(|V|)$ time.

Proof. The correctness of the algorithm can be established via induction, using arguments identical to the preceding discussion. We now discuss the running time. There are at most $|V|-1$ stars. For each star c, we need to find the g_c cheapest children, which 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. [13]. Thus, it takes $O(deg(c))$ time to go through the list to collect the g_c cheapest 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. ■

Figure 1A displays an instance of the WTSS problem. There are 11 nodes, and the numbers beside a node are its weight and threshold. The root of this tree is node 4. Stars 1, 2, and 3 correspond to Cases 1, 2, and 3, respectively. Star 1 has $X_{NI}^1 = X_I^1 = \{1\}$. Star 2 has $X_{NI}^2 = \{7,8,9\}$ and $X_I^2 = \{7,8\}$. Star 3 has $X_{NI}^3 = \{3\}$ and $X_I^3 = \{10\}$. After we contract these stars, we have the final star in Figure 1B and the solution $X_{NI}^4 = \{1,2\}$. Next, we start the backtracking procedure. Initially $X^* = X_{NI}^4 = \{1,2\}$. Both stars 1 and 2 do not receive external influence, while star 3 receives external influence. Hence,

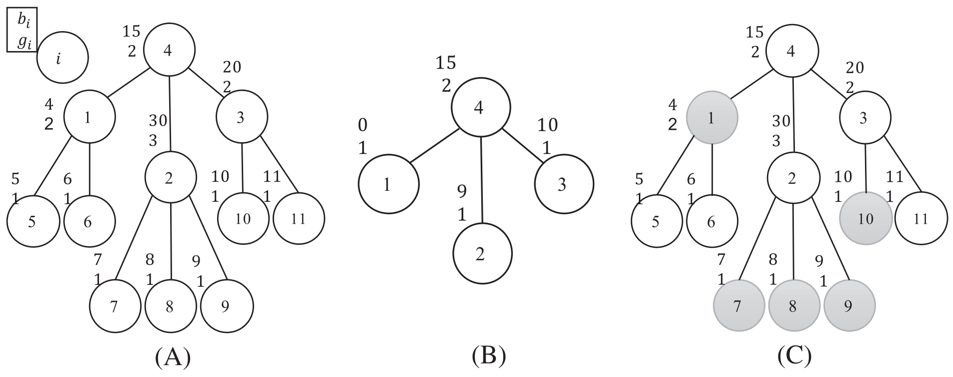


FIGURE 1 (A) WTSS instance, (B) Final star. (C) Solution of Algorithm 1

$X^* = X^* \backslash \{1,2\} \cup X_{NI}^1 \cup X_{NI}^2 \cup X_I^3$, which yields $X^* = \{1,7,8,9,10\}$ and $C^* = 4+7+8+9+10 = 38$. Figure 1C illustrates the solution obtained by Algorithm 1, where the selected nodes are shaded.

3 | POLYTOPE OF THE WTSS PROBLEM ON TREES

In this section, we first discuss existing formulations for the TSS problem in the literature. Then, we present a tight and compact extended formulation for the WTSS problem on trees. After that, we project the extended formulation onto the natural node space, which gives the complete description of the WTSS problem on trees. This projection leads to an exponentially sized set of valid inequalities. We also present a polynomial-time separation algorithm for it.

3.1 | A tight and compact extended formulation

A combinatorial optimization problem can be formulated as an IP in multiple ways. The standard way (see [11,24]) to compare different IP formulations for the same problem is to solve their LP relaxations (as this has implications for the computational tractability of a formulation). Then, these IP formulations are evaluated by the optimal objective values of their LP relaxations. Given two different IP formulations, A and B, of a given minimization problem, let z_A^{LP} and z_B^{LP} be the optimal objective value of their LP relaxations, respectively. We say that formulation A is stronger (or better) than formulation B if $z_A^{LP} > z_B^{LP}$. Typically, a stronger formulation is more computationally efficient [5].

There are two formulations for the TSS problem in the literature. A time-indexed formulation was proposed independently by Shakarian et al. [27] and Spencer et al. [28]. To model the order in which nodes become active, an artificial time index t is created, taking values from 1 to $|V|$. The formulation uses a binary variable $x_{i,t}$, which is set as 1 if node i is active in time period t, and is 0 otherwise. For any node $i \in V$, let $n(i)$ denote the set of node i's neighbors. The model "TimeIndexed" (applied to the WTSS) is as follows:

$$\text{(TimeIndexed)} \quad \text{Minimize} \quad \sum_{i \in V} b_i x_{i,1} \tag{1}$$

$$\text{Subject to} \quad x_{i,|V|} = 1 \qquad \forall i \in V, \tag{2}$$

$$g_i x_{i,t-1} + \sum_{j \in n(i)} x_{j,t-1} \geq g_i x_{i,t} \qquad \forall i \in V, \quad t \in \{2,3,\ldots,|V|\}, \tag{3}$$

$$x_{i,t} \in \{0,1\} \quad \forall i \in V, t \in \{1,2,\ldots,|V|\}. \tag{4}$$

The objective function (1) minimizes the total weight of the nodes activated in time period 1 (i.e., the nodes selected in the target set). Constraint (2) makes sure that all nodes are active in time period $|V|$ (i.e., by the end of the diffusion process). Constraint (3) states that a node i is active in the time period t only if it was active in the time period $t-1$, or if it has at least g_i neighbors that were active in the time period $t-1$. We note that this formulation is weak for the instance in Figure 2A. The LP relaxation of TimeIndexed has a fractional optimal solution with $x_{1,1} = x_{2,1} = 0.0334$ and $x_{3,1} = x_{4,1} = x_{5,1} = x_{6,1} = 0$, and an objective value of 0.0668. However, the optimal integer solution selects nodes 1 and 2 (i.e., $x_{1,1} = x_{2,1} = 1$ and $x_{3,1} = x_{4,1} = x_{5,1} = x_{6,1} = 0$) and has an objective value of 2.

Ackerman et al. [1] introduced a different formulation for the TSS. Here, for each node $i \in V$, a binary variable x_i is created denoting whether node i is selected in the target set. Then, to model the (implicit) order in terms of when the nodes become active, a linear order is created among the nodes. For any two distinct nodes i and j in V, two binary variables h_{ij} and h_{ji} are

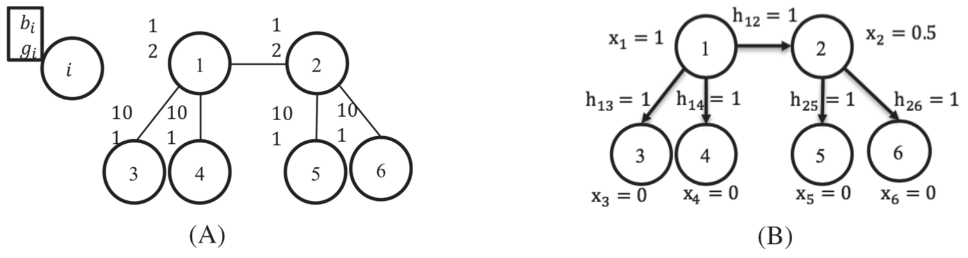


FIGURE 2 (A) WTSS instance. (B) Fractional optimal solution to ACK

created, with $h_{ij} = 1$ if node i is before node j in the linear order, and $h_{ji} = 1$ if node j is before node i in the linear order. Notice that the pair of variables h_{ij}, h_{ji} is created for every pair of nodes in the graph, regardless of whether they have an edge between them in the graph. The model "ACK" (applied to the WTSS) is as follows:

$$\text{(ACK)} \quad \text{Minimize} \quad \sum_{i\in V} b_i x_i \tag{5}$$

$$\text{Subject to} \quad h_{ij} + h_{ji} = 1 \qquad \forall i \neq j \in V, \tag{6}$$

$$\sum_{j\in n(i)} h_{ji} + g_i x_i \geq g_i \qquad \forall i \in V, \tag{7}$$

$$h_{ij} + h_{jk} + h_{ki} \leq 2 \qquad \forall i \neq j \neq k \in V, \tag{8}$$

$$x_i \in \{0,1\} \qquad \forall i \in V, \tag{9}$$

$$h_{ij} \in \{0,1\} \qquad \forall i \neq j \in V. \tag{10}$$

The objective function (5) minimizes the total weight of the nodes selected in the target set. Constraint (6) makes sure that either node i is before node j, or node j is before node i in linear order. Constraint (7) states that a node $i \in V$ must be selected in the target set, or at least g_i of its neighbors must become active before node i. Constraint (8) ensures that there is linear ordering.

Figure 2A provides a WTSS instance, and Figure 2B describes a fractional optimal solution to the LP relaxation of ACK. This solution has $x_1 = 1$, $x_2 = 0.5$, with all other x decision variables equal to zero. The nonzero h variables for the edges in the graph is shown in Figure 2B. For the remaining node pairs $h_{15} = h_{16} = h_{23} = h_{24} = h_{34} = h_{35} = h_{36} = h_{45} = h_{46} = h_{56} = 1$ and the rest are zero. The objective value is 1.5. While ACK is stronger than TimeIndexed, it is still weak, and solving its LP relaxation does not provide integer solutions for the WTSS problem on trees.

We now present a tight and compact extended formulation for the WTSS problem on trees. From the input graph G, we create a new graph G^t by subdividing each edge. For each edge $\{i, j\} \in E$, insert a dummy node d. Let D denote the set of dummy nodes. Since the dummy nodes have effectively split each edge into two in the original graph, 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. Let E^t denote the set of edges in G^t ($G^t = (V \cup D, E^t)$). The dummy nodes cannot be selected in the target set, and all have a threshold of 1 (thus, if one of its neighbors is activated, the dummy node will become activated and will propagate the influence to the other neighbor). As before, for each node $i \in V$, a binary variable x_i denotes whether or not node i is selected in the target set (these are the natural node variables). For each edge $\{i, d\} \in E^t$, where $i \in V$ and $d \in D$ (notice that G^t is bipartite and E^t only contains edges between the nodes in V and D), create two binary arc variables y_{id} and y_{di} to represent the direction of influence propagation. If node i sends influence to node d, y_{id} is 1, and 0 otherwise. As before, for any node $i \in V \cup D$, $n(i)$ denotes the set of node i's neighbors. We can now write the following compact extended formulation "BIP$_{\text{tree}}$" for the WTSS problem on trees:

$$(\text{BIP}_{\text{tree}}) \quad \text{Minimize} \quad \sum_{i\in V} b_i x_i \tag{11}$$

$$\text{Subject to} \quad \sum_{i\in n(d)} y_{id} \geq 1 \qquad \forall d \in D, \tag{12}$$

$$x_i \leq y_{id} \qquad \forall i \in V, d \in n(i), \tag{13}$$

$$y_{id} + y_{di} = 1 \qquad \forall \{i, d\} \in E^t, \tag{14}$$

$$\sum_{d\in n(i)} y_{di} + g_i x_i \geq g_i \qquad \forall i \in V, \tag{15}$$

$$x_i \in \{0,1\} \qquad \forall i \in V, \tag{16}$$

$$y_{id}, y_{di} \in \{0,1\} \qquad \forall \{i, d\} \in E^t. \tag{17}$$

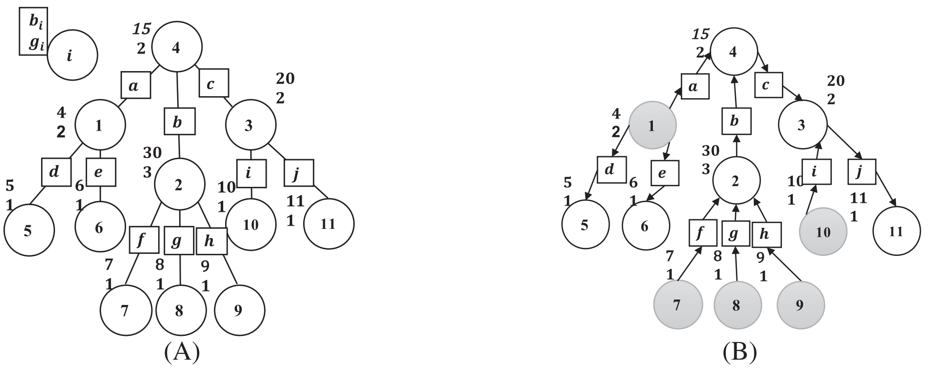


FIGURE 3 (A) G^t. (B) A valid solution to BIP_{tree}

Constraint (12) says that each dummy node has at least one incoming arc (since dummy nodes cannot be selected and have a threshold of 1). Constraint (13) says that if a node is selected, then it sends out influence to all of its neighbors. Notice that this type of constraint (e.g., $x_i \leq h_{ij}$) would not be valid in ACK, since there could be two neighboring nodes that are in the target set. Constraint (14) makes sure that on each edge, influence is only propagated in one direction. Constraint (15) states that a node $i \in V$ must be selected or must have g_i or more incoming arcs. If a node $i \in V$ is selected, then it has no incoming arcs from constraint (13), and constraint (15) is satisfied. On the other hand, if a node $i \in V$ is not selected, it must have at least g_i incoming arcs. Constraints (16) and (17) are binary constraints. We now show the validity of BIP_{tree}.

Proposition 2. *BIP_{tree} is a valid formulation for the WTSS problem on trees.*

Proof. From the discussion above, we can see that if $(\mathbf{x}, \mathbf{y})$ is a feasible solution to BIP_{tree}, then $\mathbf{x}$ must be a feasible target set (one that activates all nodes in the graph). We now show how any feasible target set vector $\mathbf{x}$ can be extended as a feasible solution $(\mathbf{x}, \mathbf{y})$ to BIP_{tree}, proving its validity. Let P be the set of nodes selected in a feasible target set. For each p in P, we set x_p as 1 and set the remaining x variables as 0. Next, all outgoing arcs of a node p in P are set as 1, and all of its incoming arcs are set as 0, that is, $y_{pd} = 1$ and $y_{dp} = 0 \,\forall\, p \in P,\ d \in n(p)$ (this ensures that constraint (13) is satisfied). Next, for any dummy node d that has y values set for only one of its adjacent edges, we set the y values, as follows. Without loss of generality, let nodes i and j be adjacent to dummy node d, and y_{id} is now set as 1. Then, we set $y_{jd} = 0$ and $y_{dj} = 1$ to propagate the influence. Afterwards, we check for any new nodes $i \in V$ that have been activated by incoming influence (arcs). If so, node i sends out influence to those adjacent dummy nodes that do not send influence to it, that is, $y_{id} = 1$ and $y_{di} = 0$ for all $d \in n(i)$, where y_{id} and y_{di} are not yet set. We repeat this whole procedure until all nodes are active, and all x and y values are set. Observe that the procedure so far satisfies constraints (12)-(15). ■

Figure 3A shows the transformed graph G^t of the instance in Figure 1A. Figure 3B illustrates the outcome of the procedure described in Proposition 2.

Next, we show that BIP_{tree} is a tight formulation. The linear relaxation of BIP_{tree} is the following LP, which we refer to as "LP_{tree}":

$$(\text{LP}_{\text{tree}}) \quad \text{Minimize} \quad \sum_{i \in V} b_i x_i \tag{18}$$

$$\text{Subject to} \quad (12), (13), (14), (15)$$

$$x_i \geq 0 \qquad \forall i \in V, \tag{19}$$

$$y_{id} \geq 0 \qquad \forall i \in V, d \in n(i), \tag{20}$$

$$y_{di} \geq 0 \qquad \forall d \in D, i \in n(d). \tag{21}$$

Let u_d, w_{id}, z_{id}, and v_i be dual variables for constraint sets (12)-(15), respectively. The dual to LP_{tree} can be written as follows, which we refer to as "DLP_{tree}":

$$(\text{DLP}_{\text{tree}}) \quad \text{Maximize} \quad \sum_{d\in D} u_d + \sum_{i\in V} g_i v_i + \sum_{\{i,d\}\in E^t} z_{id} \tag{22}$$

$$\text{Subject to} \quad w_{id} + u_d + z_{id} \le 0 \qquad \forall i \in V, d \in n(i), \tag{23}$$

$$v_i + z_{id} \le 0 \qquad \forall d \in D, i \in n(d), \tag{24}$$

$$g_i v_i - \sum_{d\in n(i)} w_{id} \le b_i \qquad \forall i \in V, \tag{25}$$

$$u_d \ge 0 \qquad \forall d \in D, \tag{26}$$

$$v_i \ge 0 \qquad \forall i \in V, \tag{27}$$

$$w_{id} \ge 0 \qquad \forall i \in V, d \in n(i). \tag{28}$$

Let conv(X) denote the convex hull of the feasible target set vectors **x**, and let ETSS denote the feasible region of LP_{tree}. We now show that $\text{Proj}_{\mathbf{x}}(ETSS) = \text{conv}(X)$. To prove this result, we demonstrate for every objective function in the x-space, an optimal solution $(\mathbf{x}, \mathbf{y})$ with integral **x**. To do so, we use the complementary slackness (CS) conditions between LP_{tree} and DLP_{tree}.

$$\left(g_i - \sum_{d\in n(i)} y_{di} - g_i x_i\right) v_i = 0 \qquad \forall i \in V, \tag{29}$$

$$\left(1 - \sum_{i\in n(d)} y_{id}\right) u_d = 0 \qquad \forall d \in D, \tag{30}$$

$$(x_i - y_{id}) w_{id} = 0 \qquad \forall i \in V, d \in n(i), \tag{31}$$

$$\left(b_i - g_i v_i + \sum_{d\in n(i)} w_{id}\right) x_i = 0 \qquad \forall i \in V, \tag{32}$$

$$(-v_i - z_{id}) y_{di} = 0 \qquad \forall d \in D, i \in n(d), \tag{33}$$

$$(-w_{id} - u_d - z_{id}) y_{id} = 0 \qquad \forall i \in V, d \in n(i). \tag{34}$$

Theorem 1. *$Proj_{\mathbf{x}}(ETSS) = conv(X)$.*

Proof. For any objective function in the x-space, we use the primal solution from Algorithm 1. Then, we construct a dual feasible solution and show that the pair of primal and dual solutions satisfy the CS conditions. The full proof is in Appendix A. ■

Theorem 1 shows that every extreme point solution found by LP_{tree} has x variables binary (this does not necessarily mean that ETSS is integral). Interestingly, we can show that any extreme point of LP_{tree} with x variables binary also has the y variables binary (which also implies that LP_{tree} provides solutions with both x and y variables integral).

Theorem 2. *Every extreme point of ETSS with x variables binary has y variables binary.*

Proof. Assume that this is not true, and that there is an extreme point $(\mathbf{x}^*, \mathbf{y}^*)$ where $\mathbf{x}^*$ is binary and $\mathbf{y}^*$ is fractional. Observe that when $x_i^* = 1$, $y_{id}^* = 1$, and $y_{di}^* = 0$ for all $d \in n(i)$ due to constraints (13) and (14). Since the x variables are binary, this implies that the fractional y values must correspond to arcs adjacent to a node $i \in V$ that is not in the target set.

We look at the supporting graph that is formed by the fractional $\mathbf{y}^*$ values. In other words, only arcs with fractional values of y are kept in the supporting graph. Consider a connected component of the supporting graph, as shown in Figure 4A. We show that an end node in this connected component that is a dummy node (e.g., nodes c and d) must have constraint (12) satisfied as a nonbinding constraint, and an end node of this connected component that is not a dummy node (e.g., node 1) must have constraint (15) satisfied as a nonbinding constraint. When an end node is in D, it has one fractional incoming arc. Because it is an end node and constraint (12) is satisfied, the other incoming arc must have a value of 1. Thus, constraint (12) is nonbinding. When an end node is in V, it has one fractional incoming arc, and all other incoming arcs are integers. Given that g_i is an integer, constraint (15) must be satisfied as a nonbinding constraint.

Consider any two end nodes, denoted as s and t, in a connected component of the supporting graph. Let $P_{st} = \{(s, l), \ldots, (k, t)\}$ and $P_{ts} = \{(t, k), \ldots, (l, s)\}$ denote the fractional paths from nodes s to t and from nodes t to s, respectively. We

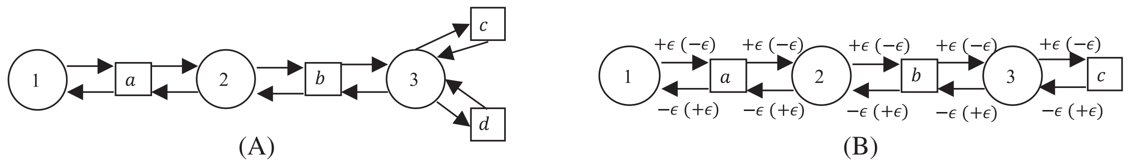


FIGURE 4 Illustrating proof of Theorem 2. (A) A connected component of the supporting graph formed by the fractional y^* values. (B) Constructing two new feasible solutions using the fractional paths between the end node 1 and c

construct two new feasible solutions $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ and $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$, as follows:

$$\bar{\mathbf{x}} = \mathbf{x}^*; \quad \bar{y}_a = \begin{cases} y_a^* + \epsilon, & a \in P_{st}, \\ y_a^* - \epsilon, & a \in P_{ts}, \\ y_a^*, & a \in E^t \backslash \{P_{st} \cup P_{ts}\}, \end{cases} \qquad \text{and} \quad \tilde{\mathbf{x}} = \mathbf{x}^*; \quad \tilde{y}_a = \begin{cases} y_a^* - \epsilon, & a \in P_{st}, \\ y_a^* + \epsilon, & a \in P_{ts}, \\ y_a^*, & a \in E^t \backslash \{P_{st} \cup P_{ts}\}, \end{cases}$$

where ϵ is a sufficiently small positive value (ϵ should be less than the slack in both the nonbinding constraints at the two end nodes s and t and should also satisfy $0 \le y_a^* \pm \epsilon \le 1$ for all $a \in \{P_{st} \cup P_{ts}\}$). Thus, $(\mathbf{x}^*, \mathbf{y}^*)$ is not an extreme point because $(\mathbf{x}^*, \mathbf{y}^*) = \frac{1}{2}(\bar{\mathbf{x}}, \bar{\mathbf{y}}) + \frac{1}{2}(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$. ■

3.2 | Projecting BIP$_{\text{tree}}$ to obtain the polytope on trees

To derive the polytope of the WTSS problem on trees, we follow a method, proposed by Balas and Pulleyblank [4], which is based on a theorem of the alternatives to project the extended formulation BIP$_{\text{tree}}$ onto the node (i.e., x) space by projecting out all arc (i.e., y) variables.

Consider LP$_{\text{tree}}$. Because $y_{id} + y_{di} = 1$, we first project out all y_{id} variables, setting them to $1 - y_{di}$ (so we only have x_i and y_{di} variables in the formulation) and obtain the following formulation whose feasible region is denoted as P_d:

$$\text{Minimize} \quad \sum_{i \in V} b_i x_i \tag{35}$$

$$\text{Subject to} \quad -\sum_{i \in n(d)} y_{di} \ge -1 \qquad \forall d \in D, \tag{36}$$

$$-y_{di} - x_i \ge -1 \qquad \forall i \in V, \ \ d \in n(i), \tag{37}$$

$$\sum_{d \in n(i)} y_{di} + g_i x_i \ge g_i \qquad \forall i \in V, \tag{38}$$

$$y_{di} \ge 0 \qquad \forall \{i, d\} \in E^t, \tag{39}$$

$$x_i \ge 0 \qquad \forall i \in V. \tag{40}$$

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

$$w_i - u_d - v_{id} \le 0 \qquad \forall i \in V, \ \ d \in n(i), \tag{41}$$

$$w_i \ge 0, u_d \ge 0, v_{id} \ge 0 \qquad \forall i \in V, \ \ d \in n(i). \tag{42}$$

Here, u_d, v_{id}, and w_i are dual multipliers corresponding to constraints (36)-(38), respectively. If P_d is written in matrix notation as $\{(\mathbf{x},\mathbf{y}) : A\mathbf{x} + G\mathbf{y} \ge \mathbf{b}, (\mathbf{x},\mathbf{y}) \ge \mathbf{0}\}$, based on Balas and Pulleyblank [4], then any feasible vector (u, v, w) to $\mathcal{W}$ defines a valid inequality: $(\mathbf{u},\mathbf{v},\mathbf{w})^T A\mathbf{x} \ge (\mathbf{u},\mathbf{v},\mathbf{w})^T \mathbf{b}$ to the projection of P_d (in the space of node (x) variables). Furthermore, the projection of P_d is described by the valid inequalities defined by the extreme rays of $\mathcal{W}$. Thus, in order to apply this method, we now identify the extreme rays of $\mathcal{W}$. We use these notations: first, $S \subseteq V$ and S is connected in the original graph G. Also, $G^t(S)$ denotes the induced subgraph of S in the transformed graph G^t and $D(S)$ denotes the set of dummy nodes belonging to $G^t(S)$. Finally, $a(S)$ denotes the set of nodes adjacent to set S in the transformed graph G^t.

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

Case 1. $u_d = \alpha$ *for one* $d \in D$. *All other* u, v, w *are* 0.
Case 2. $v_{id} = \alpha$ *for one* $\{i, d\} \in E^t$. *All other* u, v, w *are* 0.
Case 3. $w_i = \alpha$ *for all* $i \in S$. *Then* $u_d = \alpha$ *for all* $d \in D(S)$. *In addition, either* $v_{id} = \alpha$ *or* $u_d = \alpha$ *for all* $d \in a(S) \backslash D(S)$. *All other* u, v, w *are* 0.

Proof. Included in Appendix B. ■

Using these three types of extreme rays, we obtain the following inequalities to define the projection. Case 1's extreme directions generate the valid inequality $0 \geq -1$, which is not very useful, while Case 2's extreme directions generate $-x_i \geq -1$ or the inequalities, $x_i \leq 1$ for all $i \in V$. Given an extreme ray **r** of the form described in Case 3, define $V_i = \{d \in n(i) : v_{id} > 0\}$ for all $i \in S$ based on **r**. Consequently, Case 3's extreme directions generate the following valid inequality in the original graph G:

$$\sum_{i\in S}(g_i - |V_i|)x_i \geq \sum_{i\in S} g_i - |n(S)| - |S| + 1 \quad \forall i \in S \subseteq V, \;\; S \text{ is connected}, \;\; |V_i| = 0, 1, \ldots, \lambda_i^S.$$

Here, $\lambda_i^S = |V_i^S|$, where $V_i^S = n(i) \cap n(S)$ (in other words, V_i^S is the set of neighbors of node i in the original graph G that are outside the set S), and $n(S)$ denotes the neighbors of set S in the original graph G. Note that for a given set S, there are many extreme rays **r** satisfying Case 3, giving rise to sets V_i. The tightest valid inequality for a given set S is obtained when the coefficient of x_i on the left-hand side has the smallest value. This is obtained when $V_i = V_i^S$. After removing the dominated inequalities, the projection of P_d onto the x space and the polytope of the WTSS problem on trees is:

$$|n(S)| + (|S| - 1) + \sum_{i\in S}(g_i - \lambda_i^S)x_i \geq \sum_{i\in S} g_i \qquad \forall S \subseteq V, \;\; S \text{ is connected}, \tag{43}$$

$$0 \leq x_i \leq 1 \qquad \forall i \in V. \tag{44}$$

Proposition 3. *The valid inequalities* (43) *can be separated in* $O(|V|^3)$ *time.*

Proof. The generalized k-minimum spanning tree (k-MST) problem on trees is defined as follows: given a tree $G_B = (V_B, E_B)$ and an integer $k < |V_B|$, each node i in V_B has a nonnegative weight w_i and each edge in E_B has a nonnegative weight e_{ij}. The goal is to find a subtree T of k edges whose weight $\sum_{i\in T} w_i + \sum_{(i,j)\in T} e_{ij}$ is minimal.

Given an instance of the separation problem with the cardinality of S specified to be h, we show that it can be transformed to the generalized k-MST problem on trees with $k = 2(h-1)$. From the input graph G (which is a tree), we transform it into graph G^t, which was used to derive the extended formulation. Thus, $G^t = (V \cup D, E^t)$. Recall that we have a current solution **x**, and $deg(i)$ denotes the degree of node i. For all i in V, set its weight to $w_i = (1 - x_i)(deg(i) - g_i)$. Then, for all d in D, set its weight to $w_d = M$, where $M = \max\{deg(i) : i \in V\}$. This ensures that an optimal k-MST on G^t does not have nodes in D as leaves, because k is an even number. Lastly, for each $\{i, d\}$ in E^t, we have the edge weight $e_{id} = x_i$. We also set $k = 2(h-1)$ as the target cardinality for the k-MST.

For a node $i \in V$ and a given optimal k-MST T, let E_i^T be the set of its adjacent edges in T. The objective value of T is: $\sum_{i\in T\cap V}(1 - x_i)(\deg(i) - g_i) + \sum_{(i,d)\in T} x_i + (h-1)M = \sum_{i\in T\cap V}(\deg(i) - g_i - \deg(i)x_i + g_i x_i + |E_i^T|x_i) + (h-1)M$. Note that $\sum_{i\in T\cap V}|E_i^T| = 2(h-1)$ is the number of edges in T, which we add and subtract to the objective value of T, obtaining: $\sum_{i\in T\cap V}\{(g_i + |E_i^T| - \deg(i))x_i - g_i + (\deg(i) - |E_i^T|)\} + 2(h-1) + (h-1)M$. Let $\overline{T}$ denote $T \cap V$. In the original graph, $\deg(i) - |E_i^T|$ is identical to $\lambda_i^{\overline{T}}$. If we consider the nodes in $\overline{T}$, $\sum_{i\in\overline{T}}\lambda_i^{\overline{T}}$ is equal to $|\; n(\overline{T})\;|$: the cardinality of the set of neighbors of $\overline{T}$ in the original graph. Thus, the objective value is equal to $\sum_{i\in\overline{T}}(g_i - \lambda_i^{\overline{T}})x_i - \sum_{i\in\overline{T}} g_i + |n(\overline{T})| + (h-1)(M+2)$. Other than $(h-1)(M+2)$, this is exactly the left-hand side of constraint (43) with $S = \overline{T}$ after some rearrangement (we move $\sum_{i\in S} g_i$ to the left-hand side and move $|S| - 1$ to the right-hand side). Thus, if $\sum_{i\in\overline{T}}(g_i - \lambda_i^{\overline{T}})x_i - \sum_{i\in\overline{T}} g_i + |n(\overline{T})| < 1 - h$, we have a violated inequality. Or, if the objective value of this k-MST is strictly less than $(h-1)(M+1)$, we have found a violated inequality. Otherwise, there is no violated inequality for sets S with cardinality h.

Blum [7] provides an $O(k^2|V_B|)$ algorithm for the (k-MST) problem on trees. It returns the values of the best l-cardinality trees in G_B for all l values in the range $0 \leq l \leq k$. Thus, we only need to run it once by setting $k = |V_B| = 2\;|\;V|$. Thus, the time complexity is $O(|V|^3)$. ■

4 | THE WTSS PROBLEM ON CYCLES

In this section, we discuss the WTSS problem on a cycle. We first present a polynomial-time algorithm to solve the WTSS problem on cycles. We then describe the polytope of the WTSS problem on cycles. Before presenting our findings, we introduce some notation. In a cycle, a node has two neighbors. Thus, its threshold value is either one or two. We call a node i a 1-node if its g_i value is 1. Similarly, a node is called a 2-node if its g_i value is 2. Let G_1 be the set of 1-nodes and G_2 be the set of 2-nodes. We will refer to a connected component S as a segment because it is a line of nodes in a cycle. Given a segment S, there are two nodes adjacent to nodes not in S. We refer to these two nodes as end nodes. Finally, we use S^2 to denote a segment of nodes that has exactly two 2-nodes, and these two 2-nodes are its end nodes.

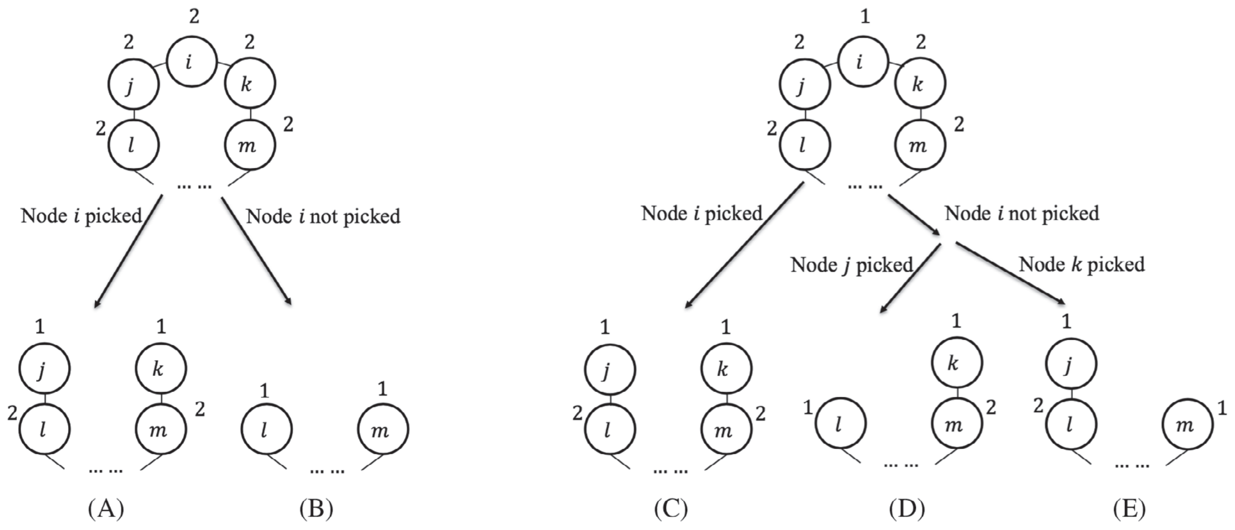


FIGURE 5 Illustration of the cases for the algorithm on cycles

4.1 | Linear-time algorithm for the WTSS problem on cycles

We make use of Algorithm 1 to solve the WTSS problem on cycles. First, we conduct two preprocessing steps. For each $S^2 \subset V$, if it contains more than one 1-node, we can replace these 1-nodes by the one with the smallest weight among them. In an optimal solution, at most, one 1-node will be selected in this S^2 segment, and it should be the one with the smallest weight. With this preprocessing step, each S^2 segment contains at most one 1-node. In the second preprocessing step, if this 1-node's weight is not strictly the smallest one in this S^2 segment, we can delete this 1-node and connect the 2-nodes to each other. This is because, if this 1-node is selected, then switching it with the 2-node having the smaller weight results in a better solution. Thus, at this point, each S^2 segment has at most one 1-node, and if it has one, this 1-node has the strictly lowest weight among these three nodes.

After preprocessing, if there are no 1-nodes left in the graph, we can arbitrarily pick a node i. Without loss of generality, let the two nodes adjacent to node i be nodes j and k, as shown in the top left of Figure 5. Then, we solve two subproblems for node i. One subproblem assumes that node i is selected. After accounting for the influence propagation from node i, it requires that the tree shown in Figure 5A be solved to ascertain the solution in this case. The other subproblem assumes that nodes j and k are selected at the same time. After accounting for the influence propagation from nodes j and k, it requires that the tree shown in Figure 5B be solved to ascertain the solution in this case.

If there are 1-nodes in the graph, we can arbitrarily pick a 1-node i. Without loss of generality, let those two 2-nodes adjacent to 1-node i be nodes j and k, as shown in the top right of Figure 5. Then, we consider two possibilities for node i. First, we assume that node i is selected. After accounting for the influence propagation from node i, the tree shown in Figure 5C must be solved to ascertain the solution in this case. If node i is not selected, then because nodes j and k are 2-nodes, at least one of them must be selected in the target set. Thus, we first consider the case that node j is selected when node i is not selected; then, we consider the case that node k is selected when node i is not selected. These require the solution of the trees shown in Figure 5D and 5E, respectively.

Overall, we call Algorithm 1 at most three times, and the preprocessing can be done in $O(|V|)$ time. Therefore, we have the following theorem:

Theorem 4. *The WTSS problem on cycles can be solved in $O(|V|)$ time.*

4.2 | Polytope of the WTSS problem on cycles

Now we derive the polytope for the WTSS problem cycles. Define a binary variable x_i for a node i in V. If node i is in the target set, $x_i = 1$. Otherwise, it is 0. Based on the idea of inequalities (43), we have the following formulation for cycles:

$$\text{(CIP)} \quad \text{Minimize} \quad \sum_{i \in V} b_i x_i \tag{45}$$

$$\text{Subject to} \quad |V| + \sum_{i \in V} g_i x_i \geq \sum_{i \in V} g_i, \tag{46}$$

$$2 + (|S| - 1) + \sum_{i \in S} (g_i - \lambda_i^S) x_i \geq \sum_{i \in S} g_i \qquad \forall S \subset V, \ S \text{ is connected}, \tag{47}$$

$$x_i \in \{0, 1\} \qquad \forall i \in V. \tag{48}$$

Constraint (47) is obtained from inequality (43) when $S \subset V$. The $2 + (|S| - 1)$ term is derived because the subgraph induced by S has 2 neighbors (i.e., $|n(S)| = 2$) and $(|S| - 1)$ edges. Constraint (46) is obtained from inequality (43) when $S = V$. The $|V|$ term is derived because there are $|V|$ edges in the cycle and $n(V) = \emptyset$, which also implies that $\lambda_i^V = 0$.

For constraint (46), moving the first term $|V|$ to the right-hand side, we can rewrite it as $\sum_{i\in G_1} x_i + 2\sum_{i\in G_2} x_i \geq |G_2|$. Then, increasing the coefficient of all 1-node variables from one to two produces a valid inequality: $2\sum_{i\in G_1} x_i + 2\sum_{i\in G_2} x_i \geq |G_2|$. Dividing both sides by two gives $\sum_{i\in V} x_i \geq \frac{|G_2|}{2}$. Finally, the left-hand side must be an integer; thus, replacing the right-hand side by its ceiling value provides a lifted valid inequality: $\sum_{i\in V} x_i \geq \left\lceil \frac{|G_2|}{2} \right\rceil$. Furthermore, when $|G_2| = 0$, it is trivial to see that the optimal solution is to pick the node with the smallest weight. Thus, we have the following valid inequality instead of (46):

$$\sum_{i\in V} x_i \geq \max\left\{1, \left\lceil \frac{|G_2|}{2} \right\rceil \right\}. \tag{49}$$

Next, we start removing redundant constraints in (47). First, when S has at most one 2-node, moving $2 + (|S| - 1)$ to the right-hand side makes the right-hand side 0 or -1 because $\sum_{i\in S} g_i$ is $|S| + 1$ or $|S|$. Thus, it is dominated by the nonnegativity constraints.

Second, in a cycle, given a segment S containing at least two 2-nodes, let nodes j and k be end nodes and node j be a 1-node. Also, in S, node h is the node adjacent to j. Thus, based on S, we have $(g_k - 1)x_k + g_h x_h + \sum_{i\in S\setminus\{j,k,h\}} g_i x_i \geq \sum_{i\in S} g_i - |S| - 1$, which is the summation of $x_h \geq 0$, and the inequality based on $S\setminus\{j\}$: $(g_k - 1)x_k + (g_h - 1)x_h + \sum_{i\in S\setminus\{j,k,h\}} g_i x_i \geq \sum_{i\in S\setminus\{j\}} g_i - |S| = \sum_{i\in S} g_i - |S| - 1$ because $g_j = 1$. The latter inequality dominates the former one. In other words, if both of these two end nodes are not 2-nodes, the inequality given by this set S is redundant.

Third, S has at least three 2-nodes, and the two end nodes are 2-nodes. Let nodes j and k be end nodes and node h be a 2-node in between j and k. We can break S into two segments: one contains the segment of nodes from node j to node h (denoted by S_j), and the other one contains the nodes from node h to node k (denoted by S_k). Then, based on S, we have $x_j + x_k + 2x_h + \sum_{i\in S\setminus\{j,k,h\}} g_i x_i \geq \sum_{i\in S} g_i - |S| - 1$. Also, S_j gives $x_j + x_h + \sum_{i\in S_j\setminus\{j,h\}} g_i x_i \geq \sum_{i\in S_j} g_i - |S_j| - 1$ and S_k gives $x_k + x_h + \sum_{i\in S_k\setminus\{k,h\}} g_i x_i \geq \sum_{i\in S_k} g_i - |S_k| - 1$. Thus, the inequality based on S is redundant because it can be obtained by the summation of the inequalities based on S_j and S_k.

Based on the above three cases, we only need to consider constraint (47) for S^2 segments. Thus, constraint (47) can be simplified, as follows:

$$\sum_{i\in S^2} x_i \geq 1 \quad \forall S^2 \subset V. \tag{50}$$

This allows us to rewrite CIP as

$$\begin{aligned} (\text{BIP}_{\text{cycle}}) \quad & \text{Minimize} \sum_{i\in V} b_i x_i \\ & \text{Subject to (49), (50),} \\ & x_i \in \{0,1\} \qquad \forall i \in V. \end{aligned} \tag{51}$$

We now show that the linear relaxation of $\text{BIP}_{\text{cycle}}$, which we refer to as LP_{cycle}, describes the polytope of the WTSS problem on cycles.

$$(\text{LP}_{\text{cycle}}) \quad z_{LP} = \text{Minimize} \sum_{i\in V} b_i x_i$$

$$\text{Subject to (49), (50),}$$

$$x_i \leq 1 \qquad \forall i \in V, \tag{52}$$

$$x_i \geq 0 \qquad \forall i \in V. \tag{53}$$

Theorem 5. *Inequalities* (49), (50), (52), and (53) *form an integral polytope for the WTSS problem on cycles.*

Proof. The constraint matrix is a 0-1 matrix. When $|G_2| \leq 1$, we only have one constraint (49), and we obtain a TU matrix.

From now on, we consider $|G_2| \geq 2$. When $|G_2|$ is an even number, inequalities (49) and (50) together form a TU matrix. Given a row submatrix, if it contains (49), then we put (49) in partition I_1 and put the rest in partition I_2. Then, in I_1, all variables appear once. In I_2, all of 1-node's variables appear at most once, and all of 2-node's variables appear at most twice. If the row submatrix does not satisfy (49), then the partition must make sure that if two constraints share a 2-node, these two constraints are not in the same partition. This can be done because $|G_2|$ is an even number. Thus, each variable appears in each partition at most once. Then, in both cases, we have $\left|\sum_{i\in I_1} a_{ij} - \sum_{i\in I_2} a_{ij}\right| \leq 1$ for all $j \in V$.

Thus, we have a TU matrix (see Theorem 2.7 in Nemhauser and Wolsey [24,p. 542] and Proposition 2.1 in Nemhauser and Wolsey [24,p. 540]).

This leaves us to consider $|G_2| \geq 3$, and $|G_2|$ is odd. In this case, the partition procedure for the TU matrix described above fails because the size of constraint (50) is an odd number. However, if we arbitrarily pick one S^2 segment, denoted by S^2_μ, and remove its corresponding constraint (50), the remaining constraint matrix is a TU matrix due to the identical argument for the case when $|G_2|$ is an even number. Then, associating a Lagrangian multiplier μ with the constraint (50) of S^2_μ, we obtain the Lagrangian relaxation of LP_{cycle}.

$$(\text{LR}^\mu_{\text{cycle}}) \quad z_{LR}(\mu) = \text{Minimize } \sum_{i \in V} b_i x_i + \mu \left(1 - \sum_{i \in S^2_\mu} x_i \right)$$

$$\begin{aligned} \text{Subject to } \quad & \sum_{i \in V} x_i \geq \left\lceil \frac{|G_2|}{2} \right\rceil, \\ & \sum_{i \in S^2} x_i \geq 1 \qquad \forall S^2 \subset \{V \backslash S^2_\mu\}, \\ & 0 \leq x_i \leq 1 \qquad \forall i \in V. \end{aligned}$$

As the constraint matrix is TU, the optimal solution of $\text{LR}^\mu_{\text{cycle}}$ is integral. Since $\text{LR}^\mu_{\text{cycle}}$ is a relaxation of LP_{cycle}, $z_{LR}(\mu) \leq z_{LP}$. Consider the Lagrangian dual $(\text{LD}^\mu_{\text{cycle}})$ $z_{LD} = \text{Maximize}_{\mu \geq 0}\ z_{LR}(\mu)$. We will demonstrate that for every objective coefficient $\mathbf{b} \in R^{|V|}$ of LP_{cycle}, there exists an integral optimal solution proving that LP_{cycle} has an integral polyhedron (see page 145 of [30]).

To visualize the argument in the proof consider the example shown in Figure 6; nodes 1 to 7 are 2-nodes. There might be 1-nodes in this instance, while they are not displayed in the picture. We have $|G_2| = 7$ and pick the S^2 segment between nodes 1 and 7 as S^2_μ. The constraints (50) of the S^2 segment between nodes 1 and 7 are relaxed, shown by a dashed line. All other constraints (50) are kept in the Lagrangian relaxation, displayed by solid lines. Now there are six constraints (50). We can ensure that if two constraints share a 2-node, these two constraints are not in the same partition.

Set $\mu = 0$ and solve $\text{LR}^0_{\text{cycle}}$. If the optimal solution, denoted by $\mathbf{x}^0_{LR}$, selects a node in S^2_μ, $\mathbf{x}^0_{LR}$ is an optimal solution of LP_{cycle} because $\mathbf{x}^0_{LR}$ is also feasible to LP_{cycle}, and $z_{LR}(0)$ is a lower bound of z_{LP}. Otherwise, we reduce the right-hand side of constraint (49) of $\text{LR}^0_{\text{cycle}}$ by 1 and solve the following problem:

$$(\text{LR}^{0-}_{\text{cycle}}) \quad z^-_{LR}(0) = \text{Minimize } \sum_{i \in V} b_i x_i$$

$$\begin{aligned} \text{Subject to } \quad & \sum_{i \in V} x_i \geq \left\lceil \frac{|G_2|}{2} \right\rceil - 1, \\ & \sum_{i \in S^2} x_i \geq 1 \qquad \forall S^2 \subset \{V \backslash S^2_\mu\}, \\ & 0 \leq x_i \leq 1 \qquad \forall i \in V. \end{aligned}$$

Notice that the optimal solution of $\text{LR}^{0-}_{\text{cycle}}$, denoted by $\mathbf{x}^{0-}_{LR}$, is also integral. It does not select any node in S^2_μ, given that $\mathbf{x}^0_{LR}$ does not have any node in S^2_μ. Let $t = \min\{b_i : i \in S^2_\mu\}$, $\tilde{i} = \arg\ \min\{b_i : i \in S^2_\mu\}$, and $\theta = z_{LR}(0) - z^-_{LR}(0)$. Clearly, $t \geq \theta$. Otherwise, $\mathbf{x}^{0-}_{LR}$ and node $\tilde{i}$ together give a lower cost solution than $\mathbf{x}^0_{LR}$: a contradiction. Furthermore, the nodes selected in $\mathbf{x}^{0-}_{LR}$ and node $\tilde{i}$ form a feasible solution to LP_{cycle} because node $\tilde{i}$ is in S^2_μ, and the number of selected nodes is at least $\left\lceil \frac{|G_2|}{2} \right\rceil$. Its objective value is $z^-_{LR}(0) + t$.

Next, set $\mu = t - \theta$ and solve $\text{LR}^{t-\theta}_{\text{cycle}}$. There are two cases. One is that the optimal solution of $\text{LR}^{t-\theta}_{\text{cycle}}$, denoted by $\mathbf{x}^{t-\theta}_{LR}$, does not contain any node in S^2_μ. In this case, we must have $\mathbf{x}^{t-\theta}_{LR} = \mathbf{x}^0_{LR}$ because the objective function is $\sum_{i \in V \backslash S^2_\mu} b_i x_i + \sum_{i \in S^2_\mu} (b_i - t + \theta) x_i + t - \theta = \sum_{i \in V \backslash S^2_\mu} b_i x_i + t - \theta$, and we assume that $\mathbf{x}^0_{LR}$ does not contain any node in S^2_μ. Thus, $z_{LR}(t - \theta) = \sum_{i \in \mathbf{x}^0_{LR}} b_i + t - \theta = z_{LR}(0) + t - \theta = z^-_{LR}(0) + t$. The other case is that there exists an optimal solution $\mathbf{x}^{t-\theta}_{LR}$ that contains exactly one node in S^2_μ. Observe that at least $\left\lceil \frac{|G_2|}{2} \right\rceil - 1$ nodes in $V \backslash S^2_\mu$ are selected in a feasible solution to satisfy the $|G_2| - 1$ constraints (50) in $\text{LR}^\mu_{\text{cycle}}$ (as shown in Figure 6; at least 3 nodes from nodes 2, 3, 4, 5, and 6 must be selected). Thus, if multiple nodes in S^2_μ are selected in $\mathbf{x}^{t-\theta}_{LR}$, deselecting all of them and selecting node $\tilde{i}$ gives a solution with no worse objective value (since $b_i - t + \theta \geq 0$ for all $i \in S^2_\mu$). The nodes selected (which must be at least $\left\lceil \frac{|G_2|}{2} \right\rceil - 1$ in number) in $\mathbf{x}^{t-\theta}_{LR}$ on $V \backslash S^2_\mu$ satisfy the $|G_2| - 1$ constraints (50) (and are of minimum cost), meaning that they are identical to $\mathbf{x}^{0-}_{LR}$. Therefore, $\mathbf{x}^{t-\theta}_{LR} = \mathbf{x}^{0-}_{LR} \cup \tilde{i}$. In this case, $z_{LR}(t-\theta) = \sum_{i \in V} b_i x_i + (t - \theta)\left(1 - \sum_{i \in S^2_\mu} x_i\right) = \sum_{i \in \mathbf{x}^{0-}_{LR}} b_i + t = z^-_{LR}(0) + t$. Consequently, in both cases, $z_{LR}(t - \theta)$ (which is a lower bound on z_{LP}) is equal to the objective value of the integer

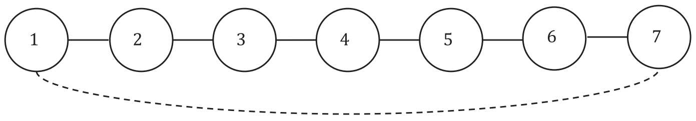


FIGURE 6 Example for the case when $|G_2| \geq 3$ and $|G_2|$ is odd

solution consisting of the nodes selected in $\mathbf{x}^{0-}_{LR}$ and node $\tilde{i}$, which is feasible to LP_{cycle}. Thus, the nodes selected in $\mathbf{x}^{0-}_{LR}$ and node $\tilde{i}$ form an optimal solution to LP_{cycle}. This completes the proof. ■

The number of sets S^2 is equal to $|G_2|$, the number of 2-nodes. Thus, the number of constraints is $1 + |G_2| + 2|V| \leq 3|V| + 1 = O(|V|)$. Hence, we have a tight and compact formulation, which gives the complete description of the polytope of the WTSS problem on cycles.

5 | THE WTSS PROBLEM ON ARBITRARY GRAPHS

Given our motivations for studying these special cases, naturally, the next step is to see if we can apply the ideas here to develop strong formulations for the WTSS problem on arbitrary graphs. In a parallel paper [26], we observe that (i) the tight and compact extended formulation for trees is also a tight and compact extended formulation for directed acyclic graphs (DAG), and (ii) the influence propagation network in any arbitrary graph is acyclic. Thus, the key idea in applying the results of this paper to arbitrary graphs is to ensure that the influence propagation network is acyclic. This is achieved by adding an exponentially sized set of inequalities that enforce this (acyclic influence propagation network) condition.

In this model for the WTSS problem on arbitrary graphs, we introduce a new variable set, h_{ij} and h_{ji}, for all $\{i, j\}$ in E (these are defined on the original graph). If $h_{ij} = 1$, it means that node i gives influence to node j. Otherwise, $h_{ij} = 0$. A new constraint set, which ensures that the directed graph formed by $\mathbf{h}$ and denoted by $G(\mathbf{h})$ is a DAG, is needed. With this in hand, we have the following formulation, which we refer to as BIP:

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

$$\text{Subject to } (13), (14), (15), (16), (17) \tag{54}$$

$$h_{ij} + h_{ji} = 1 \qquad \forall \{i,j\} \in E, \tag{55}$$

$$\sum_{(i,j)\in C} h_{ij} \leq |C| - 1 \qquad \forall \text{dicycles } C \text{ in } G, \tag{56}$$

$$h_{ij} \leq y_{id}, h_{ji} \leq y_{jd} \quad \forall \{i,j\} \in E, \ \{i,d\}, \ \{j,d\} \in E^t, \tag{57}$$

$$h_{ij}, h_{ji} \in \{0,1\} \qquad \forall \{i,j\} \in E. \tag{58}$$

We have dropped constraint (12) because it is implied by constraint sets (55) and (57). The objective function is the same as before. Constraint set (55) stipulates that influence must go in one direction between two nodes in the original graph. Constraint set (56), called k-dicycle inequalities, ensures that the influence cannot propagate around a directed cycle (dicycle). Here, k refers to the cardinality $|C|$ of the dicycle. For any dicycle C, the constraint says that at most $|C| - 1$ of the h_{ij} variables can be 1. Because the $\mathbf{h}$ variables are defined on the original graph G, and the y variables are on the transformed graph G_t, we need constraint set (57) to serve as the linking constraints, which synchronize the influence propagation process between these two graphs (notice that when $h_{ij} = 1$, $y_{id} = 1$, and when $h_{ji} = 1$, $y_{jd} = 1$).

Based on BIP, in Raghavan and Zhang [26], we design and implement a branch-and-cut approach for the WTSS problem on arbitrary graphs. The branch-and-cut approach includes a heuristic for initial solutions, ways to speed up the separation procedure for the exponential set of inequalities, a specialized branching rule, and a primal heuristic to quickly identify feasible solutions in the search process; all of which improve its performance. On a test set of 180 real-world graph instances (with up to approximately 155 000 nodes and 327 000 edges), the branch-and-cut approach finds solutions that are on average 0.90% from optimality, and solves 60 out of the 180 instances to optimality. On the other hand, the best heuristic solutions in the literature generated are on average 5.46 times greater than the solutions generated by the branch-and-cut approach.

Recently, Fischetti et al. [14] presented a set covering formulation for the LCIP. We describe how this set covering formulation can be adapted to the WTSS problem. For a node i in V, let $U \subseteq n(i)$ denote a set of (previously) active neighbors of node i during the influence propagation process, and let w_i^U denote the cost for this active set U to activate node i. In other words, if $|U| \geq g_i$, then $w_i^U = 0$, and $w_i^U = b_i$ otherwise. Given a node i and a set $U \subseteq n(i)$, the set U is referred to as a minimal influencing set for node i, if node i cannot be activated by a proper subset of U at the same cost. For each node i in V, let $\Xi_i \subseteq n(i)$ denote the set of all minimal influencing subsets. It is sufficient to focus on minimal influencing sets. For a node i in V, the minimal influencing set U is thus either a combination of exactly g_i neighbors of node i with $w_i^U = 0$ or an empty set with $w_i^U = b_i$. The model needs two sets of variables. For each node i in V and each minimal influence set U in Ξ_i, ξ_i^U is 1 if the minimal influencing set U is used to activate node i, and 0 otherwise.

For each edge $\{i, j\}$ in E, two arc variables z_{ij} and z_{ji} are created. If node i sends influence to node j, z_{ij} is 1, and 0 otherwise. The model "COV" is as follows:

$$\text{(COV)} \quad \text{Minimize} \quad \sum_{i\in V}\sum_{U\in\Xi_i} w_i^U \xi_i^U \tag{59}$$

$$\text{Subject to} \quad \sum_{U\in\Xi_i} \xi_i^U = 1 \qquad \forall i \in V, \tag{60}$$

$$\sum_{U\in\Xi_j : i\in U} \xi_j^U = z_{ij}, \quad \sum_{U\in\Xi_i : j\in U} \xi_i^U = z_{ji} \qquad \forall \{i,j\} \in E, \tag{61}$$

$$\sum_{(i,j)\in C} z_{ij} \leq |C| - 1 \qquad \forall \text{dicycles } C \text{ in } G, \tag{62}$$

$$z_{ij}, z_{ji} \in \{0, 1\} \qquad \forall \{i,j\} \in E, \tag{63}$$

$$\xi_i^U \in \{0, 1\} \qquad \forall i \in V, \;\; U \in \Xi_i. \tag{64}$$

The objective function (59) minimizes the total weight. Notice that the variable $\xi_i^\emptyset$ in COV corresponds to the node selection variable x_i in BIP. COV creates an extra $\binom{\deg(i)}{g_i}$ variables for each node corresponding to the different possible (minimal) neighbor combinations that could activate it. Constraint (60) makes sure that each node i in V is either in the target set (when $\xi_i^\emptyset = 1$) or has a set of exactly g_i active neighbors to activate it. Constraint (61) states that for an edge $\{i, j\}$, node i sends influence to node j only when node i is in the influence set that activates node j. Constraint (62) ensures that there are no directed cycles formed by $\mathbf{z}$ variables. COV has exponentially many variables and exponentially many constraints. Thus, it will require the development and application of column and cut generation techniques to apply COV as an exact approach to large graphs. As future research, it would be interesting to see how COV performs on the WTSS problem and compare it to the branch-and-cut approach based on BIP in [26].

6 | CONCLUSIONS

In this paper, we studied the weighted version of the well-studied TSS problem on trees and cycles. On trees, our contributions are 3-fold. First, we show that the WTSS problem can be solved polynomially (via DP) when the underlying graph is a tree. Our DP algorithm generalizes Chen's [8] algorithm and runs in $O(|V|)$ time. Second, we present a tight and compact extended formulation whose LP relaxation provides integral solutions for the WTSS problem on trees. Lastly, we also project this formulation onto the space of the natural node variables, and thus find the polytope of the WTSS problem on trees. On cycles, our contributions are two-fold. First, a linear-time algorithm is presented. The algorithm is based on an idea that breaks a cycle into trees and makes use of the linear-time algorithm for the WTSS problem on trees. Furthermore, we derive the polytope of the WTSS problem on cycles.

One future direction is to consider more complex graphs for polyhedral studies. Given that we know the polytopes for the WTSS problem on cycles and trees, it would be natural to consider cacti graphs, where any two simple cycles have at most one node in common. Baïou and Barahona [3] showed how to obtain the polytope of the minimum weight dominating set problem (MWDSP) on cacti graphs by applying a composition procedure over the polytopes of the MWDSP on trees and cycles. However, it is not clear how their idea can be applied to the WTSS problem.

In the WTSS problem, the diffusion process continues until it is complete (i.e., we are allowed as many steps/time periods as needed for the influence to propagate through the entire network). However, if we restrict the number of steps/time periods to one, the entire network must be influenced after one time step. We obtain a weighted version of a problem called the positive influence dominating set (PIDS) problem proposed by Wang in [29]. This models the situation in which the marketer would like the diffusion process to take place rapidly. In the PIDS problem, either a node is selected at a weight of b_i, or it requires g_i of its neighbors to be selected (notice that when both b_i and g_i are 1, we obtain the dominating set problem). In a related paper [25], we study the PIDS problem.

One generalization of the WTSS problem has unequal influence from neighbors. Unfortunately, this problem is somewhat harder to solve. Raghavan and Zhang [26] show that the WTSS problem with unequal influence is even NP-complete on stars (i.e., a tree where all nodes are leaves other than a single central node). Deriving strong formulations for this and other closely related IMPs is an avenue of our current and future research.

ORCID

S. Raghavan https://orcid.org/0000-0002-9656-5596
Rui Zhang https://orcid.org/0000-0002-4029-6585

REFERENCES

[1] E. Ackerman, O. Ben-Zwi, and G. Wolfovitz, *Combinatorial model and bounds for target set selection*, Theoret. Comput. Sci. **411** (2010), 4017–4022.
[2] A. B. Adcock, B. D. Sullivan, and M. W. Mahoney, *Tree-like structure in large social and information networks*, 13th International Conference on Data Mining (ICDM), IEEE, 2013, pp. 1–10.
[3] M. Baïou and F. Barahona, *On the integrality of some facility location polytopes*, SIAM J. Discrete Math. **23** (2009), 665–679.
[4] E. Balas and W. Pulleyblank, *The perfectly matchable subgraph polytope of a bipartite graph*, Networks **13** (1983), 495–516.
[5] C. Barnhart, E.L. Johnson, G.L. Nemhauser, G. Sigismondi, and P. Vance, *Formulating a mixed integer programming problem to improve solvability*, Oper. Res. **41** (1993), 1013–1019.
[6] O. Ben-Zwi, D. Hermelin, D. Lokshtanov, and I. Newman, *Treewidth governs the complexity of target set selection*, Discrete Optim. **8** (2011), 87–96.
[7] C. Blum, *Revisiting dynamic programming for finding optimal subtrees in trees*, Eur. J. Oper. Res. **177** (2007), 102–115.
[8] N. Chen, *On the approximability of influence in social networks*, SIAM J. Discrete Math. **23** (2009), 1400–1415.
[9] W. Chen, C. Castillo, and L.V. Lakshmanan, *Information and Influence Propagation in Social Networks*, Morgan & Claypool Publishers, San Rafael, CA, 2013.
[10] C.-Y. Chiang, L.-H. Huang, B.-J. Li, J. Wu, and H.-G. Yeh, *Some results on the target set selection problem*, J. Combin. Optim. **25** (2013), 702–715.
[11] M. Conforti, G. Cornuéjols, and G. Zambelli, *Integer Programming*, Springer, New York, 2014.
[12] G. Cordasco, L. Gargano, A.A. Rescigno, and U. Vaccaro, "*Optimizing spread of influence in social networks via partial incentives*," *Structural Information and Communication Complexity*, C. Scheideler (ed.), Springer, Cham, 2015, pp. 119–134.
[13] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, *Introduction to Algorithms*, MIT Press, Cambridge, MA, 2009.
[14] M. Fischetti, M. Kahr, M. Leitner, M. Monaci, and M. Ruthmair, *Least cost influence propagation in (social) networks*, Math. Program. Ser. B **170** (2018), 293–325.
[15] M.X. Goemans, *The Steiner tree polytope and related polyhedra*, Math. Program. **63** (1994), 157–182.
[16] M. Granovetter, *Threshold models of collective behavior*, Am. J. Sociol. **83** (1978), 1420–1443.
[17] D. Günneç, S. Raghavan, and R. Zhang, *Least-cost influence maximization on social networks*, INFORMS J. Comput. **32** (2020), 289–302.
[18] D. Günneç, S. Raghavan, and R. Zhang, *A branch-and-cut approach for the least cost influence problem on social networks*, Networks **76** (2020), 84–105.
[19] D. Kempe, J. Kleinberg, and É. Tardos, *Maximizing the spread of influence through a social network*. Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2003, pp. 137–146.
[20] H. Kwak, C. Lee, H. Park, and S. Moon, *What is Twitter, a social network or a news media?* Proceedings of the 19th International Conference on the World Wide Web, ACM, 2010, pp. 591–600.
[21] Y. Li, J. Fan, Y. Wang, and K.-L. Tan, *Influence maximization on social graphs: A survey*, IEEE Trans. Knowl. Data Eng. **30** (2018), 1852–1872.
[22] T. L. Magnanti and L. A. Wolsey, "*Optimal trees*," *Handbooks in Operations Research and Management Science*, vol. 7, chapter 9, Elsevier, Amsterdam, 1995, pp. 503–615.
[23] G. Nannicini, G. Sartor, E. Traversi, and R. Wolfler-Calvo, *An exact algorithm for robust influence maximization*, International Conference on Integer Programming and Combinatorial Optimization, Springer, 2019, pp. 313–326.
[24] G. Nemhauser and L. Wolsey, *Integer and Combinatorial Optimization*, Wiley, New York, 1988.
[25] S. Raghavan and R. Zhang, *Rapid influence maximization on social networks: The positive influence dominating set problem*, Working paper, University of Maryland, 2017.
[26] S. Raghavan and R. Zhang, *A branch-and-cut approach for the weighted target set selection problem on social networks*, INFORMS J. Optim. **1** (2019), 304–322.
[27] P. Shakarian, S. Eyre, and D. Paulo, *A scalable heuristic for viral marketing under the tipping model*, Soc. Netw. Anal. Min. **3** (2013), 1225–1248.
[28] G. Spencer, S. Carattini, and R.B. Howarth, *Short-term interventions for long-term change: Spreading stable green norms in networks*, Rev. Behav. Econ. **6** (2019), 53–93.
[29] F. Wang, H. Du, E. Camacho, K. Xu, W. Lee, Y. Shi, and S. Shan, *On positive influence dominating sets in social networks*, Theoret. Comput. Sci. **412** (2011), 265–269.
[30] L.A. Wolsey, *Integer Programming*, Wiley, New York, 1998.
[31] H.-H. Wu and S. Küçükyavuz, *A two-stage stochastic programming approach for influence maximization in social networks*, Comput. Optim. Appl. **69** (2018), 563–595.

How to cite this article: Raghavan S, Zhang R. Weighted target set selection on trees and cycles. *Networks.* 2021;77:587–609. https://doi.org/10.1002/net.21972

APPENDIX A: PROOF OF THEOREM 1

Proof. Let L be the set of leaf nodes in G^t, and let r be the root of the tree, as determined by Algorithm 1 (the central node in the last star in Algorithm 1). We start from the leaf nodes and then follow the sequence of stars that Algorithm

1 considers (the proof works with any sequence, but for expositional reasons, it is best to consider the same sequence of stars) and construct a dual feasible solution to DLP$_{\text{tree}}$ that satisfies the CS conditions.

For the purposes of this proof, we construct a specific form of the extended solution (**x**, **y**) from the (integral) solution **x** obtained by Algorithm 1. Recall that as we backtrack for each central node of a star encountered, we know whether or not it obtains external influence in the final solution. Thus, even prior to inserting the dummy nodes, we simply direct the arc from a central node c's parent to itself in the case of external influence, and the arc to a central node c's parent in the case of no influence. For any leaf node that is picked, we direct the arc from the leaf to its parent. For leaf nodes that are not picked, we direct the arc from its parent to itself. It is easy to observe (before the dummy nodes are inserted and the solution is extended) that in a solution obtained by Algorithm 1, a node i that is not selected has by design exactly g_i incoming arcs. We now insert the dummy nodes on each edge. If the influence on the edge propagates from i to j, then with the dummy node d inserted in the middle of the edge, the influence propagates from i to d to j, with the y values set accordingly. Now, to satisfy constraint (13) for any node i that is selected and has external influence, we simply reverse the arc from the node i to the dummy node that is inserted between itself and its parent. An important (and meaningful) consequence of this construction is that if a dummy node d in this extended solution has two incoming arcs (from nodes i and j, with node i being node j's child in Algorithm 1), then node i must have been picked in the solution. Notice that this results in always satisfying constraint (15) as a binding inequality, implying that the CS condition (29) is always satisfied between the primal solution (**x**, **y**) and every dual feasible solution. In other words, we can focus on the CS conditions (30)-(34) for the rest of the proof.

We start with the leaf nodes and show how to set the dual variables associated with them to satisfy the CS conditions. Specifically, for each leaf node l in L and the dummy node d adjacent to node l, we set $v_l = b_l$, $u_d = b_l$, $w_{ld} = 0$, and $z_{ld} = -b_l$. Figure A1 illustrates this construction. With these choices, constraints (23)-(25) associated with l, d, and edge $\{l, d\}$ are satisfied and always binding. Consequently, regardless of the primal solution, conditions (32)-(34) are satisfied for nodes $l \in L$ and the edge $\{i, d\}$ adjacent to it. Condition (31) is satisfied because w_{ld} is 0. For (30), if $1 - y_{ld} - y_{id}$ is not equal to 0, where node i is the other node adjacent to node d, then node l is selected when it can be activated by an incoming arc, which never happens in Algorithm 1 (otherwise, this would imply that stars solely containing the leaf nodes of the original tree select both a leaf node and its parent node at the same time in Algorithm 1). Hence, we have $1 - y_{ld} - y_{id} = 0$, and (30) is satisfied. Therefore, the CS conditions are satisfied for this part of the dual solution.

Now we consider the remaining dual variables. For the center of a star i that Algorithm 1 considers, we iteratively show how to set v_i, u_p for the dummy node p, which is node i's parent, w_{id} for all $d \in n(i)$, and z_{id} for all $\{i, d\} \in E^t$, while ensuring that all of the associated CS conditions are satisfied. Start with the first star that Algorithm 1 considers. The center of this star i must have child dummy nodes that have their u variables set. Relabel these child dummy nodes in ascending order based on their u values. Let $S_{g_i - 1}$ be the first $(g_i - 1)$ child dummy nodes and let S_{g_i} be the first g_i child dummy nodes. If node i is not the root node r, let p be the parent dummy node of node i, and set the value u_p in the following way: we have three cases corresponding identically to the three cases in how a star is handled and contracted in Algorithm 1. If $b_i < \sum_{d \in S_{g_i - 1}} u_d$, set $u_p = 0$. If $b_i > \sum_{d \in S_{g_i}} u_d$, set $u_p = u_{g_i}$. Otherwise, set $u_p = b_i - \sum_{d \in S_{g_i - 1}} u_d$. We refer to these as Cases 1, 2, and 3, respectively. If g_i is equal to $deg(i)$, then there is no S_{g_i}, and we only have Cases 1 and 3. By setting u_p in the aforementioned way for a dummy node p, which is the parent of the central node of a star encountered by Algorithm 1, (30) is satisfied. To show this, we only need to worry about the value of u_p when the dummy node p has two incoming arcs. Let i and j be the two adjacent nodes to this dummy node p (with i being the central node of the current star and j being its parent in Algorithm 1). As we have argued above, node i must have been selected by Algorithm 1. However, then we have $u_p = 0$ because node p's value would be set based on Case 1, thus satisfying (30). Figure A2 illustrates this construction. Now we focus on selecting feasible values for the remaining dual variables to satisfy conditions (31)-(34).

Sort the dummy nodes adjacent to node i based on their u values in ascending order. Let F_{g_i} be the first g_i dummy nodes. We consider two situations. In situation 1, node i is selected in the solution. Thus, $x_i = 1$. Then, we set v_i as $u_{(g_i)}$, as the g_ith smallest value in $\{u_d : d \in n(i)\}$. Let $W = g_i v_i - b_i$. Because node i is selected in the solution, b_i is smaller than $\sum_{d \in F_{g_i}} u_d$, and W has a positive value because $u_{(g_i)}$ is bigger than $\frac{b_i}{g_i}$. We have $y_{id} = 1$ for all $d \in n(i)$ due to (13) and $y_{di} = 0$ for all $d \in n(i)$ due to (14). Thus, (23) and (25) must be binding based on condition (32) and (34). Meanwhile, we need to satisfy (24). We set $z_{id} = -v_i$ and $w_{id} = v_i - u_d$ for all $d \in F_{g_i}$. Then, $z_{id} = -u_d$ and $w_{id} = 0$ for all $d \in \{n(i) \backslash F_{g_i}\}$. Lastly, let $w = W - \sum_{d \in F_{g_i}} w_{id}$. Pick any dummy node $\overline{d}$ in $n(i) \backslash F_{g_i}$ and set $w_{i\overline{d}} = w$ and $z_{i\overline{d}} = -u_{\overline{d}} - w$. Here, we reset the value of $w_{i\overline{d}}$ to distribute the excess amount of W in order to satisfy $\sum_{d \in n(i)} w_{id} = W$ and ensure that (25) is binding. For any $d \in n(i)$, (23) is binding. Furthermore, (24) is binding for any $d \in F_{g_i}$ and is satisfied as an inequality for any $d \in n(i) \backslash F_{g_i}$. Thus, condition (31) is satisfied because $x_i = y_{id} = 1$, and condition (33) is satisfied because $y_{di} = 0$. We have (23) and (25) as binding; thus, conditions (32) and (34) are satisfied. Therefore, the CS conditions are satisfied for this part of the dual solution. Figure A3 illustrates this construction.

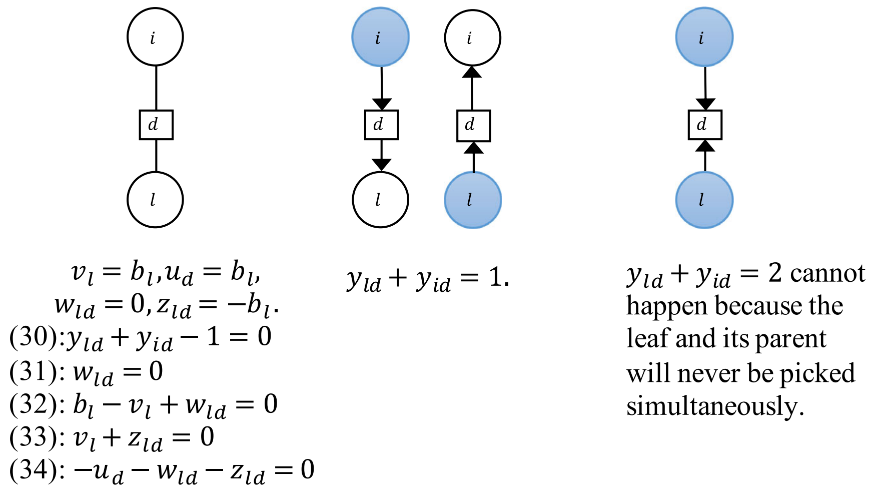

i
d
l
$v_l = b_l, u_d = b_l,$
$w_{ld} = 0, z_{ld} = -b_l.$
(30):$y_{ld} + y_{id} - 1 = 0$
(31): $w_{ld} = 0$
(32): $b_l - v_l + w_{ld} = 0$
(33): $v_l + z_{ld} = 0$
(34): $-u_d - w_{ld} - z_{ld} = 0$
$y_{ld} + y_{id} = 1.$
$y_{ld} + y_{id} = 2$ cannot happen because the leaf and its parent will never be picked simultaneously.

FIGURE A1 Leaf nodes and complementary slackness conditions [Color figure can be viewed at wileyonlinelibrary.com]

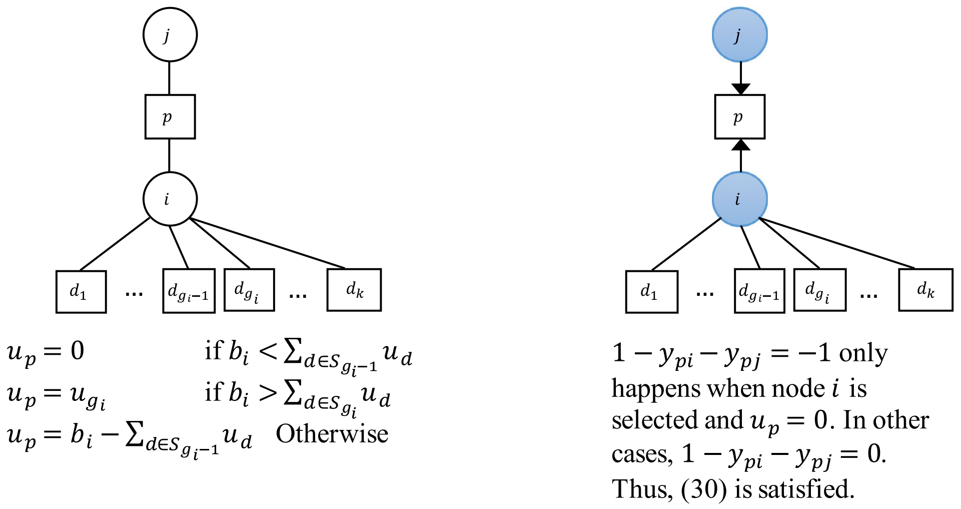
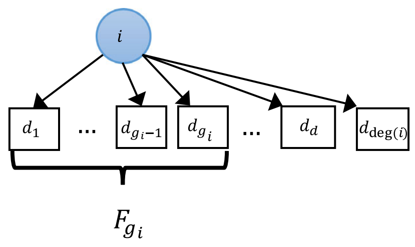

j
p
i
d_1 ... d_{g_i-1} d_{g_i} ... d_k
$u_p = 0$ if $b_i < \sum_{d \in S_{g_i-1}} u_d$
$u_p = u_{g_i}$ if $b_i > \sum_{d \in S_{g_i}} u_d$
$u_p = b_i - \sum_{d \in S_{g_i-1}} u_d$ Otherwise
$1 - y_{pi} - y_{pj} = -1$ only happens when node i is selected and $u_p = 0$. In other cases, $1 - y_{pi} - y_{pj} = 0$. Thus, (30) is satisfied.

FIGURE A2 Setting u_p and condition (30) [Color figure can be viewed at wileyonlinelibrary.com]

In situation 2, node i is not selected, but it has exactly g_i incoming arcs. We have $x_i = 0$, and condition (32) is satisfied. Let I be the set of dummy nodes that sends influence to node i, that is, $y_{di} = 1$ for all d in I. We set $v_i = u_{\underline{d}}$, where $\underline{d} = \arg\max\{u_d : d \in I\}$. Then, for all $d \in I$, we also have $y_{di} = 1$ and $y_{id} = 0$ due to (14). Thus, (24) must be binding due to condition (33). First, we set $w_{id} = 0$ for all $d \in n(i)$ and $z_{id} = -v_i$ for all $d \in I$. Let $W = g_i v_i - b_i$. If W has a positive value, then (25) is violated. Thus, following the ascending ordering of u_d for all d in I, we set $w_{id} = \min\{v_i - u_d, W - \sum_{j \in I} w_{ij}\}$ to satisfy (25). Because node i is not selected, it implies that $b_i \geq \sum_{d \in I} u_d$. Thus, $v_i g_i - b_i \leq v_i g_i - \sum_{d \in I} u_d$, given that $|I| = g_i$. Therefore, we can distribute W in this way (i.e., $\sum_{d \in I} w_{id} = W$) and ensure that condition (23) holds. Then, for all $d \in I$, (23) is satisfied, and (24) is binding. Condition (31) is satisfied because $x_i = y_{id} = 0$; condition (33) is satisfied because $z_{id} = -v_i$; and condition (34) is satisfied because $y_{id} = 0$. Also, (25) is satisfied. Next, for all $d \in n(i) \backslash I$, $y_{id} = 1$ and $y_{di} = 0$ due to (14) and (15). Hence, (23) must be binding due to condition (34). Then, we set $z_{id} = -u_d$ for all $d \in n(i) \backslash I$. Thus, for all $d \in n(i) \backslash I$, (23) is binding, and (24) is satisfied because $u_d \geq u_{(g_i)}$. Condition (31) is satisfied because $w_{id} = 0$; condition (33) is satisfied because $y_{di} = 0$; and condition (34) is satisfied because $z_{id} = -u_d$ and $w_{id} = 0$. Lastly, condition (32) is satisfied because $x_i = 0$ and (25) is satisfied. Thus, the CS conditions are satisfied for this part of the dual solution. Figure A4 illustrates this construction.

Repeating this procedure for each star in the order that they are encountered in Algorithm 1 provides a dual feasible solution to DLP$_{\text{tree}}$, which satisfies the CS conditions associated with our primal solution obtained from Algorithm 1. ■

We illustrate the procedure for constructing the dual solution with the example and primal solution in Figure 3B. Starting at the leaf nodes (5,6,7,8,9,10,11) and dummy nodes adjacent to them (d, e, f, g, h, i, j), we have $v_5 = 5$, $v_6 = 6$, $v_7 = 7$, $v_8 = 8$, $v_9 = 9$, $v_{10} = 10$, $v_{11} = 11$ and $u_d = 5$, $u_e = 6$, $u_f = 7$, $u_g = 8$, $u_h = 9$, $u_i = 10$, $u_j = 11$. For those edges between them, we



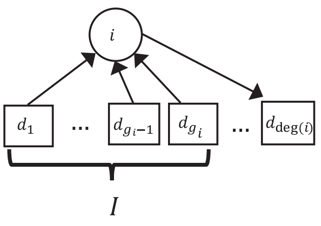
(31): $x_i = 1, y_{id} = 1$, so, $x_i - y_{id} = 0$

(32): $\sum_{d\in a(i)} w_{id} = \sum_{d\in F_{g_i}} w_{id} + w_{i\bar{d}} = W$
$= g_i v_i - b_i$, so, $b_i - g_i v_i + \sum_{d\in n(i)} w_{id} = 0$

(33): $y_{di} = 0, v_i = u_{d_{(g_i)}}$
$z_{id} = -v_i$, so, $z_{id} + v_i = 0 \ \forall d \in F_{g_i}$
$z_{id} = -u_d - w$, so, $z_{id} + v_i < 0$ for $d = \bar{d}$
$z_{id} = -u_d < -u_{d_{(g_i)}}$, so, $z_{id} + v_i < 0$ otherwise

(34): $w_{id} = v_i - u_d$, so, $w_{id} + u_d + z_{id} = 0 \ \forall d \in F_{g_i}$
$w_{id} = w$, so, $w_{id} + u_d + z_{id} = 0$ for $d = \bar{d}$
$w_{id} = 0$, so, $w_{id} + u_d + z_{id} = 0$ otherwise

FIGURE A3 Situation 1: nonleaf node *i* is selected in the solution [Color figure can be viewed at wileyonlinelibrary.com]



(31): $x_i = 0$, then, $w_{id} \geq 0, y_{id} = 0 \ \forall d \in I$
$w_{id} = 0, y_{id} = 1 \ \forall d \in n(i)\backslash I$

(32): $x_i = 0$ and $b_i - g_i v_i + \sum_{d\in n(i)} w_{id} \leq 0$.

(33): $v_i = u_{\hat{d}}$ where $\hat{d} = argmax\{u_d : d \in I\}$
$y_{di} = 1, z_{id} = -v_i$, so, $z_{id} + v_i = 0 \ \forall d \in I$
$y_{di} = 0, z_{id} = -u_d$, so, $z_{id} + v_i < 0$ otherwise

(34): $y_{id} = 0, w_{id} \leq -z_{id} - u_d$, so, $w_{id} + u_d + z_{id} \leq 0 \forall d \in I$
$y_{id} = 1$, so, $w_{id} + u_d + z_{id} = 0$ otherwise

FIGURE A4 Situation 2: nonleaf node *i* is not selected in the solution

have $w_{5d} = 0$, $w_{6e} = 0$, $w_{7f} = 0$, $w_{8g} = 0$, $w_{9h} = 0$, $w_{10i} = 0$, $w_{11j} = 0$, and $z_{5d} = -5$, $z_{6e} = -6 z_{7f} = -7$, $z_{8g} = -8$, $z_{9h} = -9$, $z_{10i} = -10$, $z_{11j} = -11$.

Then, we have nodes 1, 2, and 3 ready for the next step. We start with node 1. Node 1's parent dummy node is node *a* and $u_a = 0$ because node 1 is a Case 1 node. Node 1 is in situation 1. We have $v_1 = 5$ because $g_1 = 2$ and the second smallest value among {0,5,6} is 5. $W = 5 \times 2 - 4 = 6$. Hence, $z_{1a} = -5$, $w_{1a} = 5$ and $z_{1d} = -5$, $w_{1a} = 0$ for edges (1, *a*) and (1, *d*). Then, $w_{1e} = 6 - 5 = 1$ and $z_{1e} = -(6+1) = -7$.

Next, node 2's parent dummy node is node *b* and $u_b = 9$ because node 2 is a Case 2 node. Node 2 is in situation 2. We have $v_2 = 9$ because it has the incoming arcs (*f*, 2), (*g*, 2), and (*h*, 2), and the biggest value among {7,8,9} is 9. Hence, $z_{2f} = -9$, $z_{2g} = -9$, $z_{2h} = -9$ and $w_{2f} = 0$, $w_{2g} = 0$, $w_{2h} = 0$. $W = 27 - 30 = -3$. Then, $z_{2b} = -9$ and $w_{2b} = 0$.

We now look at node 3. Node 3's parent dummy node is node *c* and $u_c = 10$ because node 3 is a Case 3 node. Node 3 is in situation 2. Thus, $v_3 = 10$ because it has the incoming arcs (*c*, 3) and (*i*, 2), and the biggest value among {10, 10} is 10. Thus, $z_{3c} = -10$, $z_{3i} = -10$ and $w_{3c} = 0$, $w_{3i} = 0$. $W = 20 - 20 = 0$. Then, $z_{2b} = -11$ and $w_{2b} = 0$.

Now, node 4 is ready. It is in situation 2. Thus, $v_4 = 9$ because it has the incoming arcs (*a*, 4) and (*b*, 4), and the biggest value among {0, 9} is 9. Then, $z_{4a} = -9$, $z_{4b} = -9$ and $w_{3c} = 0$, $w_{3i} = 0$. $W = 18 - 15 = 3$. Start from node *a*. $w_{4a} = \min\{9 - 0, 3\} = 3$. Also, $z_{4c} = -10$ and $w_{4c} = 0$. The sum of $2v_4$, z_{3c}, z_{3i}, and z_{3j} is -10. The dual objective value is 38, which is exactly equal to the primal objective value.

APPENDIX B: PROOF OF THEOREM 3

Proof. For ease of exposition, we use Figure B1 to illustrate the notation in Case 3. Figure B1A shows the original graph *G*, and Figure B1B shows the transformed graph G^t. Here, w_1, w_2, w_3, and w_4 have positive values of α. Thus, *S* contains nodes 1, 2, 3, and 4. Notice that *S* is connected in the original graph *G*. The induced subgraph $G^t(S)$ is shown in Figure B1C. Based on $G^t(S)$, we obtain $D(S) = \{a, b, c\}$ and $a(S)\backslash D(S) = \{d, e, f, g, h, i, j\}$.

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

Case 2 is similar to Case 1.

Let $\mathbf{r} \in \mathcal{W}$ be of the form Case 3, and assume that $\mathbf{r} = \frac{1}{2}(\mathbf{r}^1 + \mathbf{r}^2)$ for some $\mathbf{r}^1, \mathbf{r}^2 \in \mathcal{W}$. Thus, for any component of $\mathbf{r}$ with the value of 0, their corresponding components in $\mathbf{r}^1$ and $\mathbf{r}^2$ are also 0. Given *i* and *d*, let p_{id}^k represent the positive component of u_d^k or v_{id}^k, $k = 1, 2$. Then, we have $w_i^1 + w_i^2 = 2\alpha$ and $p_{id}^1 + p_{id}^2 = 2\alpha$ for all $d \in n(i)$ and for all $i \in S$. Then, if there is a pair d_1 and d_2, we have $p_{id_1}^1 > p_{id_2}^1$ if and only if $p_{id_1}^2 < p_{id_2}^2$. However, constraint (41) stipulates that $w_i^k \leq p_{id}^k$,

$k = 1, 2$. Hence, $p^k_{id_1} = p^k_{id_2} = \alpha_k$, $k = 1, 2$, for all $d_1, d_2 \in n(i)$. Otherwise, either constraint (41) would be violated or $w^1_i + w^2_i < 2\alpha$ because w^k_i would take the smaller value between $p^k_{id_1}$ and $p^k_{id_2}$, $k = 1, 2$. Therefore, $\mathbf{r}^1$, $\mathbf{r}^2$ are in $R(\mathbf{r})$. Thus, $\mathbf{r}$ is extreme.

Necessity. Let $\mathbf{r}$ be an extreme vector of $\mathcal{W}$. Let $C_\mathbf{r} = \{S \subseteq V : w_i > 0\ \forall\, i \in S,\ S$ is connected in the original tree graph G and is maximal$\}$, $S_d = \{d \in D : u_d > 0\}$ and $S_{id} = \{\{i, d\} \in E^t : v_{id} > 0\}$ based on this $\mathbf{r}$. In the following proof, to prove that a given ray $\mathbf{r}$ is not 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 $\mathbf{2}r - r^1$. Then, $\mathbf{r} = \frac{1}{2}(r^1 + r^2)$ by design.

First, we consider the situation where $C_\mathbf{r} = \emptyset$ and assume that $|S_d| + |S_{id}| > 1$. Let $\mathbf{r}^1$contain all but one of the positive components in $\mathbf{r}$ with their values doubled. Thus, if $|S_d| + |S_{id}| > 1$, $\mathbf{r}$ is not extreme, contrary to the assumption. We conclude that if $C_\mathbf{r} = \emptyset$, then $|S_d| + |S_{id}| = 1$ and thus, $\mathbf{r}$ is either in the form of Case 1 or Case 2. Figure B2 illustrates this situation for the problem considered in Figure B1. Here, a shaded node indicates that the node has an associated dual variable with a positive value.

Now consider the situation when $C_\mathbf{r} \neq \emptyset$. When $|C_\mathbf{r}| > 1$, there is more than one connected component. Consider any set $S \in C_\mathbf{r}$. $\mathbf{r}^1$ has values $w^1_i = 2w_i$ and $u^1_d = 2u_d$, $v^1_{id} = 2v_{id}$ for all $d \in n(i)$ and all $i \in S$ and 0s in the other components. Thus, if $|C_\mathbf{r}| > 1$, $\mathbf{r}$ is not extreme, contrary to the assumption. Figure B3 illustrates this situation. The two components in $C_\mathbf{r}$ are denoted by S' and S''.

When $|C_\mathbf{r}| = 1$ (so that the set of nodes with $w_i > 0$ in the original tree G forms a single connected component) and $S \in C_\mathbf{r}$, define $S_0 = \{\{j, d\} \in E^t : p_{jd} > 0, j \in V \backslash S\}$. Recall that we use p_{jd} to generically represent either of the positive components u_d or v_{jd}. Suppose that $S_0 \neq \emptyset$; let $\mathbf{r}^1$ have $w^1_i = 2w_i$ and $u^1_d = 2u_d$, $v^1_{id} = 2v_{id}$ for all $d \in n(i)$ and all $i \in S$ and 0s in the other components. Then, $\mathbf{r}^2$ is feasible and has at least one positive component because S_0 is not empty, and its corresponding components are not in $\mathbf{r}^1$. Thus, if $|C_\mathbf{r}| = 1$ and $S_0 \neq \emptyset$, $\mathbf{r}$ is not extreme, contrary to the assumption. Figure B4 illustrates this situation.

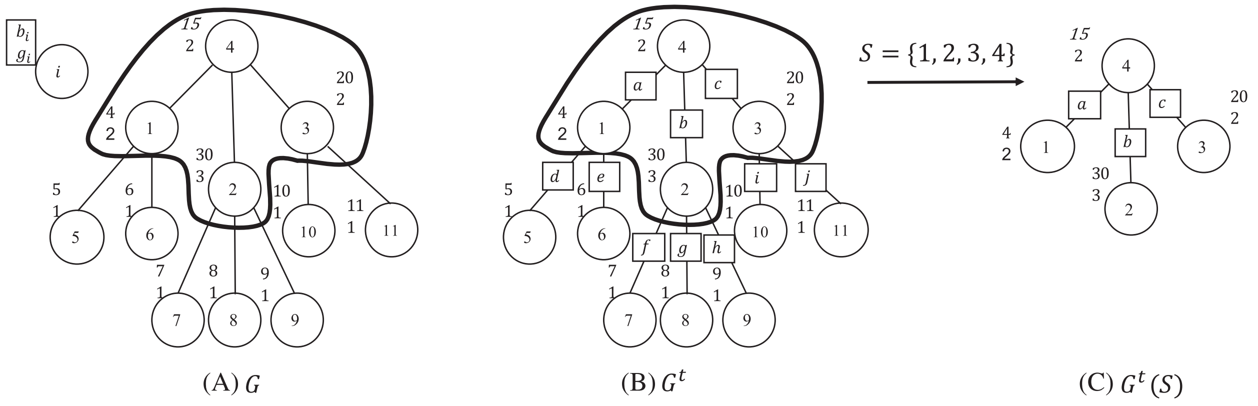


FIGURE B1 Illustration of notation in Theorem 3

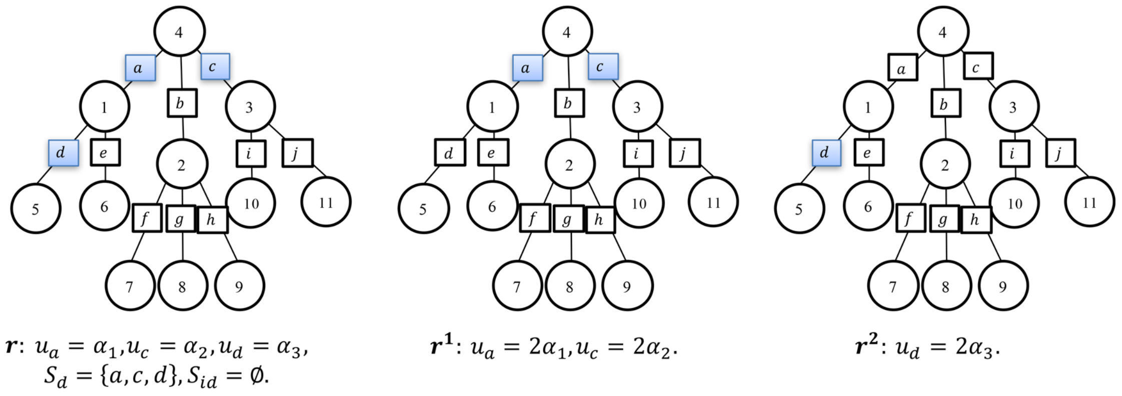
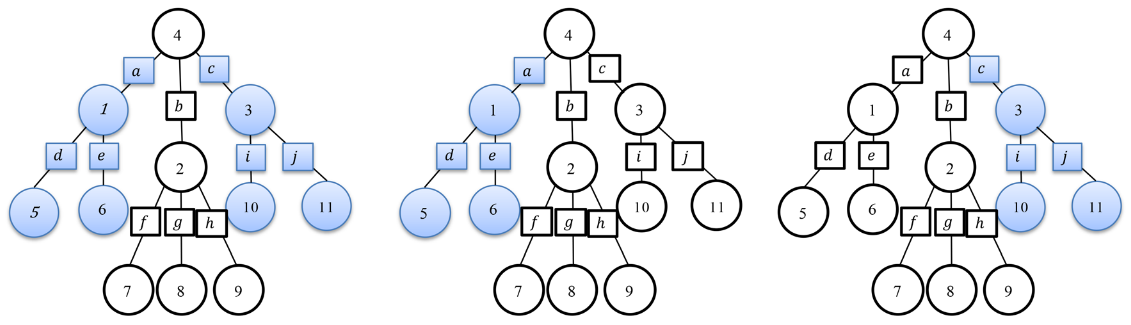

4
a
c
1
b
3
d
e
2
i
j
5
6
10
11
f
g
h
7
8
9
$\mathbf{r}$: $u_a = \alpha_1, u_c = \alpha_2, u_d = \alpha_3,$
$S_d = \{a, c, d\}, S_{id} = \emptyset.$
$\mathbf{r^1}$: $u_a = 2\alpha_1, u_c = 2\alpha_2.$
$\mathbf{r^2}$: $u_d = 2\alpha_3.$

FIGURE B2 $C_\mathbf{r} = \emptyset$ and $|S_d| + |S_{id}| > 1$: figure shows that if $C_\mathbf{r} = \emptyset$, $\mathbf{r}$ must be of the form Case 1 or Case 2 [Color figure can be viewed at wileyonlinelibrary.com]

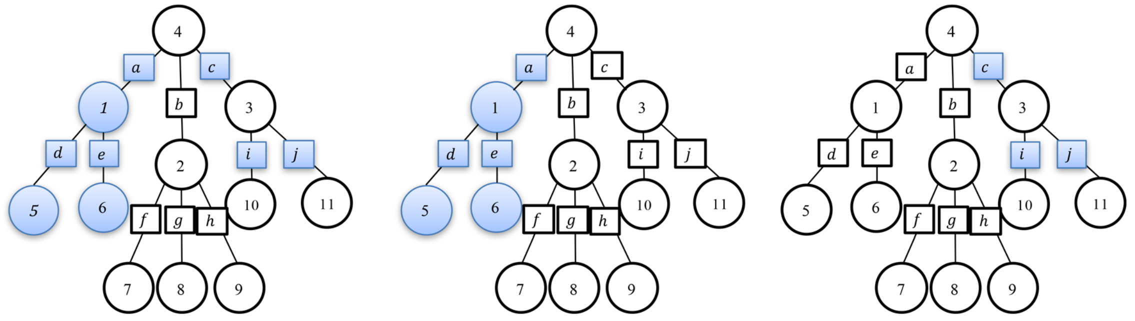


$\mathbf{r}$: $w_1 = \alpha_1, w_5 = \alpha_2, w_6 = \alpha_3,$
$w_3 = \alpha_4, w_{10} = \alpha_5, w_{11} = \alpha_6,$
$u_a = \alpha_7, u_d = \alpha_8, u_e = \alpha_9$
$u_c = \alpha_{10}, u_i = \alpha_{11}, u_j = \alpha_{12},$
$S' = \{1,5,6\}, S'' = \{3,10,11\}.$

$\mathbf{r}^1$: $w_1 = 2\alpha_1, w_5 = 2\alpha_2, w_6 = 2\alpha_3,$
$u_a = 2\alpha_7, u_d = 2\alpha_8, u_e = 2\alpha_9.$

$\mathbf{r}^2$: $w_3 = 2\alpha_4, w_{10} = 2\alpha_5,$
$w_{11} = 2\alpha_6, u_c = 2\alpha_{10},$
$u_i = 2\alpha_{11}, u_j = 2\alpha_{12}.$

FIGURE B3 When $|C_{\mathbf{r}}| > 1$, $\mathbf{r}$ is not extreme [Color figure can be viewed at wileyonlinelibrary.com]



$\mathbf{r}$: $w_1 = \alpha_1, w_5 = \alpha_2, w_6 = \alpha_3,$
$u_a = \alpha_7, u_d = \alpha_8, u_e = \alpha_9,$
$u_c = \alpha_{10}, u_i = \alpha_{11}, u_j = \alpha_{12},$
$S_0 = \{c, i, j\}.$

$\mathbf{r}^1$: $w_1 = 2\alpha_1, w_5 = 2\alpha_2, w_6 = 2\alpha_3,$
$u_a = 2\alpha_7, u_d = 2\alpha_8, u_e = 2\alpha_9.$

$\mathbf{r}^2$: $u_c = 2\alpha_{10}, u_i = 2\alpha_{11},$
$u_j = 2\alpha_{12}.$

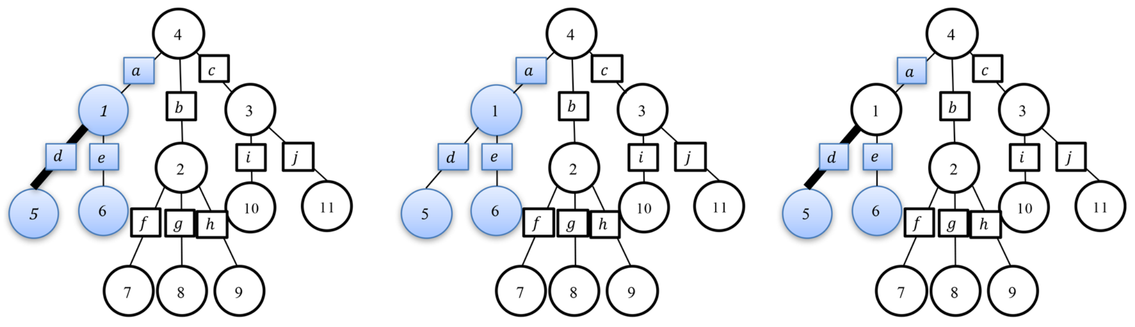
FIGURE B4 When $|C_{\mathbf{r}}| = 1$ and $S_0 \neq \emptyset$ (some u's and v's outside S have positive values), $\mathbf{r}$ is not extreme [Color figure can be viewed at wileyonlinelibrary.com]

When $|C_{\mathbf{r}}| = 1$, define $S_1 = \{\{i, d\} \in E^t : i \in S, u_d > 0 \oplus v_{id} > 0\}$, where only one of u and v variables is positive, and $S_2 = \{\{i, d\} \in E^t : i \in S, u_d > 0, v_{id} > 0\}$, where both u and v variables are positive. Suppose that $S_2 \neq \emptyset$; let $\alpha^1 = 2\min\{w_i, u_d, v_{id} : i \in S, (i, d) \in S_2\}$; then, we make $\mathbf{r}^1$ have $w_i^1 = \alpha^1$ for all $i \in S$. For $\{i, d\} \in S_2$, we have $v_{id}^1 = \alpha^1$. Also, for $\{i, d\} \in S_1$, if $u_d > 0$, we have $u_d^1 = \alpha^1$. Otherwise, we have $v_{id}^1 = \alpha^1$. The remaining components are 0s. Then, $\mathbf{r}^1$, $\mathbf{r}^2$ are different in at least one direction because the component that is $\{w_i, u_d, v_{id} : i \in S, \{i, d\} \in S_2\}$ is positive in $\mathbf{r}^1$, but zero in $\mathbf{r}^2$. Thus, we must have $|S_2| = \emptyset$. Figure B5 illustrates this situation.

Next, suppose that $|C_{\mathbf{r}}| = 1$ and $|S_2| = \emptyset$; consider a node $i \in S$ and let $w_i = \alpha$. Define $S^+ = \{\{i, d\} \in S_1 : p_{id} > \alpha\}$. When $S^+ \neq \emptyset$, consider an edge $\{i, d\} \in S^+$ with $u_d > 0$. We can make $\mathbf{r}^1$ have $u_d^1 = 2(u_d - \alpha)$ and 0s in the other components. Then, $\mathbf{r}^1$, $\mathbf{r}^2$ are different in at least one direction because $\mathbf{r}^1$ only has one positive component, and $\mathbf{r}^2$ has at least two positive components. Thus, if $|C_{\mathbf{r}}| = 1$, then $S^+ = \emptyset$. Figure B6 illustrates this situation.

Next, define $S_v = \{\{i, d\} \in G^t(S) : v_{id} > 0\}$. Suppose that $S_v \neq \emptyset$; consider an edge $\{i, d\} \in S_v$ with j adjacent to d. We decompose S into two connected components (in the original tree graph G) by using the edge $\{i, j\}$ (removing $\{i, j\}$ in G separates S into two connected components). Let $S(i)$ be the one containing node i. We make $\mathbf{r}^1$ have $w_i^1 = 2w_i$, $u_d^1 = 2u_d$, and $v_{id}^1 = 2v_{id}$ for all i in $S(i)$ and d in $n(i)$. Additionally, it has 0s in the remaining components. Thus, if $|C_{\mathbf{r}}| = 1$, then $S_v = \emptyset$. Figure B7 illustrates this situation.

This leaves a single possibility: $|C_{\mathbf{r}}| = 1$, $S_0 = \emptyset$, $S_2 = \emptyset$, $S^+ = \emptyset$, and $S_v = \emptyset$. Consider an edge $\{i, j\}$ in the original graph G where both $i, j \in S$; and the edges $\{i, d\}$ and $\{d, j\}$ in $G^t(S)$ are associated with $\{i, j\}$. Here, only $u_d > 0$ because $S_v = \emptyset$. Thus, $w_i = w_j = u_d$ because $S^+ = \emptyset$. Given that S is a connected component, we have $w_i = w_j$ for i, j in S; thus, $u_d = \alpha$ for $d \in D(S)$. This proves that if $C_{\mathbf{r}} \neq \emptyset$, then $\mathbf{r}$ must be in the form of Case 3. ■

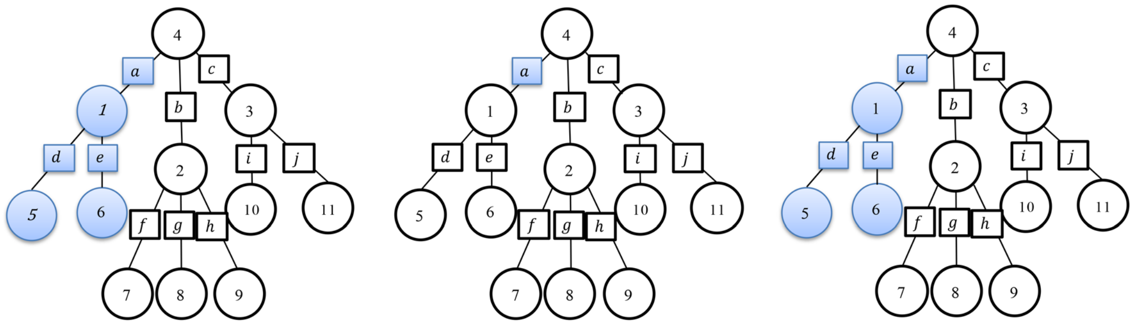


$\mathbf{r}$: $w_1=\alpha_1, w_5=\alpha_2, w_6=\alpha_3,$
$u_a=\alpha_4, u_d=\alpha_5, u_e=\alpha_6$
$v_{1d}=\alpha_7, v_{5d}=\alpha_8,$
$S_1=\{\{1,a\},\{1,e\},\{6,e\}\},$
$S_2=\{\{1,d\},\{5,d\}\}.$
Let α_1 be the smallest one.

$\mathbf{r}^1$: $w_1=2\alpha_1, w_5=2\alpha_1, w_6=2\alpha_1,$
$u_a=2\alpha_1, u_d=2\alpha_1, u_e=2\alpha_1,$
$v_{1d}=\alpha_1, v_{5d}=\alpha_1.$

$\mathbf{r}^2$: $w_5=\beta_2, w_6=\beta_3,$
$u_a=\beta_4, u_d=\beta_5, u_e=\beta_6,$
$v_{1d}=\beta_7, v_{5d}=\beta_8,$
$\beta_i=2(\beta_i-\alpha_1)$ $i=2,3\ldots,8.$

FIGURE B5 When $|C_{\mathbf{r}}|=1$, $S_0=\emptyset$, and $S_2\neq\emptyset$ (u_d and v_{id} are nonzero for some node $i\in S$), $\mathbf{r}$ is not extreme (the bold line represents the associated dual variable v and takes a positive value) [Color figure can be viewed at wileyonlinelibrary.com]

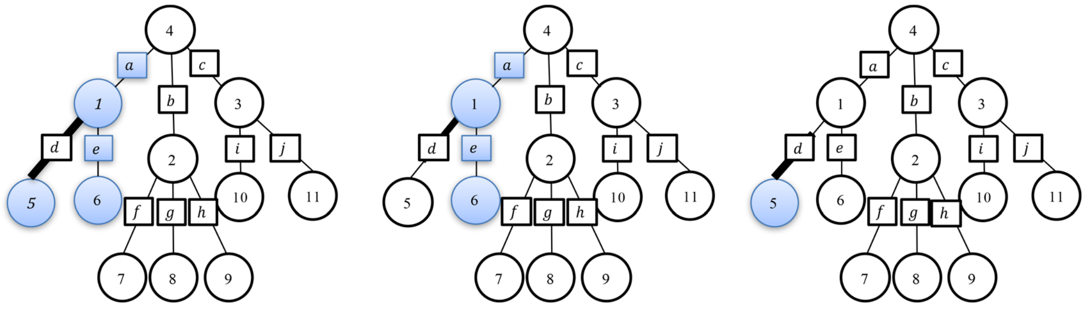


$\mathbf{r}$: $w_1=\alpha_1, w_5=\alpha_2, w_6=\alpha_3,$
$u_a=\alpha_1+\epsilon, u_d=\alpha_5, u_e=\alpha_6,$
$S^+=\{\{1,a\}\}.$

$\mathbf{r}^1$: $u_a=2\epsilon.$

$\mathbf{r}^2$: $w_1=2\alpha_1, w_5=2\alpha_2, w_6=2\alpha_3,$
$u_a=2\alpha_1, u_d=2\alpha_5, u_e=2\alpha_5.$

FIGURE B6 When $|C_{\mathbf{r}}|=1$, $S_0=\emptyset$, $S_2=\emptyset$, and $S^+\neq\emptyset$ (either u_d or v_{id} is greater than α for some node $i\in S$), $\mathbf{r}$ is not extreme [Color figure can be viewed at wileyonlinelibrary.com]



$\mathbf{r}$: $w_1=\alpha_1, w_5=\alpha_2, w_6=\alpha_3,$
$u_a=\alpha_4, u_e=\alpha_5, v_{1d}=\alpha_6, v_{5d}=\alpha_7,$
$S_v=\{\{1,d\},\{5,d\}\}.$

$\mathbf{r}^1$: $w_1=2\alpha_1, w_6=2\alpha_3,$
$u_a=2\alpha_4, u_e=2\alpha_5, v_{1d}=2\alpha_6.$

$\mathbf{r}^2$: $w_5=2\alpha_2, v_{5d}=2\alpha_7.$

FIGURE B7 When $|C_{\mathbf{r}}|=1$, $S_0=\emptyset$, $S_2=\emptyset$, $S^+=\emptyset$, and $S_v\neq\emptyset$ (v_{id} is nonzero for some edge $\{i,d\}$ in $G'(S)$, the induced graph of S), $\mathbf{r}$ is not extreme [Color figure can be viewed at wileyonlinelibrary.com]