A Branch-and-cut Algorithm for Discrete Bilevel Linear Programs

Junlong Zhang, Osman Y. Özaltın

Edward P. Fitts Department of Industrial and Systems Engineering, North Carolina State University, 400 Daniels Hall, Raleigh, NC 27695, USA.
Email: oyozalti@ncsu.edu

Abstract

We present a branch-and-cut algorithm for solving discrete bilevel linear programs where the upper-level variables are binary and the lower-level variables are either pure integer or pure binary. This algorithm performs local search to find improved bilevel-feasible solutions. We strengthen the relaxed node subproblems in the branch-and-cut search tree by generating cuts to eliminate all of the upper-level integer feasible solutions explored during the local search step. Furthermore, to avoid repeatedly solving similar bilevel integer problems in the local search, we derive structural properties of the value functions of bilevel integer programs. Most notably, we generalize the integer complementary slackness theorem to bilevel integer programs. We also propose three efficient approaches for computing the bilevel programming value functions. Our computational results show that the proposed branch-and-cut approach enhanced with the implementation of value functions and local search can solve discrete bilevel programming instances with up to 200 variables and 150 constraints in a reasonable amount of time.

Keywords: Bilevel Programming; Integer Programming; Branch and Cut; Value Functions

1 Introduction

Bilevel programs [2, 13, 16] model the hierarchical relationship between two autonomous and possibly conflicting decision makers: the *leader* and the *follower*. Each decision maker controls a distinct set of variables and the decisions are made sequentially according to a hierarchy: the upper-level decisions are made first by the leader, after which the lower-level decisions are made by the follower subject to constraints which depend on the leader's decisions. The follower's decisions in return affect the leader's objective function. Thus, the leader should act by considering the follower's response.

Bilevel programming is closely related to the static Stackelberg leader-follower game [10, 40] and interdiction problems [24, 37, 42]. Furthermore, it provides a flexible modeling approach for decentralized decision-making in several important application domains including

hazardous material transportation [22], network design [12, 36], revenue management [14], traffic planning [29, 34], energy [5], computational biology [9, 35] and defense [8] areas.

We consider discrete bilevel linear programs (DBLPs) where the upper-level variables are binary and the lower-level variables are either pure integer or pure binary. We propose a finite branch-and-cut (B&C) algorithm as an exact solution method. To avoid repeatedly solving similar lower-level problems for different upper-level solutions in the B&C algorithm, we study the properties of the value functions of bilevel integer programs. Most notably, we generalize the integer complementary slackness theorem to bilevel integer programs. We also propose three efficient approaches for computing the bilevel programming value functions. These value functions are then utilized when calculating bounds in the B&C algorithm.

Any linear mixed 0–1 programming problem can be reduced to a bilevel linear program (BLP), in which the leader's and the follower's problems are both linear programs [1]. Therefore, BLPs, and so bilevel programs in general, are strongly NP-hard [23]. For BLPs, Bard and Moore [3] proposed a branch-and-bound algorithm that branches on complementarity conditions. Hansen et al. [23] extended this algorithm by exploiting the necessary optimality conditions of the follower's problem. Another approach by Júdice and Faustino [26] used complementary pivoting to achieve ϵ-optimal solutions. We refer the reader to Ben-Ayed [6] for a detailed survey on BLPs.

For mixed-integer bilevel programs (BMIPs) including mixed-integer linear programs in the upper and lower levels, Moore and Bard [31] proposed a branch-and-bound algorithm. DeNegre [19] improved this algorithm using cutting planes. Lozano and Smith [28] studied BMIPs with integer upper-level variables and proposed an exact algorithm based on value function reformulation. Vicente et al. [39] used penalty functions to reformulate BMIPs into bilinear programs. Audet et al. [1], Dempe [15] and Wen and Yang [41] proposed solution approaches for specific classes of BMIPs where either the leader's or the follower's variables are all continuous. Dempe and Richter [18], Özaltın et al. [33] and Brotcorne et al. [7] considered the bilevel knapsack problem, an extension of the classical knapsack problem to the bilevel framework. Detailed surveys on bilevel programming solution techniques were presented in [2, 16, 30].

We review below previous work in the literature that is closely related to the problem studied in this paper. Bard and Moore [4] studied bilevel programs with binary upper- and lower-level variables. They proposed an algorithm that implicitly enumerates the upper-level variables and solves the associated lower-level problems to obtain bilevel feasible solu-

tions. The authors reported solving instances with up to 45 variables and 18 constraints. Caramia and Mari [11] and DeNegre and Ralphs [20] studied bilevel integer programs (BIPs) with integer upper- and lower-level variables. They derived valid inequalities to eliminate bilevel-infeasible solutions for a given upper-level solution in a branch-and-bound framework. Caramia and Mari [11] reported solving BIPs with up to 25 variables and 25 constraints, while DeNegre and Ralphs [20] performed tests using a set of interdiction problems with up to 34 variables and 19 constraints. Zeng and An [43] proposed a single-level reformulation for BMIPs by enforcing the optimality conditions for the continuous variables for each fixed value of the integer variables in the lower-level problem. They offered a column-and-constraint generation algorithm to solve this reformulation. The authors reported solving pure integer instances with up to 20 variables and 30 constraints. Dempe and Kue [17] also studied BIPs in which upper-level variables only affect the lower-level objective function. The authors proposed a B&C algorithm based on an optimal value reformulation of the follower's problem and presented small illustrative examples.

Our proposed solution approach differs from the existing methods in the literature because we utilize value functions of BIPs in our B&C algorithm to avoid repeatedly solving similar lower-level problems. In addition, we implement local search to find improved bilevel-feasible solutions. Finally, we strengthen the relaxed node subproblems in the B&C algorithm by generating cuts to eliminate all of the upper-level integer feasible solutions explored during the local search step. Our numerical results show that the proposed B&C algorithm enhanced with the implementation of value functions and local search can solve DBLP instances with up to 200 variables and 150 constraints in a reasonable amount of time.

The remainder of this paper is organized as follows. Section 2 presents the formulation of the DBLP. Section 3 describes the proposed B&C algorithm. Section 4 identifies various properties of the value functions of bilevel integer programs, which are subsequently used in our algorithms derived in Section 5 for constructing the value functions of BIPs. Section 6 presents the results of our computational experiments. Finally, Section 7 concludes the paper with future research directions.

2 Model formulation

We consider the following class of discrete bilevel linear programs:

$$(DBLP): \quad \max \quad c^T x + d^T y \tag{1a}$$

$$\text{subject to} \quad x \in \mathbb{B}^{n_1}, \tag{1b}$$

$$y \in \operatorname{argmax}\left\{t^T y \mid Ay \leq b - Bx,\ y \in \mathbb{Y}\right\}, \tag{1c}$$

where x and y represent the leader's and the follower's decision variables, respectively, and A and B are $m_2 \times n_2$ and $m_2 \times n_1$ constraint matrices. The discrete set $\mathbb{Y}$ restricts the follower's variables to be pure binary or pure integer. For ease of presentation, we only consider the upper-level constraint $x \in \mathbb{B}^{n_1}$ in (1b), our proposed methods apply when there are additional constraints on the upper-level variables. Without loss of generality, we assume that the entries of the constraint matrices A and B as well as the entries of the right-hand side vector b are all integers. The *constraint region* of $DBLP$ is defined as:

$$\Omega^I = \{(x, y) \mid Ay + Bx \leq b,\ x \in \mathbb{B}^{n_1},\ y \in \mathbb{Y}\}.$$

For $x \in \mathbb{B}^{n_1}$, the follower's feasible region and rational reaction set are given by $P_L(x) = \{y \in \mathbb{Y} \mid Ay \leq b - Bx\}$ and $R(x) = \operatorname{argmax}\left\{t^T y \mid y \in P_L(x)\right\}$, respectively. We assume that $P_L(x)$ is bounded for $x \in \mathbb{B}^{n_1}$ to avoid pathological cases. Finally, the inducible region of $DBLP$ is defined as $IR = \{(x, y) \mid x \in \mathbb{B}^{n_1},\ y \in R(x)\}$. Any point $(x, y) \in IR$ is *bilevel feasible*, and $DBLP$ can be re-stated as:

$$\max_{(x,y) \in IR} c^T x + d^T y. \tag{2}$$

There are two approaches for modeling the follower's response in bilevel programs [13]: the *optimistic* formulation assumes that if there are multiple optimal solutions in the follower's rational reaction set $R(x)$ for a given decision by the leader, the follower implements the most favorable solution for the leader. On the contrary, the *pessimistic* formulation assumes that the follower implements the least favorable solution for the leader. The optimistic case might arise in collaborative environments when the leader and the follower cooperate, while the pessimistic case is used to model adversarial settings. The $DBLP$ assumes the optimistic case. However, our proposed methods can also be applied in the pessimistic case as discussed in Section 4.2.

3 A branch-and-cut algorithm

In this section, we develop a branch-and-cut (B&C) algorithm to solve the $DBLP$. We first describe the key components of the proposed algorithm and then present its main framework.

3.1 Lower bounding

Lower bounds to $DBLP$ can be obtained from bilevel feasible solutions. For a given decision of the leader $\overline{x} \in \mathbb{B}^{n_1}$, a response from the follower $\hat{y} \in R(\overline{x})$ can be identified by solving the lower-level problem

$$\max \left\{ t^T y \mid y \in P_L(\overline{x}) \right\}. \tag{3}$$

Then, $(\overline{x}, \hat{y})$ is a bilevel feasible solution in the inducible region and provides a lower bound to $DBLP$. If the rational reaction set $R(\overline{x})$ is not a singleton, i.e., if problem (3) has multiple optimal solutions, the most favorable response of the follower to the leader can be selected by solving:

$$\max \left\{ d^T y \mid t^T y \geq t^T \hat{y},\ y \in P_L(\overline{x}) \right\}. \tag{4}$$

Problem (4) characterizes the follower's response by maximizing the leader's benefit, because we consider the optimistic formulation. The constraints in (4) simply impose that $y \in R(\overline{x})$. Let $\overline{y}$ be an optimal solution to (4). Then, $(\overline{x}, \overline{y})$ provides a tighter lower bound than $(\overline{x}, \hat{y})$ since $d^T \overline{y} \geq d^T \hat{y}$ for any $\hat{y} \in R(\overline{x})$.

3.2 Upper bounding

It is well known that relaxing the integrality requirements on the variables does not provide a valid upper bound for $DBLP$ [31]. Instead, we consider the following single-level problem:

$$\max_{(x,y) \in \Omega^I} c^T x + d^T y, \tag{5}$$

which is obtained by removing the lower-level optimality conditions. In other words, we let the leader control the follower's variables. Problem (5) is thus a relaxation of $DBLP$, and its optimal value provides an upper bound for $DBLP$. The linear programming relaxation of (5) is given by:

$$\max_{(x,y) \in \Omega} c^T x + d^T y, \tag{6}$$

where Ω is obtained by removing the integrality requirements in Ω^I. It is clear that the optimal value of (6) also provides an upper bound for $DBLP$.

3.3 Local search

The upper bound obtained from (6) could be very loose. In the implementation of our B&C algorithm, we get improved upper bounds by gradually reducing the size of Ω. Specifically, if an integer upper-level solution $\overline{x}$ is obtained at a B&C node, we perform a local search in the neighborhood of $\overline{x}$ and update the global lower bound accordingly. We then add a cut to remove all upper-level solutions evaluated in the neighborhood of $\overline{x}$ from Ω.

We define the *distance-k neighborhood* of an upper-level solution $\overline{x} \in \mathbb{B}^{n_1}$ as:

$$N_k(\overline{x}) = \left\{ x \in \mathbb{B}^{n_1} \;\middle|\; \sum_{j=1}^{n_1} |x_j - \overline{x}_j| \leq k \right\},$$

where x_j is the j^{th} element of vector x. Note that $N_k(\overline{x})$ consists of upper-level solutions that have at most k different elements from $\overline{x}$. We evaluate the objective value of each upper-level solution $x \in N_k(\overline{x})$ and update the lower bound of $DBLP$ whenever an improved bilevel-feasible solution is found. After performing this local search step, we can exclude all upper-level solutions in $N_k(\overline{x})$ from Ω by adding the following cut:

$$\sum_{j:\overline{x}_j=1} (1 - x_j) + \sum_{j:\overline{x}_j=0} x_j \geq k + 1. \tag{7}$$

Note that (7) only discards the upper-level solutions in $N_k(\overline{x})$.

3.4 The main framework of the B&C algorithm

Algorithm 1 outlines the B&C algorithm proposed for solving the $DBLP$. Let $\mathcal{F}$ be the list of unprocessed tree nodes in the B&C tree, and Γ be the set of cuts generated so far. The bilevel subproblem at each node $\mathcal{P}^v \in \mathcal{F}$ is given by:

$$\max_{(x,y) \in IR_v} c^T x + d^T y, \tag{8}$$

where IR_v denotes the inducible region at node $\mathcal{P}^v$ after enforcing the cuts in set Γ.

We generate a local upper bound at node $\mathcal{P}^v$ by solving the following relaxation of (8):

$$(LP_v): \quad U_v = \max_{(x,y) \in \Omega_v} c^T x + d^T y, \tag{9}$$

where Ω_v is obtained by removing the lower-level optimality conditions and integrality restrictions in IR_v. We fathom node $\mathcal{P}^v$ if problem LP_v is infeasible or its optimal value U_v is no greater than the global lower bound L. If the optimal upper-level solution x^v to LP_v is

integer, we perform a local search in $N_k(x^v)$, update L accordingly, and then add a cut to exclude all $x \in N_k(x^v)$ and solve LP_v again. If x^v is fractional, we use valid inequalities to separate the fractional solution or branch if necessary.

Algorithm 1. *A branch-and-cut algorithm to solve the DBLP.*

Step 0: (Initialization) Create the root node $\mathcal{P}^0$. Initialize the unprocessed node list $\mathcal{F} \leftarrow \{\mathcal{P}^0\}$ and the set of cuts $\Gamma \leftarrow \emptyset$. Set the global lower bound $L = -\infty$ and node count $v = 1$.

Step 1: (Node selection) If $\mathcal{F} = \emptyset$, terminate with optimal objective function value L and optimal solution (x^*, y^*); otherwise, select and delete from $\mathcal{F}$ a node $\mathcal{P}^v$.

Step 2: (Node fathoming) Solve problem LP_v. If LP_v is infeasible, fathom the current node and go to Step 1; otherwise, let (x^v, y^v) be an optimal solution and U_v be the corresponding objective function value. If $U_v \leq L$, fathom the current node and go to Step 1; otherwise, go to Step 3.

Step 3: (Feasibility check) If x^v is binary, go to Step 4; otherwise, go to Step 5.

Step 4: (Local search) Perform local search in $N_k(x^v)$ to update the global lower bound L and the optimal solution (x^*, y^*). Add the cut $\sum_{j:x_j^v=1}(1 - x_j) + \sum_{j:x_j^v=0} x_j \geq k + 1$ to the set of cuts Γ, and apply Γ globally to the B&C tree. Go to Step 2.

Step 5: (Separation) Search for valid inequalities that are violated by x^v. If any are found, add them to problem LP_v and go to Step 2; otherwise, go to Step 6.

Step 6: (Branching) Choose a dimension i, $1 \leq i \leq n_1$, such that x_i^v is fractional. Create two child nodes $\mathcal{P}^{v_1}$ and $\mathcal{P}^{v_2}$ by adding constraints $x_i \leq \lfloor x_i^v \rfloor$ and $x_i \geq \lceil x_i^v \rceil$, respectively. Set $\mathcal{F} \leftarrow \mathcal{F} \cup \{\mathcal{P}^{v_1}, \mathcal{P}^{v_2}\}$. Set $v = v + 1$ and go to Step 1.

Theorem 1. *Algorithm 1 terminates with an optimal solution (x^*, y^*) to the DBLP after a finite number of iterations.*

Theorem 1 follows because the set of binary upper-level feasible solutions is finite, and the local search in Step 4 evaluates a distinct subset of these solutions at every iteration.

Note that we add cuts to remove feasible upper-level solutions in Step 4, whereas the valid inequalities generated in Step 5 eliminate fractional upper-level solutions in problem LP_v. Node selection and branching rules used in Step 1 and Step 6, respectively, may significantly affect the computational performance. We implement Algorithm 1 using the callback functions of CPLEX (version 12.7.1). The default settings of CPLEX are used for node selection. For branching and separation, CPLEX is set to branch only on the upper-level variables x and separate only fractional upper-level solutions due to the following theorem.

Theorem 2. *[31] Let (x^v, y^v) be an optimal solution to problem LP_v at node v, and U_v be the corresponding objective function value. If no restrictions on the lower-level variables y have been made along the path to node v, then U_v is an upper bound on the optimal value of problem* (8).

4 The value functions of bilevel integer programs

In the proposed B&C algorithm, integer problems (3) and (4) are solved for different upper-level solutions during the local search. To avoid repeatedly solving similar integer programs, we will utilize the value functions of bilevel integer programs. We focus on the case where all of the lower-level variables in $DBLP$ are nonnegative integers, i.e., $\mathbb{Y} = \mathbb{Z}_+^{n_2}$ in the rest of this section. We first review the basic properties of the value functions of integer programs (IPs). We then study the value functions of bilevel integer programs (BIPs). These properties are then utilized in the algorithms proposed for constructing the BIP value function in Section 5.

4.1 Properties of the IP value functions

We define the follower's value function as:

$$\psi(\beta) = \max\left\{t^T y \mid Ay \leq \beta,\ y \in \mathbb{Z}_+^{n_2}\right\},\ \beta \in \mathbb{Z}^{m_2}. \tag{10}$$

Note that $\beta = b - B\overline{x}$ for a given upper-level solution $\overline{x} \in \mathbb{B}^{n_1}$. We represent the feasible region of the follower's problem by $S(\beta) = \{y \in \mathbb{Z}_+^{n_2} \mid Ay \leq \beta\}$ for any $\beta \in \mathbb{Z}^{m_2}$. We assume that $\psi(\beta) = -\infty$ if $S(\beta) = \emptyset$, and define $\mathbf{B} = \{\beta \in \mathbb{Z}^{m_2} \mid S(\beta) \neq \emptyset\}$. For any $\beta \in \mathbf{B}$, the optimal solution set of the follower's problem is given by $\widehat{opt}(\beta) = \text{argmax}\{t^T y \mid y \in S(\beta)\}$. Let t_j denote the j^{th} element of vector t, and a_j be the j^{th} column of matrix A. We define $B_j = \{\beta \in \mathbf{B} \mid \beta \geq a_j\}$ for $j \in \{1, \cdots, n_2\}$. Proposition 1 summarizes some of the important properties of the IP value functions.

Proposition 1. *We have the following results that are compiled in Nemhauser and Wolsey [32].*

(i) $\psi(\cdot)$ *is nondecreasing in* $\beta \in \mathbb{Z}^{m_2}$.

(ii) $\psi(\cdot)$ *is superadditive over* $\mathbf{B}$, *i.e., for all* $\beta_1, \beta_2 \in \mathbf{B}$, *if* $\beta_1+\beta_2 \in \mathbf{B}$, *then* $\psi(\beta_1)+\psi(\beta_2) \leq \psi(\beta_1 + \beta_2)$.

(iii) $\psi(a_j) \geq t_j$ *for* $j = 1, \ldots, n_2$. *If* $\psi(a_j) > t_j$, *then* $\hat{y}_j = 0$ *for all* $\beta \in \mathbf{B}$ *and* $\hat{y} \in \widehat{opt}(\beta)$.

(iv) (Integer Complementary Slackness) If $\hat{y} \in \widehat{opt}(\beta)$, *then* $\psi(Ay) = t^T y$ *and* $\psi(Ay) + \psi(\beta - Ay) = \psi(Ay) + \psi(A(\hat{y} - y)) = \psi(\beta)$, *for all* $y \in \mathbb{Z}^{n_2}_+$ *such that* $y \leq \hat{y}$.

We make the following two observations from Proposition 1.

Lemma 1. *Let* $\hat{y} \in \widehat{opt}(\beta)$ *for* $\beta \in \mathbf{B}$. *Then* $\psi(\beta') = \psi(\mathbf{0})$ *for* $\beta' \in \mathbf{B}$ *and* $\mathbf{0} \leq \beta' \leq \beta - A\hat{y}$.

Proof. It follows from (i) that $\psi(\mathbf{0}) \leq \psi(\beta') \leq \psi(\beta - A\hat{y})$. Furthermore, we have $\psi(\beta - A\hat{y}) = \psi(A(\hat{y} - \hat{y})) = \psi(\mathbf{0})$ from (iv). □

Lemma 2. *Let* $\hat{y} \in \widehat{opt}(\beta)$ *for* $\beta \in \mathbf{B}$. *Then* $\psi(\beta') = \psi(\beta)$ *for* $\beta' \in \mathbf{B}$ *and* $A\hat{y} \leq \beta' \leq \beta$.

Proof. It follows from (i) that $\psi(A\hat{y}) \leq \psi(\beta') \leq \psi(\beta)$. In addition, $\psi(A\hat{y}) = \psi(\beta)$ since $\hat{y} \in \widehat{opt}(\beta)$. □

Proposition 2. *[27] For all* $\beta \in \mathbf{B} \setminus \cup_{j=1}^{n_2} B_j$, $\psi(\beta) = 0$.

Proposition 3. *[21] For any* $\beta \in \cup_{j=1}^{n_2} B_j$,

$$\psi(\beta) = \max_j \left\{ t_j + \psi(\beta - a_j) \mid a_j \in \mathbf{B},\ j = 1, \ldots, n_2 \right\}.$$

A right-hand side vector $\beta \in \mathbf{B}$ is called *level-set minimal* if $\psi(\beta - e_i) < \psi(\beta)$ for all $i = 1, \cdots, m_2$, where e_i is the i^{th} unit vector [38]. Let $\tilde{\mathbf{B}}$ be the set of all level-set minimal vectors. In the following theorem, Trapp et al. [38] showed that the IP value function $\psi(\cdot)$ can be entirely characterized on $\tilde{\mathbf{B}}$.

Theorem 3. *[38] For any* $\beta \in \mathbf{B}$,

$$\psi(\beta) = \max_{\tilde{\beta} \in \tilde{\mathbf{B}}: \tilde{\beta} \leq \beta} \psi(\tilde{\beta}).$$

4.2 Properties of the BIP value functions

We consider the following BIP value function:

$$\phi(\beta) = \max\left\{d^T y \mid t^T y \geq \psi(\beta),\ y \in S(\beta)\right\},\ \beta \in \mathbb{Z}^{m_2}.$$

Note that $\beta = b - B\overline{x}$ for a given upper-level solution $\overline{x} \in \mathbb{B}^{n_1}$, and $\phi(\beta)$ denotes the contribution of the follower's decision to the leader's objective. We assume that $\phi(\beta) = -\infty$ if $S(\beta) = \emptyset$. Otherwise, we characterize the follower's response to the leader by $opt(\beta) = \text{argmax}\left\{d^T y \mid t^T y \geq \psi(\beta),\ y \in S(\beta)\right\}$. We define $\phi(\beta)$ as a maximization problem to model the optimistic case. The pessimistic case can be considered by changing maximization to minimization in the definition of $\phi(\beta)$.

Remark 1. *$\phi(\cdot)$ is not monotone in $\beta \in \mathbb{Z}^{m_2}$, and not superadditive over $\mathbf{B}$.*

Proposition 4 generalizes the integer complementary slackness theorem to bilevel integer programs. This is an important theoretical result on its own. However, it also has computational significance because complementary slackness can be leveraged when constructing $\phi(\cdot)$ to easily obtain the objective values for some of the right-hand sides.

Proposition 4 (*Bilevel integer complementary slackness*)**.** *If $\hat{y} \in opt(\beta)$ for $\beta \in \mathbf{B}$, then $\phi(Ay) = d^T y$ and $\phi(Ay) + \phi(\beta - Ay) = \phi(Ay) + \phi(A(\hat{y} - y)) = \phi(\beta)$ for all $y \leq \hat{y}$, $y \in \mathbb{Z}^{n_2}_+$.*

Proof. If $\hat{y} \in opt(\beta)$, then $\hat{y} \in \widehat{opt}(\beta)$. It follows from integer complementary slackness in Proposition 1 that $y \in \widehat{opt}(Ay)$ and $\psi(Ay) = t^T y$ for all $y \leq \hat{y}$, $y \in \mathbb{Z}^{n_2}_+$. We will do the proof in three steps. First, we will show $\phi(Ay) = d^T y$, second we will show $\phi(Ay) + \phi(A(\hat{y} - y)) = \phi(\beta)$, and finally we will show $\phi(\beta - Ay) = \phi(A(\hat{y} - y))$.

i) Suppose that $y \notin opt(Ay)$. Then, $\exists \tilde{y} \in \widehat{opt}(Ay)$ such that $d^T\tilde{y} > d^T y$. Consider the vector $\overline{y} = \tilde{y} + \hat{y} - y$. We have $\overline{y} \in \mathbb{Z}^{n_2}_+$ and

$$\begin{aligned}
A\overline{y} &= A\tilde{y} + A\hat{y} - Ay \leq A\hat{y} \leq \beta,\ (\text{since } A\tilde{y} \leq Ay)\\
t^T\overline{y} &= t^T\tilde{y} + t^T\hat{y} - t^T y = t^T\hat{y} = \psi(\beta),\ (\text{since } y, \tilde{y} \in \widehat{opt}(Ay))\\
d^T\overline{y} &= d^T\tilde{y} + d^T\hat{y} - d^T y > d^T\hat{y},\ (\text{since } d^T\tilde{y} > d^T y)
\end{aligned}$$

which indicates that $\overline{y} \in \widehat{opt}(\beta)$ and it has a larger objective function value than $\hat{y}$. This contradicts $\hat{y} \in opt(\beta)$. We thus conclude that $y \in opt(Ay)$ and $\phi(Ay) = d^T y$.

ii) Consider now $\hat{y} - y$. Since $\hat{y} - y \in \mathbb{Z}^{n_2}_+$ and $\hat{y} - y \leq \hat{y}$, we know $\hat{y} - y \in opt(A(\hat{y} - y))$ from the result proved in part *i*). Then $\phi(Ay) + \phi(A(\hat{y} - y)) = d^T y + d^T(\hat{y} - y) = d^T\hat{y} = \phi(\beta)$.

iii) From integer complementary slackness in Proposition 1, we know that $\psi(\beta - Ay) = \psi(A(\hat{y} - y)) = t^T(\hat{y} - y)$. Also, $A(\hat{y} - y) \leq \beta - Ay$. Hence $\hat{y} - y \in \widehat{opt}(\beta - Ay)$. Suppose $\hat{y} - y \notin opt(\beta - Ay)$. Then $\exists \tilde{y}$ such that $d^T\tilde{y} > d^T(\hat{y} - y)$ and $\tilde{y} \in \widehat{opt}(\beta - Ay)$. Consider $\tilde{y} + y = \tilde{y} + \hat{y} - (\hat{y} - y)$. We have $A\tilde{y} \leq \beta - Ay$ and thus $A(\tilde{y} + y) \leq \beta$. We also have $\tilde{y} + y \in \mathbb{Z}_+^{n_2}$ and

$$
\begin{aligned}
&t^T(\tilde{y} + y) = t^T\tilde{y} + t^T\hat{y} - t^T(\hat{y} - y) = t^T\hat{y} = \psi(\beta), \text{ (since } \tilde{y} \text{ and } \hat{y} - y \in \widehat{opt}(\beta - Ay)) \\
&d^T(\tilde{y} + y) = d^T\tilde{y} + d^T\hat{y} - d^T(\hat{y} - y) > d^T\hat{y},
\end{aligned}
$$

which indicate that $\tilde{y} + y \in \widehat{opt}(\beta)$ and it has a larger objective function value than $\hat{y}$. This contradicts the fact that $\hat{y} \in opt(\beta)$. We thus conclude that $\hat{y} - y \in opt(\beta - Ay)$ and $\phi(\beta - Ay) = \phi(A(\hat{y} - y))$. □

Next, Lemma 3 and Lemma 4 generalize Lemma 1 and Lemma 2 to bilevel integer programs, respectively.

Lemma 3. *Let* $\hat{y} \in opt(\beta)$ *for* $\beta \in \mathbf{B}$. *Then* $\phi(\beta') = \phi(\mathbf{0})$ *for* $\mathbf{0} \leq \beta' \leq \beta - A\hat{y}$, $\beta' \in \mathbf{B}$.

Proof. If $\hat{y} \in opt(\beta)$, then $\hat{y} \in \widehat{opt}(\beta)$. Thus, $\psi(\beta') = \psi(\mathbf{0})$ for $\mathbf{0} \leq \beta' \leq \beta - A\hat{y}$, $\beta' \in \mathbf{B}$ from Lemma 1. Then

$$
\begin{aligned}
\phi(\mathbf{0}) &= \max\left\{d^T y \mid t^T y \geq \psi(\mathbf{0}),\ Ay \leq \mathbf{0},\ y \in \mathbb{Z}_+^{n_2}\right\} \\
&\leq \max\left\{d^T y \mid t^T y \geq \psi(\beta'),\ Ay \leq \beta',\ y \in \mathbb{Z}_+^{n_2}\right\} = \phi(\beta') \\
&\leq \max\left\{d^T y \mid t^T y \geq \psi(\beta - A\hat{y}),\ Ay \leq \beta - A\hat{y},\ y \in \mathbb{Z}_+^{n_2}\right\} = \phi(\beta - A\hat{y}).
\end{aligned}
$$

The result follows since $\phi(\beta - A\hat{y}) = \phi(A(\hat{y} - \hat{y})) = \phi(\mathbf{0})$ by Proposition 4. □

Lemma 4. *Let* $\hat{y} \in opt(\beta)$ *for* $\beta \in \mathbf{B}$. *Then* $\phi(\beta') = \phi(\beta)$ *for* $A\hat{y} \leq \beta' \leq \beta$, $\beta' \in \mathbf{B}$.

Proof. If $\hat{y} \in opt(\beta)$, then $\hat{y} \in \widehat{opt}(\beta)$. Thus, $\psi(\beta') = \psi(\beta)$ for $A\hat{y} \leq \beta' \leq \beta$, $\beta' \in \mathbf{B}$ from Lemma 2. Then

$$
\begin{aligned}
\phi(A\hat{y}) &= \max\left\{d^T y \mid t^T y \geq \psi(A\hat{y}),\ Ay \leq A\hat{y},\ y \in \mathbb{Z}_+^{n_2}\right\} \\
&\leq \max\left\{d^T y \mid t^T y \geq \psi(\beta'),\ Ay \leq \beta',\ y \in \mathbb{Z}_+^{n_2}\right\} = \phi(\beta') \\
&\leq \max\left\{d^T y \mid t^T y \geq \psi(\beta),\ Ay \leq \beta,\ y \in \mathbb{Z}_+^{n_2}\right\} = \phi(\beta).
\end{aligned}
$$

The result follows since $\phi(\beta) = \phi(A\hat{y})$ by Proposition 4. □

Once we compute $\hat{y} \in opt(\beta)$ for $\beta \in \mathbf{B}$, Lemma 3 and 4 yield the objective value for additional right-hand sides without solving the corresponding bilevel integer programs.

Lemma 5. *$\phi(\beta) = 0$ for $\beta \in \mathbf{B} \backslash \cup_{j=1}^{n_2} B_j$.*

Proof. The result follows since $\widehat{opt}(\beta) = \{\mathbf{0}\}$ for $\beta \in \mathbf{B} \backslash \cup_{j=1}^{n_2} B_j$. □

Proposition 5. *For any $\beta \in \cup_{j=1}^{n_2} B_j$, if there exists $\hat{y} \in opt(\beta)$ such that $\hat{y} \neq \mathbf{0}$, then*

$$\phi(\beta) = \max_j \left\{ d_j + \phi(\beta - a_j) \mid \psi(\beta) = t_j + \psi(\beta - a_j),\ j = 1, \ldots, n_2 \right\}.$$

Proof. For any $\beta \in \cup_{j=1}^{n_2} B_j$ and any $j \in \{1, \ldots, n_2\}$ satisfying $\psi(\beta) = t_j + \psi(\beta - a_j)$,

$$\begin{aligned}
d_j + \phi(\beta - a_j) &= d_j + \max \left\{ d^T y \mid t^T y \geq \psi(\beta - a_j),\ Ay \leq \beta - a_j,\ y \in \mathbb{Z}_+^{n_2} \right\} \\
&= \max \left\{ d^T y + d_j \mid t^T y \geq \psi(\beta) - t_j,\ Ay \leq \beta - a_j,\ y \in \mathbb{Z}_+^{n_2} \right\} \\
&= \max \left\{ d^T (y + e_j) \mid t^T (y + e_j) \geq \psi(\beta),\ A(y + e_j) \leq \beta,\ y \in \mathbb{Z}_+^{n_2} \right\} \\
&= \max \left\{ d^T y \mid t^T y \geq \psi(\beta),\ Ay \leq \beta,\ y_j \geq 1,\ y \in \mathbb{Z}_+^{n_2} \right\} \\
&\leq \max \left\{ d^T y \mid t^T y \geq \psi(\beta),\ Ay \leq \beta,\ y \in \mathbb{Z}_+^{n_2} \right\} = \phi(\beta).
\end{aligned}$$

Hence

$$\phi(\beta) \geq \max_j \left\{ d_j + \phi(\beta - a_j) \mid \psi(\beta) = t_j + \psi(\beta - a_j),\ j = 1, \ldots, n_2 \right\}.$$

Let $\hat{y} \in opt(\beta)$ with $\hat{y} \neq \mathbf{0}$. We know that $\hat{y} \in \widehat{opt}(\beta)$. Suppose $\hat{y}_{j^*} > 0$ for $j^* \in \{1, \ldots, n_2\}$. Then $e_{j^*} \leq \hat{y}$ and by integer complementary slackness in Proposition 1, $\psi(\beta) = \psi(a_{j^*}) + \psi(\beta - a_{j^*}) = t_{j^*} + \psi(\beta - a_{j^*})$. Furthermore, by bilevel integer complementary slackness in Proposition 4, $\phi(\beta) = \phi(a_{j^*}) + \phi(\beta - a_{j^*}) = d_{j^*} + \phi(\beta - a_{j^*})$. Hence

$$\phi(\beta) \leq \max_j \left\{ d_j + \phi(\beta - a_j) \mid \psi(\beta) = t_j + \psi(\beta - a_j),\ j = 1, \ldots, n_2 \right\}.$$

Then, the result follows. □

4.3 Bilevel minimal vectors

In this section, we show that the BIP value function $\phi(\cdot)$ can be entirely characterized on the set of *bilevel minimal* vectors:

Definition 1. *A vector $\beta \in \mathbf{B}$ is bilevel minimal if $\phi(\beta - e_i) < \phi(\beta)$ for all $i \in \{1, \cdots, m_2\}$ such that $\psi(\beta - e_i) = \psi(\beta)$. Let $\bar{\mathbf{B}}$ be the set of all bilevel minimal vectors.*

Remark 2. *Any $\beta \in \mathbf{B}$ that is level-set minimal is also bilevel minimal, i.e., $\tilde{\mathbf{B}} \subseteq \bar{\mathbf{B}}$.*

Remark 3. *There are instances where some $\beta \in \mathbf{B}$ is bilevel minimal but not level-set minimal, i.e., $\tilde{\mathbf{B}} \not\supseteq \bar{\mathbf{B}}$. Consider the following example:*

$$\begin{aligned} \phi(\beta) = \max \quad & 3y_1 + 2y_2 + y_3 \\ \textit{subject to} \quad & y_1 + 2y_2 + 2y_3 \leq \beta_1, \\ & 2y_1 + 2y_2 + y_3 \leq \beta_2, \\ & y_1 + 2y_2 + 2y_3 \geq \psi(\beta),\ y \in \mathbb{Z}^3_+, \end{aligned}$$

and $\psi(\beta) = \max\left\{y_1 + 2y_2 + 2y_3 | y_1 + 2y_2 + 2y_3 \leq \beta_1, 2y_1 + 2y_2 + y_3 \leq \beta_2, y \in \mathbb{Z}^3_+\right\}$. Right-hand side vector $\beta = (2,2)^T$ is bilevel minimal as $\psi((1,2)^T) = 1 < \psi((2,2)^T) = 2$, $\psi((2,1)^T) = \psi((2,2)^T) = 2$ and $\phi((2,1)^T) = 1 < \phi((2,2)^T) = 2$. However, $\beta = (2,2)^T$ is clearly not level-set minimal since $\psi((2,1)^T) = \psi((2,2)^T)$.

Lemma 6 (*Monotonicity*)**.** *$\phi(\beta') \leq \phi(\beta)$ for $\beta, \beta' \in \mathbf{B}$ such that $\beta' \leq \beta$ and $\psi(\beta') = \psi(\beta)$.*

Proof. Let $\hat{y} \in opt(\beta')$. Then $A\hat{y} \leq \beta' \leq \beta$. Also, $t^T\hat{y} = \psi(\beta') = \psi(\beta)$. Thus $\hat{y} \in \widehat{opt}(\beta)$ and $d^T\hat{y} = \phi(\beta') \leq \phi(\beta)$. □

Proposition 6. *$\beta \in \bar{\mathbf{B}}$ if and only if $\phi(\beta') < \phi(\beta)$ for all $\beta' \leq \beta$, $\beta' \neq \beta$ such that $\psi(\beta') = \psi(\beta)$.*

Proof. $\Rightarrow$ Suppose there exists $\beta' \leq \beta$, $\beta' \neq \beta$ such that $\psi(\beta') = \psi(\beta)$ and $\phi(\beta') \geq \phi(\beta)$. Select $i \in \{1, \cdots, m_2\}$ such that $\beta_i > \beta'_i$ and consider the vector $\beta - e_i$. We have $\beta' \leq \beta - e_i \leq \beta$ and thus $\psi(\beta') \leq \psi(\beta - e_i) \leq \psi(\beta)$ due to the nondecreasing property of $\psi(\cdot)$. Then $\psi(\beta') = \psi(\beta - e_i)$ since $\psi(\beta') = \psi(\beta)$. Let $\hat{y} \in opt(\beta')$. Then $A\hat{y} \leq \beta' \leq \beta - e_i$. Also, $t^T\hat{y} = \psi(\beta') = \psi(\beta - e_i)$. Thus $\hat{y} \in \widehat{opt}(\beta - e_i)$ and $\phi(\beta') \leq \phi(\beta - e_i)$. Hence $\phi(\beta - e_i) \geq \phi(\beta)$ and by Lemma 6, $\phi(\beta - e_i) = \phi(\beta)$. This indicates that β is not bilevel minimal.

$\Leftarrow$ Suppose $\phi(\beta') < \phi(\beta)$ for all $\beta' \leq \beta$, $\beta' \neq \beta$ such that $\psi(\beta') = \psi(\beta)$. Then $\phi(\beta - e_i) < \phi(\beta)$ for all $i \in \{1, \cdots, m_2\}$ with $\psi(\beta - e_i) = \psi(\beta)$. Hence β is bilevel minimal by Definition 1. □

Lemma 7. *For any $\beta \in \mathbf{B} \setminus \bar{\mathbf{B}}$ there exists $\beta' \leq \beta$, $\beta' \neq \beta$ such that $\psi(\beta') = \psi(\beta)$ and $\phi(\beta') = \phi(\beta)$.*

Proof. By Proposition 6, for any $\beta \in \mathbf{B} \setminus \bar{\mathbf{B}}$ there exists $\beta' \leq \beta$ with $\beta' \neq \beta$ and $\psi(\beta') = \psi(\beta)$ such that $\phi(\beta') \geq \phi(\beta)$. We then have $\phi(\beta') = \phi(\beta)$ by Lemma 6. □

Lemma 8. *$A\hat{y} = \bar{\beta}$ for $\bar{\beta} \in \bar{\mathbf{B}}$ and $\hat{y} \in opt(\bar{\beta})$.*

Proof. Suppose $A\hat{y} \leq \bar{\beta}$ and $A\hat{y} \neq \bar{\beta}$. Then for any $A\hat{y} \leq \beta \leq \bar{\beta}$ we have $\phi(\beta) = \phi(\bar{\beta})$ by Lemma 4 and $\psi(\beta) = \psi(\bar{\beta})$ by Lemma 2, which contradicts the fact that $\bar{\beta} \in \bar{\mathbf{B}}$. □

Corollary 1. *For $\beta \in \mathbf{B}$, if there exists $\hat{y} \in opt(\beta)$ such that $A\hat{y} \neq \beta$, then $\beta \notin \bar{\mathbf{B}}$.*

Proposition 7. *If $\bar{\beta} \in \bar{\mathbf{B}}$ and $\hat{y} \in opt(\bar{\beta})$, then $Ay \in \bar{\mathbf{B}}$ for any $y \in \mathbb{Z}_+^{n_2}$ and $y \leq \hat{y}$.*

Proof. Suppose for some $\bar{\beta} \in \bar{\mathbf{B}}$ and $\hat{y} \in opt(\bar{\beta})$, there exists $y \in \mathbb{Z}_+^{n_2}$ and $y \leq \hat{y}$ but $Ay \notin \bar{\mathbf{B}}$. By Lemma 7 there exists $\beta' \leq Ay$ with $\beta' \neq Ay$ such that $\psi(\beta') = \psi(Ay)$ and $\phi(\beta') = \phi(Ay)$. Let $y' \in opt(\beta')$ and then $Ay' \leq \beta'$, $\psi(\beta') = t^T y'$ and $\phi(\beta') = d^T y'$. We also have $\phi(Ay) = d^T y$ by Proposition 4 and $\psi(Ay) = t^T y$ from integer complementary slackness in Proposition 1.

Consider the nonnegative vector $\tilde{y} = \hat{y} - y + y'$. We have $A\tilde{y} = A(\hat{y} - y + y') \leq \bar{\beta} - Ay + \beta' \leq \bar{\beta}$ and $A\tilde{y} \neq \bar{\beta}$ since $\beta' \leq Ay$ and $\beta' \neq Ay$. Also, $t^T\tilde{y} = t^T(\hat{y} - y + y') = \psi(\bar{\beta}) - \psi(Ay) + \psi(\beta') = \psi(\bar{\beta})$. Since $A\tilde{y} \leq \bar{\beta}$ and due to the nondecreasing property of $\psi(\cdot)$, $t^T\tilde{y} = \psi(A\tilde{y}) = \psi(\bar{\beta})$. Hence $\tilde{y} \in \widehat{opt}(A\tilde{y})$ and $\tilde{y} \in \widehat{opt}(\bar{\beta})$. In addition, $d^T\tilde{y} = d^T(\hat{y} - y + y') = \phi(\bar{\beta}) - \phi(Ay) + \phi(\beta') = \phi(\bar{\beta})$. Hence $\phi(A\tilde{y}) \geq \phi(\bar{\beta})$. By Lemma 6, $\phi(A\tilde{y}) = \phi(\bar{\beta})$. We thus have found a vector $A\tilde{y} \leq \bar{\beta}$ with $A\tilde{y} \neq \bar{\beta}$, $\psi(A\tilde{y}) = \psi(\bar{\beta})$ and $\phi(A\tilde{y}) = \phi(\bar{\beta})$. This contradicts the fact that $\bar{\beta} \in \bar{\mathbf{B}}$. □

Corollary 2. *For $\bar{\beta} \in \bar{\mathbf{B}}$ and $\hat{y} \in opt(\bar{\beta})$, if $\hat{y}_j \geq 1$ for $j \in \{1, \cdots, n_2\}$, then $a_j \in \bar{\mathbf{B}}$.*

As a result of Corollary 2, those columns in constraint matrix A that are not bilevel minimal can be eliminated. In particular, let $\mathcal{A}$ be the index set of columns in A that are bilevel minimal. For any $\beta \in \mathbb{Z}^{m_2}$, define

$$\phi'(\beta) = \max\left\{\sum_{j\in\mathcal{A}} d_j y_j \;\middle|\; \sum_{j\in\mathcal{A}} t_j y_j \geq \psi'(\beta),\ \sum_{j\in\mathcal{A}} a_j y_j \leq \beta,\ y \in \mathbb{Z}_+^{n_2}\right\},$$

where

$$\psi'(\beta) = \max\left\{\sum_{j\in\mathcal{A}} t_j y_j \;\middle|\; \sum_{j\in\mathcal{A}} a_j y_j \leq \beta,\ y \in \mathbb{Z}_+^{n_2}\right\}.$$

Let $S'(\beta) = \{y \in \mathbb{Z}_+^{n_2} \mid \sum_{j\in\mathcal{A}} a_j y_j \leq \beta\}$, and define $\phi'(\beta) = \psi'(\beta) = -\infty$ if $S'(\beta) = \emptyset$. Let $opt'(\beta) = \operatorname{argmax}\left\{\sum_{j\in\mathcal{A}} d_j y_j \mid \sum_{j\in\mathcal{A}} t_j y_j \geq \psi'(\beta),\ y \in S'(\beta)\right\}$.

Proposition 8. *$\psi(\bar{\beta}) = \psi'(\bar{\beta})$ and $\phi(\bar{\beta}) = \phi'(\bar{\beta})$ for $\bar{\beta} \in \bar{\mathbf{B}}$.*

Proof. Let $y' \in opt'(\bar{\beta})$ for $\bar{\beta} \in \bar{\mathbf{B}}$. Construct $\tilde{y}$ from y' by letting $\tilde{y}_j = y'_j$ for all $j \in \mathcal{A}$ and $\tilde{y}_j = 0$ for all $j \notin \mathcal{A}$. Then $\tilde{y} \in S(\bar{\beta})$ and $\psi(\bar{\beta}) \geq t^T\tilde{y} = \sum_{j\in\mathcal{A}} t_j y'_j = \psi'(\bar{\beta})$. Let $\hat{y} \in opt(\bar{\beta})$. By Corollary 2, $\hat{y}_j = 0$ for all $j \in \{1, \cdots, n_2\}$ and $j \notin \mathcal{A}$. Then $\hat{y} \in S'(\bar{\beta})$ and $\psi'(\bar{\beta}) \geq \sum_{j\in\mathcal{A}} t_j\hat{y}_j = \psi(\bar{\beta})$. Hence $\psi'(\bar{\beta}) = \psi(\bar{\beta})$. Note that $\hat{y}$ is an optimal solution to $\psi'(\bar{\beta})$ and thus $\phi'(\bar{\beta}) \geq \sum_{j\in\mathcal{A}} d_j\hat{y}_j = \phi(\bar{\beta})$. Moreover, $\tilde{y}$ is an optimal solution to $\psi(\bar{\beta})$ and thus $\phi(\bar{\beta}) \geq d^T\tilde{y} = \sum_{j\in\mathcal{A}} d_j y'_j = \phi'(\bar{\beta})$. Hence $\phi(\bar{\beta}) = \phi'(\bar{\beta})$. □

4.4 Integral monoids

We introduce a superset of the bilevel minimal vector set $\bar{\mathbf{B}}$ that is easier to generate and maintains many of the desirable properties. An *integral monoid* [25] is a set of vectors in $\mathbb{Z}^m$ which forms a semi-group under addition. Let M be the integral monoid generated by nonnegative integer linear combinations of the columns of A, i.e., $M = \{Ay \mid y \in \mathbb{Z}^{n_2}_+\}$. The *truncated integral monoid* is defined as $\bar{M} = M \cap \mathbf{B}$.

Remark 4. *The set of bilevel minimal vectors* $\bar{\mathbf{B}} \subseteq \bar{M}$.

Theorem 4. *[38] For any* $\beta \in \mathbf{B}$,

$$\psi(\beta) = \max_{\pi\in\bar{M}:\pi\leq\beta} \psi(\pi).$$

Theorem 5. *For any* $\beta \in \mathbf{B}$,

$$\phi(\beta) = \max_{\pi\in\bar{M}:\pi\leq\beta,\psi(\pi)=\psi(\beta)} \phi(\pi).$$

Proof. For any $\beta \in \mathbf{B}$ and $\pi \in \bar{M}$ satisfying $\pi \leq \beta$ and $\psi(\pi) = \psi(\beta)$,

$$\begin{aligned}
\phi(\beta) &= \max\left\{d^T y \mid t^T y \geq \psi(\beta),\ Ay \leq \beta,\ y \in \mathbb{Z}^{n_2}_+\right\} \\
&= \max\left\{d^T y \mid t^T y \geq \psi(\pi),\ Ay \leq \beta,\ y \in \mathbb{Z}^{n_2}_+\right\} \\
&\geq \max\left\{d^T y \mid t^T y \geq \psi(\pi),\ Ay \leq \pi,\ y \in \mathbb{Z}^{n_2}_+\right\} = \phi(\pi).
\end{aligned}$$

Hence

$$\phi(\beta) \geq \max_{\pi\in\bar{M}:\pi\leq\beta,\psi(\pi)=\psi(\beta)} \phi(\pi).$$

Let $\hat{y} \in opt(\beta)$. Then $\phi(\beta) = \phi(A\hat{y})$ by Lemma 4, and $\psi(\beta) = \psi(A\hat{y})$ by Lemma 2. In addition, $A\hat{y} \in \bar{M}$ and $A\hat{y} \leq \beta$. Hence

$$\phi(\beta) = \phi(A\hat{y}) \leq \max_{\pi\in\bar{M}:\pi\leq\beta,\psi(\pi)=\psi(\beta)} \phi(\pi).$$

The result then follows. □

5 Constructing the BIP value functions

Utilizing the properties derived in Section 4, we first design a dynamic programming (DP) algorithm for constructing the BIP value function $\phi(\cdot)$ over $\mathbf{B}$. This algorithm does not require solving any IP. However, it is memory intensive as it stores $\phi(\cdot)$ and $\psi(\cdot)$ for all right-hand sides in $\mathbf{B}$. Thus, it can only be used when the number of rows in the follower's problem is relatively small. To solve problems with larger number of rows, we propose a two-step approach to generate $\phi(\cdot)$ over the truncated integral monoid set $\bar{M}$. Finally, we present an algorithm for computing the BIP value function when the lower-level variables y are all binary. The constructed BIP value functions and the truncated integral monoid $\bar{M}$ are then used in the local search procedure of the proposed B&C algorithm.

5.1 Constructing the BIP value function using DP

We propose a dynamic programming approach for finding the BIP value function $\phi(\cdot)$. This approach, however, only applies to the case when the constraint matrix A is nonnegative. We thus assume that A is nonnegative in this section. Without loss of generality, we assume that constraint matrix B is also nonnegative, and let $\mathbf{B} = \{\beta \in \mathbb{Z}^{m_2} \mid \mathbf{0} \leq \beta \leq b\}$. The proposed approach is presented in Algorithm 2. For $\beta \in \mathbf{B}$, $l(\beta)$ is the lower bound of $\psi(\beta)$, and $y(\beta)$ is a feasible solution to the follower's problem with $l(\beta) = t^T y(\beta)$. Moreover, $u(\beta)$ is the leader's benefit from $y(\beta)$, i.e., $u(\beta) = d^T y(\beta)$. The algorithm terminates when $u(\beta) = \phi(\beta)$ for all $\beta \in \mathbf{B}$.

Algorithm 2. *A DP algorithm to evaluate $\phi(\beta)$ for all $\beta \in \mathbf{B}$ when $\mathbb{Y} = \mathbb{Z}_+^{n_2}$.*

Step 0: Initialize $l^0(\beta) = 0$, $y^0(\beta) = 0$ and $u^0(\beta) = 0$ for all $\beta \in \mathbf{B}$. For $j = 1, \ldots, n_2$, if $a_j \in \mathbf{B}$, set $l^0(a_j) = t_j$, $y^0(a_j) = e_j$ and $u^0(a_j) = d_j$, and insert a_j into a vector list $\mathcal{L}$. Set $l^1(\beta) = l^0(\beta)$, $y^1(\beta) = y^0(\beta)$ and $u^1(\beta) = u^0(\beta)$ for all $\beta \in \mathbf{B}$. Denote the i^{th} element of vector β by β_i. Set $k \leftarrow 1$.

Step 1: Denote the k^{th} vector in $\mathcal{L}$ by β^k and let $\beta = \beta^k$. Update all $\beta' \geq \beta^k$, $\beta' \in \mathbf{B}$ with the following lexicographic order:

(1a) Set $\beta_1 \leftarrow \beta_1 + 1$.

If $l^k(\beta) < l^k(\beta^k) + l^k(\beta - \beta^k)$, set $l^k(\beta) = l^k(\beta^k) + l^k(\beta - \beta^k)$, $y^k(\beta) = y^k(\beta^k) + y^k(\beta - \beta^k)$ and $u^k(\beta) = u^k(\beta^k) + u^k(\beta - \beta^k) = d^T y^k(\beta)$;

else if $l^k(\beta) = l^k(\beta^k) + l^k(\beta - \beta^k)$ and $u^k(\beta) < u^k(\beta^k) + u^k(\beta - \beta^k)$, set $y^k(\beta) = y^k(\beta^k) + y^k(\beta - \beta^k)$ and $u^k(\beta) = u^k(\beta^k) + u^k(\beta - \beta^k) = d^T y^k(\beta)$.

(1b) If $\beta_1 \geq b_1$, go to Step (1c); otherwise, go to Step (1a).

(1c) If $\beta_i \geq b_i$ for all $i = 1, \ldots, m_2$, go to Step 2. Otherwise, let $s = \min\{i : \beta_i < b_i\}$. Set $\beta_i \leftarrow \beta_i^k$ for $i = 1, \ldots, s-1$. Set $\beta_s \leftarrow \beta_s + 1$ and go to Step (1a).

Step 2: If $k = |\mathcal{L}|$, terminate with solution $\phi(\cdot) = u^k(\cdot)$. Otherwise, set $l^{k+1}(\beta) \leftarrow l^k(\beta)$, $y^{k+1}(\beta) \leftarrow y^k(\beta)$ and $u^{k+1}(\beta) \leftarrow u^k(\beta)$ for all $\beta \in \mathbf{B}$. Set $k \leftarrow k+1$ and go to Step 1.

Theorem 6. *Algorithm 2 terminates with optimal values to $\phi(\beta)$ for all $\beta \in \mathbf{B}$ in at most n_2 iterations.*

Proof. Let $k^* = |\mathcal{L}|$. The algorithm then terminates at iteration k^*. By Theorem 4 in [27], $\psi(\beta) = l^{k^*}(\beta) = t^T y^{k^*}(\beta)$ and $y^{k^*}(\beta)$ is an optimal solution to $\psi(\beta)$ for all $\beta \in \mathbf{B}$. For $\beta \in \mathbf{B} \setminus \cup_{j=1}^{n_2} B_j$, we initialize $u^0(\beta) = 0$ in Step 0 and do not update it subsequently. By Lemma 5, $\phi(\beta) = 0$ for any $\beta \in \mathbf{B} \setminus \cup_{j=1}^{n_2} B_j$.

For $\beta \in \cup_{j=1}^{n} B_j$, $u^{k^*}(\beta) \leq \phi(\beta)$ because $y^{k^*}(\beta)$ is optimal to $\psi(\beta)$ and $u^{k^*}(\beta) = d^T y^{k^*}(\beta)$. Suppose there exists $\beta \in \cup_{j=1}^{n} B_j$ such that $u^{k^*}(\beta) < \phi(\beta)$ and $u^{k^*}(\beta') = \phi(\beta')$ for all $\beta' \leq \beta$, $\beta' \neq \beta$ and $\beta' \in \cup_{j=1}^{n} B_j$. It follows from Proposition 5 that there exists $j^* \in \{1, \ldots, n_2\}$, such that $u^{k^*}(\beta) < d_{j^*} + \phi(\beta - a_{j^*})$ and $\psi(\beta) = t_{j^*} + \psi(\beta - a_{j^*})$. Note that $\psi(a_{j^*}) = t_{j^*}$ from integer complementary slackness. Since $u^{k^*}(a_{j^*}) \geq d_{j^*}$ and $u^{k^*}(\beta - a_{j^*}) = \phi(\beta - a_{j^*})$, it follows that $u^{k^*}(a_{j^*}) + u^{k^*}(\beta - a_{j^*}) \geq d_{j^*} + \phi(\beta - a_{j^*}) > u^{k^*}(\beta)$. In addition, $l^{k^*}(\beta) = \psi(\beta) = \psi(a_{j^*}) + \psi(\beta - a_{j^*}) = l^{k^*}(a_{j^*}) + l^{k^*}(\beta - a_{j^*})$. This leads to a contradiction because if $l^{k^*}(\beta) = l^{k^*}(a_{j^*}) + l^{k^*}(\beta - a_{j^*})$ and $u^{k^*}(\beta) < u^{k^*}(a_{j^*}) + u^{k^*}(\beta - a_{j^*})$, then $u^{k^*}(\beta)$ should have been updated to $u^{k^*}(a_{j^*}) + u^{k^*}(\beta - a_{j^*})$ in Step (1a). Hence, $u^{k^*}(\beta) = \phi(\beta)$ for all $\beta \in \cup_{j=1}^{n} B_j$. The result then follows from $k^* = |\mathcal{L}| \leq n_2$. □

Step 1 of Algorithm 2 requires $O(|\mathbf{B}|)$ calculations and the overall running time of the algorithm is $O(n_2|\mathbf{B}|)$.

5.2 Constructing the BIP value function using integral monoids

We present a two-step approach to generate $\phi(\cdot)$ over the truncated integral monoid set $\bar{M}$. First, Procedure 1 generates the truncated integral monoid $\bar{M}$ and simultaneously calculates

$\psi(\pi)$ for $\pi \in \bar{M}$. The generated monoid elements are stored in set $\mathcal{M}$. Then, Procedure 2 computes the BIP value function $\phi(\pi)$ for $\pi \in \mathcal{M}$. As we assume that the feasible region of the lower-level problem (1c) is bounded for any upper-level decision $x \in \mathbb{B}^{n_1}$, there is an upper bound u_j on each lower-level variable y_j, $j = 1, \cdots, n_2$. Also, let π_{max} be the maximum right-hand side vector that $(b - Bx)$ can take for any $x \in \mathbb{B}^{n_1}$.

Procedure 1. *Generate truncated integral monoid $\bar{M}$ and compute $\psi(\cdot)$ over $\bar{M}$.*

1: **Generate Monoid** (j, τ)
2: **for** $\mu = u_j$ to 0 **do**
3: Construct vector $\pi = \tau + \mu a_j$.
4: **If** $\pi \leq \pi_{max}$ **then**
5: **If** $j < n_2$ **then**
6: Call **Generate Monoid** $(j + 1, \pi)$.
7: **else**
8: **If** $\pi \notin \mathcal{M}$ **then**
9: Solve IP associated with $\psi(\pi)$ to obtain optimal solution $\hat{y}$.
10: Store π in $\mathcal{M}$ and $\psi(\pi) = t^T \hat{y}$.
11: **for all** $y \in \mathbb{Z}_+^{n_2}$ such that $y \leq \hat{y}$ and $y \neq \hat{y}$ **do**
12: Compute monoid element Ay.
13: **If** $Ay \notin \mathcal{M}$ **then** store Ay in $\mathcal{M}$ and $\psi(Ay) = t^T y$.

After initialization with $j = 1$, $\tau = \mathbf{0}$ and $\mathcal{M} = \emptyset$, Procedure 1, which is first described by Trapp et al. [38], recursively generates monoid elements using integer linear combinations of the columns of constraint matrix A (Lines 1-6). When a new monoid element π that is not already contained in $\mathcal{M}$ is generated, we solve the IP associated with $\psi(\pi)$ to get an optimal solution $\hat{y}$ (Lines 8-10). After we store π and the value $\psi(\pi)$ in $\mathcal{M}$, we leverage the integer complementary slackness property in Proposition 1 to discover additional monoid elements $\pi' \leq \pi$ for which $\psi(\pi')$ is easily obtainable (Lines 11-13). Unlike Trapp et al. [38], we do not eliminate a monoid element that is not level-set minimal in Procedure 1 because that monoid element can still be bilevel minimal as illustrated in Remark 3.

Procedure 2. *Compute $\phi(\cdot)$ over $\bar{M}$.*

1: Initialize $\phi(\pi) = -\infty$ for all $\pi \in \mathcal{M}$.

2: **for all** $\pi \in \mathcal{M}$ **do**
3: **If** $\phi(\pi) = -\infty$ **then**
4: Solve the IP associated with $\phi(\pi)$ to obtain optimal solution $\hat{y}$.
5: **If** $A\hat{y} \neq \pi$ **then**
6: Delete π from $\mathcal{M}$. (Corollary 1)
7: **for all** $y \in \mathbb{Z}_+^{n_2}$ such that $y \leq \hat{y}$ and $y \neq \hat{y}$ **do**
8: Compute monoid element Ay.
9: Set $\phi(Ay) = d^T y$. (Proposition 4)

Procedure 2 iterates over all π in $\bar{M}$. If $\phi(\pi)$ has not been updated (i.e., $\phi(\pi) = -\infty$), we solve the IP associated with $\phi(\pi)$ to get an optimal solution $\hat{y}$. We can solve that IP because the follower's value function $\psi(\pi)$ is already computed in Procedure 1. If $A\hat{y} \neq \pi$, π is not bilevel minimal and thus it can be eliminated from $\bar{M}$ by Corollary 1. Finally, we leverage bilevel integer complementary slackness result in Proposition 4 to set $\phi(\pi')$ for $\pi' \leq \pi$, $\pi' \neq \pi$ and $\pi' \in \bar{M}$ without solving their associated IPs.

5.3 Constructing the BIP value function when lower-level variables are binary

We present a DP algorithm to construct the BIP value function when the lower-level variables in $DBLP$ are all binary, i.e., $\mathbb{Y} = \mathbb{B}^{n_2}$. We assume that constraint matrices A and B are nonnegative. For $k = 1, \ldots, n_2$, define the following partial value functions:

$$\overline{\phi}_k(\beta) = \max \left\{ \sum_{j=1}^{k} d_j y_j \;\middle|\; \sum_{j=1}^{k} t_j y_j \geq \overline{\psi}_k(\beta),\ \sum_{j=1}^{k} a_j y_j \leq \beta,\ y \in \{0,1\}^k \right\},\ \beta \in \mathbf{B}, \tag{11}$$

where $\mathbf{B}$ is defined as in Section 5.1 and

$$\overline{\psi}_k(\beta) = \max \left\{ \sum_{j=1}^{k} t_j y_j \;\middle|\; \sum_{j=1}^{k} a_j y_j \leq \beta,\ y \in \{0,1\}^k \right\},\ \beta \in \mathbf{B}.$$

Note that $\overline{\phi}_{n_2}(\beta)$ denotes the leader's benefit from the optimal response of the follower for $\beta \in \mathbf{B}$. Algorithm 3 is motivated by the DP algorithms described in Brotcorne at al. [7] and Özaltın et al. [33].

Algorithm 3. *A DP algorithm to evaluate $\phi(\beta)$ for all $\beta \in \mathbf{B}$ when $\mathbb{Y} = \mathbb{B}^{n_2}$.*

Step 0: Initialize $\overline{\psi}_0(\beta) = 0$ and $\overline{\phi}_0(\beta) = 0$ for all $\beta \in \mathbf{B}$. Denote the i^{th} element of vector β by β_i. Set $\overline{\psi}_1(\beta) \leftarrow \overline{\psi}_0(\beta)$ and $\overline{\phi}_1(\beta) \leftarrow \overline{\phi}_0(\beta)$ for all $\beta \in \mathbf{B}$. Set $k \leftarrow 1$.

Step 1: Let $\beta = a_k$. Update all $\beta' \geq a_k$, $\beta' \in \mathbf{B}$ with the following lexicographic order:

(1a) **If** $\overline{\psi}_k(\beta) < \overline{\psi}_{k-1}(\beta - a_k) + t_k$, set $\overline{\psi}_k(\beta) \leftarrow \overline{\psi}_{k-1}(\beta - a_k) + t_k$ and $\overline{\phi}_k(\beta) \leftarrow \overline{\phi}_{k-1}(\beta - a_k) + d_k$;
else if $\overline{\psi}_k(\beta) = \overline{\psi}_{k-1}(\beta - a_k) + t_k$ and $\overline{\phi}_k(\beta) < \overline{\phi}_{k-1}(\beta - a_k) + d_k$, set $\overline{\phi}_k(\beta) \leftarrow \overline{\phi}_{k-1}(\beta - a_k) + d_k$.

(1b) Set $\beta_1 \leftarrow \beta_1 + 1$. If $\beta_1 \geq b_1$, go to Step (1c); otherwise, go to Step (1a).

(1c) If $\beta_i \geq b_i$ for all $i = 1, \ldots, m_2$, go to Step 2. Otherwise, let $s = \min\{i : \beta_i < b_i\}$. Set $\beta_i \leftarrow \beta_i^k$ for $i = 1, \ldots, s-1$. Set $\beta_s \leftarrow \beta_s + 1$ and go to Step (1a).

Step 2: If $k = n_2$, terminate with solution $\phi(\beta) = \overline{\phi}_{n_2}(\beta)$ for all $\beta \in \mathbf{B}$. Otherwise, $\overline{\psi}_{k+1}(\beta) \leftarrow \overline{\psi}_k(\beta)$ and $\overline{\phi}_{k+1}(\beta) \leftarrow \overline{\phi}_k(\beta)$ for all $\beta \in \mathbf{B}$. Set $k \leftarrow k+1$ and go to Step 1.

Theorem 7. *Algorithm 3 terminates with optimal values to $\phi(\beta)$ for all $\beta \in \mathbf{B}$ in at most n_2 iterations.*

Step 1 of Algorithm 3 requires $O(|\mathbf{B}|)$ calculations and the overall running time of the algorithm is $O(n_2|\mathbf{B}|)$. The major difference between Algorithms 2 and 3 is that at each iteration k, Algorithm 2 updates the lower bound $l(\beta)$ of $\psi(\beta)$ for all $\beta \geq a_k$, $\beta \in \mathbf{B}$ by using the supperadditivity of $\psi(\cdot)$. Thus, Algorithm 2 is not directly applicable when $\mathbb{Y} = \mathbb{B}^{n_2}$ because the value function of a binary IP is not superadditive. Algorithm 3 computes the exact value of $\overline{\psi}_k(\beta)$ for all $\beta \geq a_k$, $\beta \in \mathbf{B}$ by utilizing the dynamic programming recursion $\overline{\psi}_k(\beta) = \max\{\overline{\psi}_{k-1}(\beta), \overline{\psi}_{k-1}(\beta - a_k) + t_k\}$.

6 Computational experiments

We demonstrate the performance of our algorithms on a set of randomly generated instances. The algorithms are implemented in C++, compiled using Microsoft Visual Studio 2015 and tested on a desktop computer equipped with a single 3.4 GHz CPU, 24 GB of RAM and CPLEX 12.7.1. A 3-hour runtime is allowed for each instance and solution times are reported in seconds.

Table 1: Characteristics of instance classes in Testbeds 1 and 2.

Tm-K	m_2	T1-K (Testbed 1)					T2-K (Testbed 2)				
		n_1	n_2	A	B	b	n_1	n_2	A	B	b
Tm-1	6	50	70	[5,9]	[3,7]	[21,26]	70	90	[4,8]	[1,7]	[21,26]
Tm-2	6	60	80	[5,9]	[3,7]	[22,27]	75	95	[4,8]	[1,7]	[22,27]
Tm-3	6	70	90	[5,9]	[3,7]	[23,28]	80	100	[4,8]	[1,7]	[23,28]
Tm-4	6	80	100	[5,9]	[3,7]	[24,29]	85	105	[4,8]	[1,7]	[24,29]
Tm-5	6	90	110	[5,9]	[3,7]	[25,30]	90	110	[4,8]	[1,7]	[25,30]
Tm-6	7	50	70	[3,7]	[3,7]	[13,18]	70	90	[2,6]	[1,5]	[14,18]
Tm-7	7	60	80	[3,7]	[3,7]	[14,19]	75	95	[2,6]	[1,5]	[15,19]
Tm-8	7	70	90	[3,7]	[3,7]	[15,20]	80	100	[2,6]	[1,5]	[16,20]
Tm-9	7	80	100	[3,7]	[3,7]	[16,21]	85	105	[2,6]	[1,5]	[17,21]
Tm-10	7	90	110	[3,7]	[3,7]	[17,22]	90	110	[2,6]	[1,5]	[18,22]

Table 2: Characteristics of instance classes in Testbed 3.

T3-K	m_2	n_1	n_2	A	B	b
T3-1	20	70	90	[3,8]	[3,8]	[11,17]
T3-2	20	80	100	[3,8]	[3,8]	[12,18]
T3-3	20	90	110	[4,8]	[3,8]	[13,19]
T3-4	50	70	90	[3,8]	[3,8]	[15,21]
T3-5	50	80	100	[3,8]	[3,8]	[16,22]
T3-6	50	90	110	[4,8]	[3,8]	[16,22]
T3-7	100	70	90	[3,8]	[3,8]	[16,23]
T3-8	100	80	100	[3,8]	[3,8]	[17,24]
T3-9	100	90	110	[4,8]	[3,8]	[18,25]
T3-10	150	70	90	[3,8]	[3,8]	[16,24]
T3-11	150	80	100	[3,8]	[3,8]	[17,25]
T3-12	150	90	110	[3,8]	[3,8]	[18,26]

6.1 Experiment design

We randomly generate three different testbeds, of which testbeds 1 and 2 are generated for the DBLP with pure integer and pure binary lower-level variables, respectively. Testbed 3 is also generated for the DBLP with pure integer lower-level variables but the instances in Testbed 3 contain more constraints than the instances in Testbeds 1 and 2. There are 10 different instance classes in Testbeds 1 and 2 as shown in Table 1. Testbed 3 contains 12 different instance classes as shown in Table 2.

The instance classes are named Tm-K, where $m = 1, 2, 3$ is the testbed index and $K = 1, \cdots, 12$ is the instance class index. The numbers listed under A, B and b in Tables 1 and 2 are the lower and upper bounds of the uniform distribution that is used to generate nonzero elements of these parameters. We set the densities of constraint matrices A and B to 0.4, which is modeled by a Bernoulli distribution. Deterministic parameters c, d and t are generated from uniform distribution $U[1, 10]$. We randomly generate 3 instances for each instance class in Tables 1 and 2. The instances are named Tm-K-X, where $X = 1, 2, 3$ is the instance index. Our instances are available online [44].

We apply column elimination to reduce the size of the A matrix before constructing the value functions of instances in Testbeds 1 and 3. Specifically, we eliminate column j if $\psi(a_j) > t_j$ using the third result in Proposition 1. In addition, we eliminate column j if $a_j \notin \bar{\mathbf{B}}$ based on Corollary 2. For Testbed 1 and Testbed 2, we use Algorithm 2 and Algorithm 3, respectively, to construct $\phi(\cdot)$ over $\mathbf{B}$. For Testbed 3, we first execute Procedure 1 to generate the truncated integral monoid $\bar{M}$ and the follower's value function $\psi(\cdot)$ over $\bar{M}$. We then use Procedure 2 to construct the value function $\phi(\cdot)$ over $\bar{M}$.

Note that using Algorithm 2 to construct $\phi(\cdot)$ over $\mathbf{B}$ is only possible when the number of rows is at most 7 in Testbed 1. Algorithm 2 would require prohibitively large memory allocations for Testbed 3, because the instances in Testbed 3 have up to 150 rows. Using Procedures 1 and 2 to construct $\phi(\cdot)$ over $\bar{M}$, however, is considerably slower than Algorithm 2 for Testbed 1, because these procedures involve solving IPs whereas Algorithm 2 does not. As a result, we recommend using Algorithm 2 to construct $\phi(\cdot)$ when the number of rows in the follower's problem is relatively small. Otherwise, Procedures 1 and 2 should be used.

After constructing $\phi(\cdot)$, the B&C algorithm returns the optimal upper-level solution. The objective function value of a given upper-level solution $\bar{x}$ equals $c^T\bar{x}+\phi(b-B\bar{x})$. For Testbed 3, we compute $\psi(b-B\bar{x})$ and $\phi(b-B\bar{x})$ using Theorems 4 and 5, respectively. For Testbed 1 and Testbed 3, we set the neighborhood distance $k = 4$ in the local search procedure. We set $k = 2$ for Testbed 2. These values are chosen based on our computational observations.

6.2 Computational results

Tables 3, 4 and 5 report our computational results for instances in Testbeds 1, 2 and 3, respectively. Columns t_{BC} and t_{total} in these tables display the B&C algorithm solution time and the total solution time, respectively. Column t_{vf} in Tables 3 and 4 shows the value function computing time. Column t_{ce} in Table 5 presents the time spent in column elimination. We do not report t_{ce} in Table 3 because t_{ce} is less than 1 second for all the instances in Testbed 1. Column t_{mg} in Table 5 gives the total time used by Procedures 1 and 2 to generate the truncated integral monoid $\bar{M}$ and the associated value functions $\psi(\cdot)$ and $\phi(\cdot)$ over $\bar{M}$. The optimal objective values of all the instances can be found online [44].

In Tables 3 and 4, t_{vf} increases as n_2 or the magnitude of the right-hand side vector b increases because the running times of Algorithm 2 and Algorithm 3 are both $O(n_2|\mathbf{B}|)$. Table 5 shows that t_{ce} increases as the number of rows m_2 increases. This is also expected because when there are more rows in the constraint matrix A, we need to solve more IPs

Table 3: Computational results for instances in Testbed 1.

T2-K	Instance 1				Instance 2				Instance 3			
	$\|\mathbf{B}\|$	t_{vf}	t_{BC}	t_{total}	$\|\mathbf{B}\|$	t_{vf}	t_{BC}	t_{total}	$\|\mathbf{B}\|$	t_{vf}	t_{BC}	t_{total}
T1-1	1.9E8	6	0	6	2.4E8	8	12	20	1.7E8	4	184	189
T1-2	2.8E8	7	48	54	2.9E8	10	2	12	2.7E8	8	23	31
T1-3	4.0E8	16	42	59	3.7E8	10	23	33	3.5E8	10	21	32
T1-4	4.8E8	14	5	19	3.6E8	12	204	217	4.3E8	17	2419	2437
T1-5	4.9E8	24	433	457	4.8E8	17	72	89	6.8E8	26	14	40
T1-6	3.5E8	11	1	12	2.4E8	9	469	478	2.4E8	9	1	10
T1-7	5.9E8	18	1	19	5.1E8	14	2491	2506	3.6E8	12	3	15
T1-8	5.3E8	18	140	158	1.0E9	39	8	48	7.3E8	40	41	81
T1-9	1.3E9	82	274	356	1.6E9	107	1845	1952	8.2E8	32	89	121
T1-10	1.4E9	52	9	62	1.4E9	69	704	774	1.5E9	-	-	-*

* 3-hour runtime limit is exceeded.

Table 4: Computational results for instances in Testbed 2.

T2-K	Instance 1				Instance 2				Instance 3			
	$\|\mathbf{B}\|$	t_{vf}	t_{BC}	t_{total}	$\|\mathbf{B}\|$	t_{vf}	t_{BC}	t_{total}	$\|\mathbf{B}\|$	t_{vf}	t_{BC}	t_{total}
T2-1	1.7E8	26	0	27	1.7E8	25	123	148	1.8E8	26	8291	8317
T2-2	3.2E8	48	49	97	2.3E8	34	48	82	2.1E8	32	231	263
T2-3	2.7E8	-	-	-*	4.1E8	70	135	205	3.6E8	61	0	61
T2-4	3.6E8	62	1	63	4.4E8	77	2148	2225	4.6E8	84	4039	4124
T2-5	4.6E8	86	12	98	6.5E8	122	5233	5355	3.5E8	63	3086	3149
T2-6	4.5E8	64	1272	1336	3.3E8	58	0	58	4.7E8	75	1	76
T2-7	4.0E8	62	0	62	7.1E8	112	0	112	6.0E8	94	475	570
T2-8	8.7E8	165	517	682	8.8E8	148	7	156	7.5E8	125	1	126
T2-9	1.7E9	343	95	438	1.3E9	-	-	-*	1.3E9	256	1815	2071
T2-10	1.5E9	321	6220	6540	2.0E9	393	4	397	1.5E9	307	0	308

* 3-hour runtime limit is exceeded.

Table 5: Computational results for instances in Testbed 3.

T3-K	Instance 1				Instance 2				Instance 3			
	t_{ce}	t_{mg}	t_{BC}	t_{total}	t_{ce}	t_{mg}	t_{BC}	t_{total}	t_{ce}	t_{mg}	t_{BC}	t_{total}
T3-1	2	412	32	445	1	1631	193	1824	1	71	4	76
T3-2	1	972	141	1114	1*	14066	628	14695	1	1689	172	1862
T3-3	1	3904	959	4864	1	1289	731	2021	1	1462	308	1771
T3-4	2	126	151	279	2	541	1427	1970	2	137	440	578
T3-5	3	649	6101	6752	2	676	4791	5469	2	895	6472	7369
T3-6	3	261	3723	3986	3	142	1864	2008	3	245	1253	1501
T3-7	5	82	121	208	4	56	53	113	4	62	53	119
T3-8	5	226	883	1114	5	350	2372	2727	5	193	648	846
T3-9	6	209	2631	2845	6	193	2266	2465	6	429	7132	7567
T3-10	7	42	19	68	7	46	22	75	7	49	20	76
T3-11	9	161	255	424	9	148	219	376	9	139	230	378
T3-12	10	662	4171	4842	9	724	5054	5788	10	413	1769	2192

* We let this instance run more than 3 hours.

Table 6: More details on monoid generation for instances in Testbed 3.

T3-K	Instance 1				Instance 2				Instance 3			
	t_{P1}	t_{P2}	$\lvert\bar{M}\rvert$	n_{IP}	t_{P1}	t_{P2}	$\lvert\bar{M}\rvert$	n_{IP}	t_{P1}	t_{P2}	$\lvert\bar{M}\rvert$	n_{IP}
T3-1	327	85	5.4E4	5.5E4	963	668	2.1E5	1.8E5	50	21	3.8E4	2.7E4
T3-2	554	418	1.8E5	1.7E5	7065	7001	6.3E5	5.8E5	918	771	2.1E5	2.3E5
T3-3	2521	1383	3.0E5	4.0E5	764	525	1.6E5	2.1E5	868	594	1.9E5	1.9E5
T3-4	60	66	1.5E5	1.2E5	258	283	4.4E5	3.3E5	67	70	1.6E5	1.3E5
T3-5	298	351	4.8E5	3.6E5	316	360	5.2E5	4.0E5	414	482	6.3E5	4.4E5
T3-6	123	138	2.4E5	2.2E5	69	73	1.5E5	1.3E5	117	128	2.2E5	2.0E5
T3-7	39	43	7.7E4	7.7E4	27	29	5.2E4	5.2E4	30	32	5.7E4	5.8E4
T3-8	108	117	1.9E5	1.9E5	171	179	3.1E5	3.0E5	92	101	1.7E5	1.7E5
T3-9	101	108	1.7E5	1.7E5	93	100	1.5E5	1.6E5	208	222	3.2E5	3.5E5
T3-10	20	22	2.8E4	3.4E4	21	24	2.9E4	3.7E4	23	26	3.2E4	3.9E4
T3-11	75	86	1.1E5	1.1E5	69	79	1.0E5	1.0E5	65	74	9.7E4	1.0E5
T3-12	286	375	3.3E5	3.7E5	310	414	3.7E5	3.9E5	184	229	2.3E5	2.5E5

to check if a column a_j is bilevel minimal according to Definition 1. The B&C algorithm solution time t_{BC} in Tables 3 and 4, however, does not exhibit a regular pattern and varies significantly even across the instances in the same instance class. For example, in instance class T1-4 in Table 3, t_{BC} is 5 seconds for T1-4-1 but 2419 seconds for T1-4-3.

Table 6 presents more details on truncated integral monoid generation for Testbed 3. In particular, columns t_{P1} and t_{P2} report the time spent in Procedures 1 and 2, respectively. Note that t_{mg} in Table 5 is equal to $t_{P1} + t_{P2}$. Column $|\bar{M}|$ in Table 6 shows the size of the truncated integral monoid $\bar{M}$ after Procedure 2 is executed. Note that $|\bar{M}|$ may be smaller than the number of monoid elements generated in Procedure 1 because some of the monoid elements may be deleted from $\bar{M}$ in lines 5 and 6 of Procedure 2. Column n_{IP} in Table 6 gives the total number of IPs solved in Procedures 1 and 2, not including the number of IPs solved in column elimination. We observe from Table 6 that $|\bar{M}|$ is rather small compared to $|\mathbf{B}|$ in Tables 3 and 4.

7 Conclusions

We study a class of discrete bilevel linear programming problems in which the upper-level variables are all binary and the lower-level variables are all integer or all binary. The main contribution of this article is two-fold. First, we derive some theoretical properties of the value functions of bilevel integer programs. Most notably, we generalize the integer complementary slackness theorem to bilevel integer programs. This is an important theoretical result on its own. However, it also has computational significance because complementary slackness can be leveraged when constructing the value functions of bilevel integer programs

to easily obtain the objective values for some of the right-hand sides. We also introduce the set of bilevel minimal right-hand side vectors $\bar{\mathbf{B}}$ and show that the BIP value functions can be described by using only the elements in $\bar{\mathbf{B}}$. We further investigate the truncated integral monoid $\bar{M} \supseteq \bar{\mathbf{B}}$ as membership in $\bar{\mathbf{B}}$ is difficult to verify.

Second, utilizing the derived properties, we develop two efficient dynamic programming algorithms to build the BIP value functions when there are at most seven constraints in the lower-level. For problems with more constraints, we provide two procedures to generate the truncated integral monoid $\bar{M}$ and the associated value functions. Finally, given the BIP value functions, we propose a B&C algorithm to find the optimal upper-level solution. The computational results show that we can solve instances of this class with up to 200 variables and 150 constraints in a reasonable amount time.

The solution approach presented in this paper may be extended for solving bilevel programs with mixed-integer upper- and lower-level variables. Furthermore, our approach can be generalized to solve discrete bilevel linear programs with stochastic right-hand sides in the lower-level problem. We will also consider discrete bilevel nonlinear programs with quadratic or fractional objective functions. Finally, we are interested in using the proposed approach in practical applications of bilevel programming such as network interdiction.

References

[1] Charles Audet, Pierre Hansen, Brigitte Jaumard, and Gilles Savard. Links between linear bilevel and mixed 0–1 programming problems. *J. Optim. Theory Appl.*, 93(2):273–300, 1997.

[2] Jonathan F Bard. *Practical Bilevel Optimization: Algorithms and Applications*. Kluwer Academic Publishers, Dordrecht, 1998.

[3] Jonathan F Bard and James T Moore. A branch and bound algorithm for the bilevel programming problem. *SIAM J. Sci. Stat. Comput.*, 11(2):281–292, 1990.

[4] Jonathan F Bard and James T Moore. An algorithm for the discrete bilevel programming problem. *Naval Res. Logist.*, 39(3):419–435, 1992.

[5] Jonathan F Bard, John Plummer, and Jean Claude Sourie. A bilevel programming approach to determining tax credits for biofuel production. *European J. Oper. Res.*, 120(1):30–46, 2000.

[6] Omar Ben-Ayed. Bilevel linear programming. *Comput. Oper. Res.*, 20(5):485–501, 1993.

[7] Luce Brotcorne, Saïd Hanafi, and Raïd Mansi. A dynamic programming algorithm for the bilevel knapsack problem. *Oper. Res. Lett.*, 37(3):215–218, 2009.

[8] Gerald Brown, Matthew Carlyle, Javier Salmerón, and Kevin Wood. Defending critical infrastructure. *Interfaces*, 36(6):530–544, 2006.

[9] Anthony P Burgard, Priti Pharkya, and Costas D Maranas. Optknock: a bilevel programming framework for identifying gene knockout strategies for microbial strain optimization. *Biotechnol. Bioeng.*, 84(6):647–657, 2003.

[10] Dong Cao and Lawrence C Leung. A partial cooperation model for non-unique linear two-level decision problems. *European J. Oper. Res.*, 140(1):134–141, 2002.

[11] Massimiliano Caramia and Renato Mari. Enhanced exact algorithms for discrete bilevel linear problems. *Optim. Lett.*, 9(7):1447–1468, 2015.

[12] Suh-Wen Chiou. Bilevel programming for the continuous transport network design problem. *Transport. Res. B-Meth.*, 39(4):361–383, 2005.

[13] Benoît Colson, Patrice Marcotte, and Gilles Savard. An overview of bilevel optimization. *Ann. Oper. Res.*, 153(1):235–256, 2007.

[14] Jean-Philippe Côté, Patrice Marcotte, and Gilles Savard. A bilevel modelling approach to pricing and fare optimisation in the airline industry. *J. Revenue Pricing Manag.*, 2(1):23–36, 2003.

[15] Stephan Dempe. Discrete bilevel optimization problems. *Technical Report D-04109, Institut fur Wirtschaftsinformatik, Universitat Leipzig, Leipzig, Germany*, 2001.

[16] Stephan Dempe. *Foundations of Bilevel Programming*. Kluwer Academic Publishers, Dordrecht, 2002.

[17] Stephan Dempe and F Mefo Kue. Solving discrete linear bilevel optimization problems using the optimal value reformulation. *J. Global Optim.*, pages 1–23, 2016.

[18] Stephan Dempe and Klaus Richter. *Bilevel programming with knapsack constraints*. TU Bergakademie, Fakultät für Mathematik und Informatik, 2000.

[19] Scott DeNegre. *Interdiction and discrete bilevel linear programming.* PhD thesis, 2011.

[20] ST DeNegre and Ted K Ralphs. A branch-and-cut algorithm for integer bilevel linear programs. In *Operations Research and Cyber-infrastructure*, pages 65–78. Springer, 2009.

[21] PC Gilmore and RE Gomory. The theory and computation of knapsack functions. *Oper. Res.*, 14(6):1045–1074, 1966.

[22] Fatma Gzara. A cutting plane approach for bilevel hazardous material transport network design. *Oper. Res. Lett.*, 41(1):40–46, 2013.

[23] Pierre Hansen, Brigitte Jaumard, and Gilles Savard. New branch-and-bound rules for linear bilevel programming. *SIAM J. Sci. Stat. Comput.*, 13(5):1194–1217, 1992.

[24] Eitan Israeli and R Kevin Wood. Shortest-path network interdiction. *Networks*, 40(2):97–111, 2002.

[25] Robert G Jeroslow. Some basis theorems for integral monoids. *Math. Oper. Res.*, 3(2):145–154, 1978.

[26] JJ Júdice and AM Faustino. A sequential LCP method for bilevel linear programming. *Ann. Oper. Res.*, 34(1):89–106, 1992.

[27] Nan Kong, Andrew J Schaefer, and Brady Hunsaker. Two-stage integer programs with stochastic right-hand sides: a superadditive dual approach. *Math. Program.*, 108(2):275–296, 2006.

[28] L Lozano and J C Smith. A value-function-based exact approach for the bilevel mixed integer programming problem. *Oper. Res., to appear*, 2017.

[29] Athanasios Migdalas. Bilevel programming in traffic planning: models, methods and challenge. *J. Global Optim.*, 7(4):381–405, 1995.

[30] Athanasios Migdalas, Panos M Pardalos, and Peter Värbrand. *Multilevel Optimization: Algorithms and Applications*, volume 20. Springer Science & Business Media, 2013.

[31] James T Moore and Jonathan F Bard. The mixed integer linear bilevel programming problem. *Oper. Res.*, 38(5):911–921, 1990.

[32] George L. Nemhauser and Laurence A. Wolsey. *Integer and Combinatorial Optimization*. Wiley-Interscience, New York, NY, USA, 1988.

[33] Osman Y Özaltın, Oleg A Prokopyev, and Andrew J Schaefer. The bilevel knapsack problem with stochastic right-hand sides. *Oper. Res. Lett.*, 38(4):328–333, 2010.

[34] Michael Patriksson and R Tyrrell Rockafellar. A mathematical model and descent algorithm for bilevel traffic management. *Transport. Sci.*, 36(3):271–291, 2002.

[35] Shaogang Ren, Bo Zeng, and Xiaoning Qian. Adaptive bi-level programming for optimal gene knockouts for targeted overproduction under phenotypic constraints. *BMC Bioinformatics*, 14(2):S17, 2013.

[36] Siqian Shen, J Cole Smith, and Roshan Goli. Exact interdiction models and algorithms for disconnecting networks via node deletions. *Discrete Optim.*, 9(3):172–188, 2012.

[37] Yen Tang, Jean-Philippe P Richard, and J Cole Smith. A class of algorithms for mixed-integer bilevel min–max optimization. *J. Global Optim.*, 66(2):225–262, 2016.

[38] Andrew C Trapp, Oleg A Prokopyev, and Andrew J Schaefer. On a level-set characterization of the value function of an integer program and its application to stochastic programming. *Oper. Res.*, 61(2):498–511, 2013.

[39] Luis Vicente, Gilles Savard, and J Judice. Discrete linear bilevel programming problem. *J. Optim. Theory Appl.*, 89(3):597–614, 1996.

[40] Heinrich Von Stackelberg. *The Theory of the Market Economy*. Oxford University Press, 1952.

[41] Ue-Pyng Wen and YH Yang. Algorithms for solving the mixed integer two-level linear programming problem. *Comput. Oper. Res.*, 17(2):133–142, 1990.

[42] R Kevin Wood. Deterministic network interdiction. *Math. Comput. Model.*, 17(2):1–18, 1993.

[43] Bo Zeng and Yu An. Solving bilevel mixed integer program by reformulations and decomposition. *Optimization online*, 2014.

[44] Junlong Zhang and Osman Y Özaltın. Test instances for discrete bilevel linear programming. https://sites.google.com/a/ncsu.edu/ozaltin/home/bilevel/, May 2017.