European Journal of Operational Research 291 (2021) 661–679

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

An enhanced branch-and-bound algorithm for bilevel integer linear programming

Shaonan Liu [a], Mingzheng Wang [b,*], Nan Kong [c], Xiangpei Hu [b]

[a] School of Economics and Management, Dalian University of Technology, Dalian 116024, China
[b] School of Management, Zhejiang University, Hangzhou 310058, China
[c] Weldon School of Biomedical Engineering, Purdue University, West Lafayette, USA

ARTICLE INFO

Article history:
Received 7 November 2019
Accepted 1 October 2020
Available online 8 October 2020

Keywords:
Integer programming
Bilevel programming
Branch and bound
Enhanced branching

ABSTRACT

Bilevel integer linear programming (BILP) problems have been studied for decades. Many exact algorithms have been proposed in recent years for small- or medium-sized instances. However, few of these algorithms were shown to be efficient on large-sized instances. In this paper, we present an enhanced branch-and-bound algorithm for a class of BILP problems, which can discard a subspace from the search space in each iteration larger than that in a benchmark branch-and-bound algorithm. The corresponding enhanced branching rule can efficiently slow down the creation of new node problems so as to significantly reduce the computation time. Our scheme may be suboptimal if the lower-level problem is not unique optimal as the enhanced branching rule may discard bilevel feasible solutions that may turn out to be optimal to the bilevel programming. We present computational studies to evaluate the algorithm speedup and solution quality of our algorithm, compared with state-of-the-art algorithms from the literature on a large testbed of general BILP instances, some of which are still unsolved. The computational results show that our enhanced branching rule can achieve significant speedup on the benchmark branching rule with satisfying solution quality. In particular, our algorithm shows superior performance on large-sized BILP instances with a relatively complex lower-level problem.

1. Introduction

Bilevel programming describes the interaction between two autonomous and possibly conflicting decision makers: a leader and a follower. It plays a fundamental role in many real-world applications, when competitive agents operate in a hierarchical way with conflicting objectives. Discrete bilevel linear programming problems, which contain integer decision variables, have been commonly seen in facility location (Cao & Chen, 2006; Caramia & Mari, 2016), network design (Ceylan & Bell, 2004), bilevel knapsack (Caprara, Carvalho, Lodi & Woeginger, 2016; Lozano & Smith, 2016 and Tang, Richard & Smith, 2016), traffic systems (Brotcorne, Labbé, Marcotte & Savard, 2001, Labbé, Marcotte & Savard, 1998), capacity planning (Florensa et al., 2017; Garcia-Herreros et al., 2016), and natural gas regulation (Dempe, Kalashnikov & RıÓs-Mercado, 2005, 2011; Kalashnikov & Ríos-Mercado, 2006; Kalashnikov, Pérez & Kalashnykova, 2010). For these problems, specially designed exact algorithms were proposed. However, these algorithms were tested efficient only on small- to medium-sized instances. On the other hand, there is a lack of exact algorithms dealing with large-sized instances. This can be explained by the inherent complexity of bilevel programming problem, which is known to be NP-hard even when the leader's and follower's problems are both linear programs (Jeroslow, 1985).

To fill in this gap, we propose an enhanced branch-and-bound algorithm for bilevel integer linear programming (BILP) problems where decision variables are all integers. We take a branching rule previously proposed by Xu and Wang (2014) as the benchmark and embed an enhanced branching idea in it to efficiently slow down the creation of new node problems by eliminating newly created node problems in each iteration. The corresponding enhanced branching rule is proved to be able to efficiently cut off larger subspace from the search space, so as to significantly reduce the computation time. However, it may discard bilevel feasible solutions if the lower-level problem is not uniquely optimal, which may lead to sub-optimality in BILP. Nevertheless, we adapt a well-established sufficient-and-necessary condition on the solution uniqueness of linear programming and provide a reasonable global optimality checking mechanism for BILP. To test the effi-

* Corresponding author.
E-mail addresses: l_shnan@163.com (S. Liu), wangmzh@zju.edu.cn (M. Wang), nkong@purdue.edu (N. Kong), drhxp@dlut.edu.cn (X. Hu).

https://doi.org/10.1016/j.ejor.2020.10.002
0377-2217/

ciency of our algorithm, including algorithm speedup (compared to the benchmark branching rule) and solution quality (i.e., the gap between our output solution and the optimal solution), we compare our algorithm with state-of-the-art algorithms on a common testbed of three sets of general BILP instances from the literature and a set of randomly generated BILP instances. To the best of our knowledge, these instances are of the largest size among general BILP instances tested in the literature. The computational results show that our enhanced branching rule can achieve superiority on both algorithm speedup and solution quality, especially for large-sized BILP instances with a relatively complex lower-level problem.

The rest of this paper is organized as follows. In Section 2, we review the most relevant approaches to discrete bilevel linear programming from the literature. We describe our model formulation and related definitions in Section 3. We introduce our enhanced branch-and-bound algorithm in Section 4. The performance of our algorithm is evaluated in Section 5 by means of computational experiments on three sets of general BILP instances from the literature and randomly generated ones. We draw conclusions in Section 6.

2. Literature review

Even though there exists a large body of literature devoted to bilevel optimization, there are relatively few exact algorithms designed for discrete bilevel linear programming. In this section, we introduce relevant references on solution methods of *bilevel 0–1 mixed integer linear programming*, *bilevel mixed integer linear programming (BMILP)* and *bilevel integer linear programming (BILP)*.

For bilevel 0–1 mixed integer linear programming, an early study can be found in Bard and Moore (1992), where both upper- and lower-level variables are restricted to be binary. They proposed an algorithm that implicitly enumerates upper level variables and solves the associated lower-level problems to obtain bilevel feasible solutions. The authors tested their algorithm on instances with up to 45 variables and 18 constraints. More than twenty years later, Zhang and Özaltın (2017)) proposed a branch-and-cut algorithm enhanced by incorporating the ideas of value function and local search for problems where upper level variables are restricted to be binary. Their computational results showed that their algorithm can solve instances with up to 200 variables and 150 constraints in a reasonable amount of time. For a special case known as the bilevel knapsack problem with interdiction constraints, where the leader seeks to minimize the follower's objective, exact algorithms were proposed by Caprara et al. (2016), Lozano and Smith (2016), Tang et al. (2016) and Croce and Scatamacchia (2019, 2020). Taking a straightforward cutting plane approach for the upper level problem, Caprara et al. (2016) proposed an exact algorithm that exploits the structural properties of the bilevel knapsack problem. Lozano and Smith (2016) proposed a sampling scheme to optimize the bilevel knapsack problem in which the follower problem can take any form. Tang et al. (2016) proposed three generic solution algorithms and required leader variables to be binary, whereas the follower's problem can be general mixed-integer. Croce and Scatamacchia (2019, 2020) derived effective lower bounds for the bilevel knapsack problem with interdiction constraints and presented an exact method that exploits the structure of the induced follower's problem. The authors tested their algorithms on bilevel knapsack problem instances with interdiction constraints that have up to 110 variables in total.

The literature of BMILP started with early work in the 1990s by Moore and Bard (1990), which was first to propose a branch-and-bound approach. The largest instance that the authors reported solution results on has 40 variables and 18 constraints. Vicente, Savard and Judice (1996) derived characterizing properties for different types of discrete linear bilevel programming problems. The authors studied the geometry of the feasible set and discussed the existence of an optimal solution. About twenty years later, Xu and Wang (2014) designed an exact algorithm within a branch-and-bound framework for BMILP with bounded and integral assumption on upper level variables. The authors tested their algorithm on instances of size rising to 920 variables and 368 constraints. The average computation time ranged from 10 minutes to 4 hours, and the longest solution time taken was nearly six hours. Employing a class of intersection cuts valid to BMILP under mild assumptions, Fischetti, Ljubić, Monaci & Sinnl, 2018, Fischetti, Ljubić, Monaci and Sinnl (2016 proposed a branch-and-cut algorithm for BMILP problems and developed a new family of cuts for BMILP problems. Further, Fischetti, Ljubić, Monaci and Sinnl (2017a) extended their algorithm in Fischetti et al. (2018) by suggesting new types of intersection cuts and introduced the so-called hypercube intersection cut, which allows for nonlinear terms to appear in both constraints and objective functions. The authors tested their algorithm on instances with up to 80,000 variables and 5000 constraints from the literature. Other BMILP algorithms that use cutting plane techniques within a branch-and-bound framework can be found in Hemmati and Smith (2016) and Tahernejad, Ralphs and DeNegre (2016). Note that algorithms in these papers only solve small-sized instances.

Besides a branch-and-bound framework, a few other lines of research for BMILP were proposed based on reformulation, Benders decomposition, and parametric programming. Zeng and An (2014) proposed a novel scheme based on reformulation and decomposition for BMILP problems with upper level constraints only depending on upper level variables. Their approach was extended by Yue, Gao, Zeng and You (2019) to a projection-based variant, which allows upper level constraints to depend on lower-level variables. Saharidis and Ierapetritou (2009) proposed a Benders-decomposition-based algorithm, which is based on the idea of decomposing the initial problem into a relaxation of the original problem and a series of restriction problems. Based on multi-parametric programming theory, Avraamidou and Pistikopoulos (2019) and Faísca, Dua, Rustem, Saraiva and Pistikopoulos (2007) proposed algorithms to solve bilevel quadratic programming problems, where the objective functions of the upper-level and lower-level problems are allowed to be quadratic. Note that the above work only presented computational experiments on small-sized instances.

There are also several approaches to solving real-world applications modeled as BMILP, e.g., Caramia and Mari (2016) for facility location problems, Dempe et al. (2011), Kalashnikov and Ríos-Mercado (2006), and Dempe et al. (2005) for natural gas regulation. Caramia and Mari (2016) proposed a decomposition based algorithm that resembles the algorithm in Saharidis and Ierapetritou (2009), but it is properly designed to cope with the bilevel structure of the facility location problem and the integrality of a subset of variables under the control of the leader. The authors tested their algorithm on a set of benchmark instances available in the literature with up to 150 facilities and 150 clients. Dempe et al. (2011), Kalashnikov and Ríos-Mercado (2006), and Dempe et al. (2005) linearized their BMILP problems and designed algorithms based on a penalty function approach. The real-world instances they solved have dimensions up to 1000. Recently, Zare, Borrero, Zeng and Prokopyev (2019) presented two strong-duality-based reformulations of a class of BMILP problems with the key idea of exploiting the binary expansion of upper level integer variables. They tested their approaches on three instance classes, i.e. bounded BMILP instances with less than 500 variables and constraints, BMILP with interdiction constraints instances with less than 300 variables and 400 constraints, and bilevel facility location

instances with up to 40 facilities and 240 products. They showed that their approaches can lead to orders of magnitude reduction in computation time.

For BILP problems which are most relevant to our paper, an earlier study can be found in DeNegre and Ralphs (2009) which proposed a branch-and-cut algorithm using cutting plane techniques, based on the previous branch-and-bound algorithm for BMILP in Moore and Bard (1990). They tested their algorithm on a set of interdiction instances with knapsack constraints at the upper level with up to 34 variables and 19 constraints. The computational results showed that their approach makes improvement on Moore and Bard (1990) with fewer nodes in their branch-and-bound tree. Similarly, in a branch-and-bound framework, Caramia and Mari (2015) derived valid inequalities to eliminate bilevel infeasible solutions for a given upper level solution. They reported solving BILP problems with up to 25 variables and 25 constraints. Wang and Xu (2017) presented an excellent algorithm called watermelon algorithm for BILP, which integrates branch-and-bound and cutting plane techniques. The watermelon algorithm is the first exact BILP algorithm that does not rely on additional simplifying assumptions. The authors tested their method on the same instances as in Xu and Wang (2014) with integrality restrictions and showed that the watermelon algorithm outperforms previous branch-and-bound algorithms in DeNegre and Ralphs (2009) and Xu and Wang (2014), Moore and Bard (1990). Besides branch-and-bound algorithms, Domínguez and Pistikopoulos (2010) introduced two multiparametric algorithms for BILP and BMILP, and tested their algorithms on small-sized instances.

3. Model formulation and definitions

We present the BILP problem as:

$$\max_{x,y} \ \zeta = c^T x + d_1^T y \tag{1}$$

$$\text{s.t. } A_1 x + B_1 y \le b_1, \tag{2}$$

$$0 \le x \le X, \tag{3}$$

$$x \in \mathbb{Z}^{n_1}, \tag{4}$$

$$y \in \operatorname*{argmax}_{\tilde{y}} \left\{ d_2^T \tilde{y} : A_2 x + B_2 \tilde{y} \le b_2, 0 \le \tilde{y} \le Y, \ \tilde{y} \in \mathbb{Z}^{n_2} \right\}, \tag{5}$$

where $A_1 \in \mathbb{Q}^{m_1 \times n_1}$, $A_2 \in \mathbb{Z}^{m_2 \times n_1}$, $B_1 \in \mathbb{Q}^{m_1 \times n_2}$, $B_2 \in \mathbb{Z}^{m_2 \times n_2}$, $b_1 \in \mathbb{Q}^{m_1}$, $b_2 \in \mathbb{Z}^{m_2}$, $c \in \mathbb{Q}^{n_1}$, $d_1 \in \mathbb{Q}^{n_2}$, $d_2 \in \mathbb{Q}^{n_2}$, $X \in \mathbb{Q}^{n_1}$, and $Y \in \mathbb{Q}^{n_2}$ are finite rational or integer parameters.

Since rational parameters can be equivalently converted to integers by multiplication, without loss of generality, the lower-level parameters $(A_2, \ B_2, b_2)$ are assumed to be integers. As formulation (1)–(5) indicates, if the lower-level problem has alternative optimal solutions, then the follower will select a $\hat{y}$ that maximizes the upper level objective function, thus benefiting the leader. Thus the formulation (1)–(5) is commonly known as the optimistic formulation of the problem. In contrast, under the pessimistic formulation, the follower will pick a $\hat{y}$ that either violates the upper level constraint or otherwise makes the least contribution to the upper level objective function. More detailed discussions regarding the two formulations can be found in Dempe, Mordukhovich and Zemkoho (2014), Lozano and Smith (2017), Xu and Wang (2014), and Wiesemann, Tsoukalas, Kleniati and Rustem (2013).

In the following, we introduce several definitions to facilitate the description of our algorithm. We use a vector with a subscript j to refer to the jth element of the vector. We define the set $\bar{\mathbb{R}} = \mathbb{R} \cup \{+\infty\} \cup \{-\infty\}$ as the extended real number line including positive and negative infinity. For a given $x \in \mathbb{Z}^{n_1}$, we denote $\mathcal{L}(x)$ as the **lower-level problem (LLP)**:

$$\max_{\tilde{y}} \ \left\{ d_2^T \tilde{y} : \ A_2 x + B_2 \tilde{y} \le b_2; \ 0 \le \tilde{y} \le Y; \ \tilde{y} \in \mathbb{Z}^{n_2} \right\}.$$

For a given set of parameters $(l, u \in \bar{\mathbb{R}}^{m_2}, w \in \bar{\mathbb{R}})$, we define a **node problem** $\mathcal{B}(l, u, w)$ by the following parametric BILP problem:

$$\max_{x,y} \ \zeta = c^T x + d_1^T y$$

$$\text{s.t. } A_1 x + B_1 y \le b_1, \tag{6}$$

$$l \le A_2 x \le u, \tag{7}$$

$$d_2^T y \ge w, \tag{8}$$

$$0 \le x \le X, \tag{9}$$

$$x \in \mathbb{Z}^{n_1}, \tag{10}$$

$$y \in \operatorname*{argmax}_{\tilde{y}} \left\{ d_2^T \tilde{y} : A_2 x + B_2 \tilde{y} \le b_2, 0 \le \tilde{y} \le Y, \ \tilde{y} \in \mathbb{Z}^{n_2} \right\}. \tag{11}$$

By definition, formulation (1)–(5) is equivalent to $\mathcal{B}(-\infty, \infty, -\infty)$, which will be used as the root node problem. We define the **relaxation problem** $\mathcal{R}(l, u, w)$ by the following integer linear programming (ILP) problem, which is referred to as the high point problem in Moore and Bard (1990):

$$\max_{x,y} \ \zeta = c^T x + d_1^T y$$

$$\text{s.t. } A_1 x + B_1 y \le b_1,$$

$$A_2 x + B_2 y \le b_2,$$

$$l \le A_2 x \le u,$$

$$d_2^T y \ge w,$$

$$0 \le x \le X,$$

$$0 \le y \le Y,$$

$$x \in \mathbb{Z}^{n_1}, \ y \in \mathbb{Z}^{n_2}.$$

For a given BILP $\mathcal{B}(l, u, w)$, a solution (x, y) is called bilevel feasible if it satisfies constraints (6)–(11). A solution is called bilevel infeasible if it is not bilevel feasible. A solution (x^*, y^*) is called bilevel optimal if it is bilevel feasible and we have $c^T x^* + d_1^T y^* \ge c^T x^0 + d_1^T y^0$ for any other bilevel feasible solution (x^0, y^0). A BILP problem is called optimal if a bilevel optimal solution exists (unique or not). A BILP problem is called infeasible if no bilevel feasible solution exists. In this paper, both $\mathcal{B}(l, u, w)$, its relaxation $\mathcal{R}(l, u, w)$ and $\mathcal{L}(x)$ are bounded for any $l, u \in \bar{\mathbb{R}}^{m_2}, w \in \bar{\mathbb{R}}$, and $x \in \mathbb{Z}^{n_1}$. Therefore, for a BILP problem defined in (1)–(5), there are only two possible outcomes: optimal or infeasible.

4. An enhanced branch-and-bound algorithm

Our algorithm utilizes a branch-and-bound framework, which iteratively solves the relaxation problem and removes its optimal

solution from the search space if the optimal solution of the relaxation problem is labeled bilevel infeasible. The novelty lies in an enhanced branching rule. In the following, we first introduce a benchmark branching rule and then propose the enhanced branching rule. This is followed by a presentation of our algorithm.

4.1. A benchmark branching rule

The idea of our algorithm starts by solving the relaxation problem of the root node problem. If the relaxation problem is infeasible, the BILP is also infeasible. If the relaxation problem is optimal with an optimal solution (x^R, y^R), we solve the LLP $\mathcal{L}(x^R)$. The following Lemma 1 provides a necessary and sufficient condition for (x^R, y^R) to be bilevel optimal.

Lemma 1. *Let (x^R, y^R) be an optimal solution to $\mathcal{R}(\hat{l}, \hat{u}, \hat{w})$. Then (x^R, y^R) is optimal to $\mathcal{B}(\hat{l}, \hat{u}, \hat{w})$ if and only if y^R is optimal to $\mathcal{L}(x^R)$.*

Proof. The "only if" direction is a direct result of the bilevel optimality of (x^R, y^R).

For the "if" direction, we have that if y^R is optimal to $\mathcal{L}(x^R)$, then (x^R, y^R) is a bilevel feasible solution to $\mathcal{B}(\hat{l}, \hat{u}, \hat{w})$, and thus it provides a lower bound of $\mathcal{B}(\hat{l}, \hat{u}, \hat{w})$. Meanwhile, (x^R, y^R) achieves an upper bound of $\mathcal{B}(\hat{l}, \hat{u}, \hat{w})$ since it is optimal to $\mathcal{R}(\hat{l}, \hat{u}, \hat{w})$. Hence, the "if" statement follows. □

However, it may be more common that y^R is not optimal to $\mathcal{L}(x^R)$. Then (x^R, y^R) is a bilevel infeasible solution which should be eliminated from the search space. The following Lemma 2, taking Lemmas 3 and 4 of Xu and Wang (2014) for reference, introduces a subspace containing (x^R, y^R) but no bilevel feasible solutions.

Lemma 2. ***(Xu & Wang, 2014).*** *If (x^R, y^R) is optimal to $\mathcal{R}(\hat{l}, \hat{u}, \hat{w})$ but bilevel infeasible. Suppose that y^L is an optimal solution to $\mathcal{L}(x^R)$, then the following subspace:*

$$\mathcal{P} = \left\{(x, y) : A_2x + B_2y^L \le b_2, d_2^T y < d_2^T y^L\right\}$$

contains (x^R, y^R) but no bilevel feasible solutions.

Proof. Since y^L is an optimal solution to $\mathcal{L}(x^R)$ but y^R is not, we have $y^L \in \mathbb{Z}^{n_2}$, $0 \le y^L \le Y$, $A_2x^R + B_2y^L \le b_2$ and $d_2^T y^R < d_2^T y^L$, that is $(x^R, y^R) \in \mathcal{P}$. For any $(\bar{x}, \bar{y}) \in \mathcal{P}$, we show that $\bar{y}$ can't be optimal to $\mathcal{L}(\bar{x})$. Actually, we have $A_2\bar{x} + B_2y^L \le b_2$ and $d_2^T\bar{y} < d_2^T y^L$, namely, y^L is a feasible solution to $\mathcal{L}(\bar{x})$ and superior to $\bar{y}$. □

By Lemma 2, we remove the subspace $\mathcal{P}$ from the search space, i.e., the feasible region of $\mathcal{R}(\hat{l}, \hat{u}, \hat{w})$, so as to eliminate the bilevel infeasible solution without eliminating any bilevel feasible solution. The way the search space is updated could introduce additional non-convexity for the remaining search space since it is inside the larger convex search space but outside the smaller convex subspace $\mathcal{P}$. To avoid introducing additional non-convexity, we partition the x-y space into $(m_2 + 2)$ pieces: $\mathcal{P}$, $\mathcal{P}_1, \ldots, \mathcal{P}_{m_2+1}$, where $\mathcal{P}_k$, $k = 1, \ldots, m_2$, are defined as:

$$\mathcal{P}_k = \{(x, y) : \left(A_2x + B_2y^L\right)_i \le (b_2)_i, \forall i = 1, \ldots, k-1; \left(A_2x + B_2y^L\right)_k > (b_2)_k\},$$

$\mathcal{P}_{m_2+1}$ is defined as

$$\mathcal{P}_{m_2+1} = \{(x, y) : A_2x + B_2y^L \le b_2, d_2^T y \ge d_2^T y^L\}.$$

Then we create the $(m_2 + 1)$ new node problems $\mathcal{B}(l^k, u^k, w^k)$, $\forall k = 1, \ldots, (m_2 + 1)$ from the intersections between the feasible region of parent node problem and the $(m_2 + 1)$ pieces: $\mathcal{P}_1, \ldots, \mathcal{P}_{m_2+1}$, respectively. As a result, $(m_2 + 1)$ new node problems are introduced with the following branching rule, presented by Xu and Wang (2014). We take it as the benchmark but restrict the parameters (l, u, w) of a node problem $\mathcal{B}(l, u, w)$ to integers as we investigate pure integer cases in this paper.

A benchmark branching rule (Xu & Wang, 2014): *Let (x^R, y^R) be an optimal solution to $\mathcal{R}(\hat{l}, \hat{u}, \hat{w})$. Suppose y^L is an optimal solution to $\mathcal{L}(x^R)$ but y^R is not. The following $(m_2 + 1)$ new node problems, denoted as $\mathcal{B}(l^k, u^k, w^k)$, $\forall k = 1, \ldots, (m_2 + 1)$, can be created from the parent node problem $\mathcal{B}(\hat{l}, \hat{u}, \hat{w})$:*

For $k = 1, \ldots, m_2$, we have

$$l_j^k = \begin{cases} \left(b_2 - B_2y^L\right)_k + 1, & \text{if } j = k, \\ \hat{l}_j, & \text{otherwise;} \end{cases}$$

$$u_j^k = \begin{cases} \min\left\{\left(b_2 - B_2y^L\right)_j, \hat{u}_j\right\}, & \text{if } j = 1, \ldots, k-1, \\ \hat{u}_j, & \text{otherwise;} \end{cases}$$

$$w^k = \hat{w}.$$

For $k = m_2 + 1$, we have

$$l^{m_2+1} = \hat{l},$$

$$u^{m_2+1} = \min\left\{b_2 - B_2y^L, \hat{u}\right\},$$

$$w^{m_2+1} = d_2^T y^L.$$

Now we prove that the new node problems are strictly strengthened by proving that: (a) $l_k^k = (b_2 - B_2y^L)_k + 1 > \hat{l}_k$; (b) $w^{m_2+1} = d_2^T y^L > \hat{w}$.

For (a): we have that $(b_2 - B_2y^L)_k + 1 > (b_2 - B_2y^L)_k \ge (A_2x^R)_k \ge \hat{l}_k$. The first inequality is straightforward; the second one follows that y^L is feasible to $\mathcal{L}(x^R)$; and the last one follows that x^R is feasible to $\mathcal{R}(\hat{l}, \hat{u}, \hat{w})$.

For (b): we have that $d_2^T y^L > d_2^T y^R \ge \hat{w}$. The former inequality follows from the definition of y^L and the latter follows that y^R is feasible to $\mathcal{R}(\hat{l}, \hat{u}, \hat{w})$.

The benchmark branching rule partitions the parent node problem into $(m_2 + 1)$ new node problems, and thus depends on the number of lower-level constraints. If the lower-level problem has a large number of complex constraints, the algorithm may become computationally expensive to execute as a large number of new node problems need to be created each time. Next, we introduce the enhanced branching rule.

4.2. An enhanced branching rule

To save the computation time, We attempt to further contract the remaining search space, after removing the subspace $\mathcal{P}$ from the search space and creating $(m_2 + 1)$ new node problems from the parent node problem $\mathcal{B}(\hat{l}, \hat{u}, \hat{w})$. We propose an enhanced branching rule which can efficiently eliminate a certain number of new node problems and strengthen the remaining node problems at each iteration.

First, we introduce a property of the lower-level problem as in the following Lemma 3.

Lemma 3. *Let y^L be an optimal solution to $\mathcal{L}(x^R)$ for some $x^R \in \mathbb{Z}^{n_1}$. If $\exists i \in \{1, \ldots, m_2\}$ such that $(A_2x^R + B_2y^L)_i < (b_2)_i$, then y^L is the optimal solution to $\mathcal{L}(x)$ for any x satisfying $A_2x^R + B_2y^L \le A_2x + B_2y^L \le b_2$.*

Proof. If there exists some x^0 satisfying $A_2x^R + B_2y^L \le A_2x^0 + B_2y^L \le b_2$ such that y^L is not optimal to $\mathcal{L}(x^0)$, then there exists some y^0 $(0 \le y^0 \le Y, y^0 \in \mathbb{Z}^{n_2})$ such that $A_2x^0 + B_2y^0 \le b_2$ and $d_2^T y^0 > d_2^T y^L$. Then we have $A_2x^R + B_2y^0 \le b_2$, which implies that y^0 is also feasible for $\mathcal{L}(x^R)$ and $d_2^T y^0 > d_2^T y^L$. This violates the optimality of y^L for $\mathcal{L}(x^R)$. □

Lemma 3 motivates us to search for a bilevel feasible solution to BILP within the set

$$\mathcal{S}_0\left(x^R, y^L\right) = \left\{x \in \mathbb{Z}^{n_1} | A_2 x \in \left[A_2 x^R, b_2 - B_2 y^L\right]\right\}.$$

Furthermore, if y^L is a unique optimal solution to $\mathcal{L}(x^R)$, we can prove that y^L is a unique optimal solution to $\mathcal{L}(\hat{x})$ for any $\hat{x} \in \mathcal{S}_0(x^R, y^L)$. Then we select those $\hat{x}$ from $\mathcal{S}_0(x^R, y^L)$ such that $(\hat{x}, y^L)$ satisfies the upper level constraints to ensure the bilevel feasibility, i.e., $\hat{x} \in \mathcal{S}_1(x^R, y^L)$, where

$$\mathcal{S}_1\left(x^R, y^L\right) = \left\{x |\ A_1 x + B_1 y^L \le b_1,\ 0 \le x \le X,\ x \in \mathcal{S}_0\left(x^R, y^L\right)\right\}.$$

Finally, we choose the one from $\mathcal{S}_1(x^R, y^L)$ that achieves the best upper level objective value as the optimal solution to BILP problem with $A_2 x \in [A_2 x^R, b_2 - B_2 y^L]$. We summarize the above idea in the following Lemma 4 and Lemma 5.

Lemma 4. *If y^L is a unique optimal solution to $\mathcal{L}(x^R)$, then y^L is a unique optimal solution to $\mathcal{L}(\hat{x})$ for any $\hat{x} \in \mathcal{S}_0(x^R, y^L)$.*

Proof. To prove Lemma 4, we only need to prove that the feasible region of $\mathcal{L}(x^R)$ contains that of $\mathcal{L}(\hat{x})$. We denote $\mathcal{F}(x^R) = \{y \in [0, Y] | A_2 x^R + B_2 y \le b_2,\ y \in \mathbb{Z}^{n_2}\}$ and $\mathcal{F}(\hat{x}) = \{y \in [0, Y] | A_2 \hat{x} + B_2 y \le b_2,\ y \in \mathbb{Z}^{n_2}\}$ as the feasible region of $\mathcal{L}(x^R)$ and $\mathcal{L}(\hat{x})$, respectively. Since $\hat{x} \in \mathcal{S}_0(x^R, y^L)$, we have $A_2 \hat{x} \ge A_2 x^R$. For any $\hat{y} \in \mathcal{F}(\hat{x})$, we have $B_2 \hat{y} \le b_2 - A_2 \hat{x} \le b_2 - A_2 x^R$, that is $\hat{y} \in \mathcal{F}(x^R)$ and $\mathcal{F}(\hat{x}) \subseteq \mathcal{F}(x^R)$. □

Lemma 5. *Let y^L be a unique optimal solution to $\mathcal{L}(x^R)$ for some $x^R \in \mathbb{Z}^{n_1}$. If the following problem, denoted as $\mathcal{Q}(x^R, y^L)$:*

$$\begin{aligned} \max_x\ & c^T x + d_1^T y^L \\ \text{s.t. } & A_1 x + B_1 y^L \le b_1, \\ & 0 \le x \le X, \\ & A_2 x \ge A_2 x^R, \\ & A_2 x \le b_2 - B_2 y^L, \\ & x \in \mathbb{Z}^{n_1}. \end{aligned}$$

is optimal, we denote the optimal solution as x^Q, then (x^Q, y^L) is an optimal solution to BILP (1)-(5) with $A_2 x \in [A_2 x^R, b_2 - B_2 y^L]$. Otherwise, if $\mathcal{Q}(x^R, y^L)$ is infeasible, then there is no bilevel feasible solution to BILP (1)-(5) with $A_2 x \in [A_2 x^R, b_2 - B_2 y^L]$.

Proof. Based on Lemmas 3 and 4, since x^Q satisfies $A_2 x^R + B_2 y^L \le A_2 x^Q + B_2 y^L \le b_2$, y^L is a unique optimal solution to $\mathcal{L}(x^Q)$. Therefore, (x^Q, y^L) is bilevel feasible. Since x^Q is the optimal solution to $\mathcal{Q}(x^R, y^L)$, (x^Q, y^L) is the best bilevel feasible solution that can be found for formulation (1)–(5) with $A_2 x \in [A_2 x^R, b_2 - B_2 y^L]$. If $\mathcal{Q}(x^R, y^L)$ is infeasible, for any x satisfies $A_2 x^R + B_2 y^L \le A_2 x + B_2 y^L \le b_2$, (x, y^L) violates at least one constraint in the upper level. Therefore, there is no bilevel feasible solution to (1)–(5) with $A_2 x \in [A_2 x^R, b_2 - B_2 y^L]$. □

Based on Lemma 5 above, there is no better bilevel feasible solution than (x^Q, y^L) or no bilevel feasible solution at all within the point set: $\{x \in \mathbb{Z}^{n_1} | A_2 x \in [A_2 x^R, b_2 - B_2 y^L]\}$. Therefore, after removing the subspace $\mathcal{P}$ from the search space (based on the benchmark branching rule), we solve $\mathcal{Q}(x^R, y^L)$ and record the solution (if there is any), and then we carve the point set: $\{x \in \mathbb{Z}^{n_1} | A_2 x \in [A_2 x^R, b_2 - B_2 y^L]\}$ out from the remaining search space. Specifically, we find those subscripts $p \in \{1, \ldots, m_2\}$ such that $(A_2 x^R + B_2 y^L)_p < (b_2)_p$, and cut off the corresponding interval $(A_2 x)_p \in [(A_2 x^R)_p, (b_2 - B_2 y^L)_p]$ from the feasible region of the $(m_2 + 1)$ new node problems: $\mathcal{B}(l^k, u^k, w^k)$, $\forall k = 1, \ldots, (m_2 + 1)$, described in the benchmark branching rule. A few new node problems will be eliminated in this process. The number of the eliminated new node problems is determined by the relationship between $[(A_2 x^R)_p, (b_2 - B_2 y^L)_p]$ and $[\hat{l}_p, \hat{u}_p]$ (contain or be contained). There are totally 4 relationships: (I). $(A_2 x^R)_p = \hat{l}_p \le \hat{u}_p \le (b_2 - B_2 y^L)_p$; (II). $(A_2 x^R)_p = \hat{l}_p \le (b_2 - B_2 y^L)_p \le \hat{u}_p - 1$; (III). $\hat{l}_p + 1 \le (A_2 x^R)_p \le \hat{u}_p \le (b_2 - B_2 y^L)_p$; (IV). $\hat{l}_p + 1 \le (A_2 x^R)_p \le (b_2 - B_2 y^L)_p \le \hat{u}_p - 1$. As we will see next, the first three relationships can reduce the number of new node problems while the last one will double the number of new node problems.

We analyze the four relationships in detail as:

(I) In relationship I, if there exists a p such that $(A_2 x^R)_p = \hat{l}_p \le \hat{u}_p \le (b_2 - B_2 y^L)_p$, all the $(m_2 + 1)$ new node problems can be eliminated, i.e., the corresponding parent node problem can be totally cut off. Based on Lemma 5, the polyhedron $A_2 x \in [A_2 x^R, b_2 - B_2 y^L]$ can be carved out from the search space. That is, the interval $[(A_2 x^R)_p, (b_2 - B_2 y^L)_p]$ can be cut off from $[\hat{l}_p, \hat{u}_p]$, which results in an empty set for $(A_2 x)_p$. Subsequently, $\mathcal{R}(\hat{l}, \hat{u}, \hat{w})$ becomes infeasible.

Consequently, if there exists a p such that $(A_2 x^R)_p = \hat{l}_p \le \hat{u}_p \le (b_2 - B_2 y^L)_p$, then the parent node $\mathcal{B}(\hat{l}, \hat{u}, \hat{w})$ can be totally cut off after $\mathcal{Q}(x^R, y^L)$ is solved and the solution (if there is any) is recorded.

(II) In relationship II, if there exists a p such that $(A_2 x^R)_p = \hat{l}_p \le (b_2 - B_2 y^L)_p \le \hat{u}_p - 1$, then the feasible region of $\mathcal{B}(l^k, u^k, w^k)$, $\forall\ k = p + 1, \ldots, m_2 + 1$, gets empty since the corresponding bound $\hat{l}_p \le (A_2 x)_p \le \min\{(b_2 - B_2 y^L)_p, \hat{u}_p\}$ is totally contained in $[(A_2 x^R)_p, (b_2 - B_2 y^L)_p]$, while the first p node problems are strictly strengthened.

Specially, if there exists a p such that $(A_2 x^R)_p = \hat{l}_p \le (b_2 - B_2 y^L)_p \le \hat{u}_p - 1$, then the first p new node problems, $\mathcal{B}(l^k, u^k, w^k)$, $\forall k = 1, \ldots, p$, can be strengthened as:

$$l_j^k = \begin{cases} \left(b_2 - B_2 y^L\right)_j + 1 & \text{if } j = k, p, \\ \hat{l}_j & \text{otherwise;} \end{cases}$$

$$u_j^k = \begin{cases} \min\left\{\left(b_2 - B_2 y^L\right)_j, \hat{u}_j\right\} & \text{if } j = 1, \ldots, k-1, \\ \hat{u}_j & \text{otherwise;} \end{cases}$$

$$w^k = \hat{w},$$

while the other $(m_2 + 1 - p)$ new node problems, $\mathcal{B}(l^k, u^k, w^k)$, $\forall\ k = p + 1, \ldots, m_2 + 1$, can be eliminated.

(III) In relationship III, if there exists a p such that $\hat{l}_p + 1 \le (A_2 x^R)_p \le \hat{u}_p \le (b_2 - B_2 y^L)_p$, the feasible region of $\mathcal{B}(l^p, u^p, w^p)$ becomes empty since the corresponding bound $(b_2 - B_2 y^L)_p + 1 \le (A_2 x)_p \le \hat{u}_p$ is totally contained in $[(A_2 x^R)_p, (b_2 - B_2 y^L)_p]$, while the other node problems are strictly strengthened.

Consequently, if there exists a p such that $\hat{l}_p + 1 \le (A_2 x^R)_p \le \hat{u}_p \le (b_2 - B_2 y^L)_p$, then the node $\mathcal{B}(l^p, u^p, w^p)$ can be eliminated, while the other m_2 new node problems can be strengthened as:

For node $\mathcal{B}(l^k, u^k, w^k)$, $\forall k = 1, \ldots, p-1, p+1, \ldots, m_2$, we have that

$$l_j^k = \begin{cases} \left(b_2 - B_2 y^L\right)_k + 1 & \text{if } j = k, \\ \hat{l}_j & \text{otherwise;} \end{cases}$$

$$u_j^k = \begin{cases} \min\left\{\left(b_2 - B_2 y^L\right)_j, \hat{u}_j\right\} & \text{if } j = 1, \ldots, k-1,\ j \ne p, \\ \left(A_2 x^R\right)_p - 1 & \text{if } j = p, \\ \hat{u}_j & \text{otherwise;} \end{cases}$$

$$w^k = \hat{w}.$$

For node $\mathcal{B}(l^{m_2+1}, u^{m_2+1}, w^{m_2+1})$, we have that

$$l^{m_2+1} = \hat{l};$$

$$u_j^{m_2+1} = \begin{cases} \left(A_2x^R\right)_p - 1 & \text{if } j = p, \\ \min\left\{\left(b_2 - B_2y^L\right)_j, \hat{u}_j\right\} & \text{otherwise;} \end{cases}$$

$$w^{m_2+1} = d_2^T y^L.$$

(IV) In relationship IV, if there exists a p such that $\hat{l}_p + 1 \le (A_2x^R)_p \le (b_2 - B_2y^L)_p \le \hat{u}_p - 1$, the number of new node problems will be doubled. Actually, cutting off each interval $[(A_2x^R)_p, (b_2 - B_2y^L)_p]$ from $[\hat{l}_p, \hat{u}_p]$ results in two disjoint intervals $[\hat{l}_p, (A_2x^R)_p - 1]$ and $[(b_2 - B_2y^L)_p + 1, \hat{u}_p]$, which corresponds to two new node problems. The algorithm will be drastically slowed down. Therefore, in relationship IV, we will not cut off the interval at all.

From the above, we integrate the effects of relationship I, II and III when cutting off $\{x \in \mathbb{Z}^{n_1} | A_2x \in [A_2x^R, b_2 - B_2y^L]\}$ from the remaining search space, and summarize it with the following enhanced branching rule.

An enhanced branching rule: *Let (x^R, y^R) be an optimal solution to $\mathcal{R}(\hat{l}, \hat{u}, \hat{w})$. Suppose y^L is an optimal solution to $\mathcal{L}(x^R)$ but y^R is not.*

(1) *If there exists a p s.t. $(A_2x^R)_p = \hat{l}_p \le \hat{u}_p \le (b_2 - B_2y^L)_p$, the node problem $\mathcal{B}(\hat{l}, \hat{u}, \hat{w})$ can be totally eliminated.*

(2) *Otherwise, the following $(q_1 - r)$ new node problems, denoted as $\mathcal{B}(l^k, u^k, w^k)$, $k = 1, \ldots, q_1$, $k \ne s_t$, $t = 1, \ldots, r$, can be created from the parent node problem $\mathcal{B}(\hat{l}, \hat{u}, \hat{w})$:*

$$l_j^k = \begin{cases} \left(b_2 - B_2y^L\right)_j + 1 & \text{if } j = k, q_1, \ldots, q_H, \\ \hat{l}_j & \text{otherwise;} \end{cases}$$

$$u_j^k = \begin{cases} \min\left\{\left(b_2 - B_2y^L\right)_j, \hat{u}_j\right\} & \text{if } j = 1, \ldots, k-1, j \notin \{s_1, \ldots, s_T\}, \\ \left(A_2x^R\right)_j - 1 & \text{if } j = s_1, \ldots, s_T, \\ \hat{u}_j & \text{otherwise;} \end{cases}$$

$$w^k = \begin{cases} d_2^T y^L & \text{if } k = m_2 + 1, \\ \hat{w} & \text{otherwise,} \end{cases}$$

where $q_1 < q_2 < \ldots < q_H$ are the subscripts satisfying relationship II (if relationship II is never satisfied, denote $q_1 = m_2 + 1$), and $s_1 < s_2 < \ldots < s_T$ are the subscripts satisfying relationship III, and r is the number of $\{s_1, s_2, \ldots, s_T\}$ such that $s_t \le q_1$, $t = 1, \ldots, T$ (if such subscript does not exist, denote $r = 0$).

In the following, we quantify the number of new node problems that the enhanced branching can eliminate in Lemma 6 and prove that the $(q_1 - r)$ new node problems created by the enhanced branching rule are strictly strengthened in Lemma 7.

Lemma 6. *In each iteration of the branch-and-bound framework, the number of new node problems that are eliminated by the enhanced branching is $m_2 + 1$ or $m_2 + 1 - q_1 + r$, where q_1 is the minimum subscript satisfying relationship II (if relationship II does not exist, denote $q_1 = m_2 + 1$), and r is the number of subscripts that are smaller than q_1 and satisfy relationship III.*

Proof. If there exists a subscript p such that $(A_2x^R)_p = \hat{l}_p \le \hat{u}_p \le (b_2 - B_2y^L)_p$ (i.e., relationship I), cutting off $[(A_2x^R)_p, (b_2 - B_2y^L)_p]$ from $[\hat{l}_p, \hat{u}_p]$ results in an empty feasible region for the parent node problem. The corresponding parent node problem can be totally cut off, i.e., all $(m_2 + 1)$ new node problems can be eliminated.

Otherwise, we find the smallest subscript satisfying relationship II and denote it as q_1, such that $(A_2x^R)_{q_1} = \hat{l}_{q_1} \le (b_2 - B_2y^L)_{q_1} \le \hat{u}_{q_1} - 1$. Since cutting off $[(A_2x^R)_{q_1}, (b_2 - B_2y^L)_{q_1}]$ from $[\hat{l}_{q_1}, \hat{u}_{q_1}]$ results in an empty feasible region of $\mathcal{B}(l^k, u^k, w^k)$, $\forall\, k = q_1 + 1, \ldots, m_2 + 1$, the last $m_2 + 1 - q_1$ new node problems are eliminated. Further, we find all the subscripts $s_1 < s_2 < \ldots < s_r$ that are smaller than q_1 and satisfy relationship III, i.e., $\hat{l}_{s_t} + 1 \le (A_2x^R)_{s_t} \le \hat{u}_{s_t} \le (b_2 - B_2y^L)_{s_t}$, $t = 1, \ldots, r$. Since cutting off $[(A_2x^R)_{s_t}, (b_2 - B_2y^L)_{s_t}]$ from $[\hat{l}_{s_t}, \hat{u}_{s_t}]$ results in an empty feasible region of $\mathcal{B}(l^k, u^k, w^k)$, $\forall\, k = s_1, s_2, \ldots, s_r$, the corresponding r new node problems are eliminated. All in all, there are $m_2 + 1 - q_1 + r$ new node problems eliminated totally by the enhanced branching. □

Lemma 7. *The $(q_1 - r)$ new node problems created by the enhanced branching rule are further strictly strengthened compared with those in the benchmark branching rule.*

Proof. We prove the lemma by showing that: (a) $(b_2 - B_2y^L)_j + 1 \ge \hat{l}_j$; (b) $(A_2x^R)_p - 1 < \min\{(b_2 - B_2y^L)_p, \hat{u}_p\}$. For (a), we have that $(b_2 - B_2y^L)_j + 1 > (b_2 - B_2y^L)_j \ge (A_2x)_j \ge \hat{l}_j$. For (b), we have that $(A_2x^R)_p - 1 < (A_2x^R)_p \le (b_2 - B_2y^L)_p$ and $(A_2x^R)_p - 1 < (A_2x^R)_p \le \hat{u}_p$. □

Our enhanced branching rule will degenerate to the benchmark branching rule for the bilevel mix-integer linear programming which contains continuous lower-level problem, since the interval $A_2x \in [A_2x^R, b_2 - B_2y^L]$ is empty for such instances.

We should notice that our enhanced branching rule is proposed based on the assumption in Lemma 5 that the optimal solution to lower-level problem is unique. Otherwise, if there is any other optimal solution $\bar{y} \ne y^L$, we cannot draw the same conclusion, and the enhanced branching rule in this situation may discard some underlying bilevel feasible solutions, which may be superior to (x^Q, y^L) such that the solution we get may not be bilevel optimal but bilevel feasible. Next, we discuss uniqueness of the optimal solution y^L to the lower-level problem $\mathcal{L}(x^R)$.

Recall that $\mathcal{L}(x^R)$ is formulated as follows:

$$\max_{\tilde{y}} \; d_2^T\tilde{y},$$

$$\text{s.t. } B_2\tilde{y} \le b_2 - A_2x^R,$$

$$0 \le \tilde{y} \le Y,$$

$$\tilde{y} \in \mathbb{Z}^{n_2}.$$

For similarity of formulation, we omit $0 \le \tilde{y} \le Y$ by blending it in with $B_2\tilde{y} \le b_2 - A_2x^R$. First, we construct a linear programming problem denoted as $\mathcal{T}(x^R, y^L)$ as follows:

$$\max_{\tilde{y}} \; 0,$$

$$s.t.\; B_2\tilde{y} \le b_2 - A_2x^R,$$

$$d_2^T\tilde{y} = d_2^Ty^L.$$

Note that any optimal solution to $\mathcal{L}(x^R)$ is also optimal to $\mathcal{T}(x^R, y^L)$. We denote $(B_2)_i$ as the ith row of B_2 and define

$$J = \left\{i | (B_2)_i y^L = \left(b_2 - A_2x^R\right)_i\right\}.$$

We denote $(B_2)_J$ as the matrice whose rows are $(B_2)_i$, $i \in J$.

Based on Theorem 2 in Mangasarian (1979), if and only if there exists no y satisfying

$$d_2^Ty = 0,\;\; (B_2)_Jy \le 0,\;\; y \ne 0,$$

y^L is a unique optimal solution to $\mathcal{T}(x^R, y^L)$, and hence y^L is a unique optimal solution to $\mathcal{L}(x^R)$. We should notice that this is a

sufficient but not necessary condition for y^L to be a unique optimal solution to $\mathcal{L}(x^R)$. This condition is equivalent to the following program denoted as $\mathcal{U}(x^R, y^L)$:

$$\max_y \ y^2,$$

$$\text{s.t. } d_2^T y = 0,$$

$$(B_2)_J y \le 0,$$

$$J = \left\{ i | (B_2)_i y^L = \left(b_2 - A_2 x^R \right)_i \right\},$$

has an optimal solution value of zero.

In particular, we propose an optimality checking mechanism for BILP as follows. If $\mathcal{U}(x^R, y^L)$ has an optimal solution value of zero in each step, the output result of our algorithm is guaranteed optimal; otherwise, there is no global optimality guarantee for BILP.

In our computational experiments (Section 6), we test the solution quality of our algorithm with the enhanced branching rule, i.e., how big the gap is between our solution and the optimal solution, in solving randomly generated BILP instances without the uniqueness of lower-level problem's optimal solution.

4.3. An enhanced branch-and-bound algorithm

We are now ready to present our branch-and-bound algorithm employing the enhanced branching rule, which takes the parameter set $(A_1, A_2, B_1, B_2, b_1, b_2, c, d_1, d_2, X, Y)$ as input and outputs a promising solution (x^*, y^*, ζ^*) to the BILP (1)-(5). The notation of $(x^* = \emptyset, y^* = \emptyset, \zeta^* = -\infty)$ is used as the output for infeasible instance. We summarize the steps of the algorithm below, where parameter z^j is used to record the objective value of the relaxation of node j for bounding purpose, OPT is set as an indicator in the algorithm that indicates the potential exactness of the output. That is, if OPT remains to 1 when the algorithm terminates, the output is guaranteed optimal based on the above optimality checking mechanism. In addition, parameter N is used to specify the number of active nodes in the branch-and-bound tree.

$(x^*, y^*, \zeta^*) = \text{Algorithm}(A_1, A_2, B_1, B_2, b_1, b_2, c, d_1, d_2, X, Y)$.

Step 0 (Initialization): Create the root node $\mathcal{B}(l^1, u^1, w^1)$ with $l^1 = -\infty, u^1 = \infty, \ w^1 = -\infty$. Initialize $x^* = \emptyset, y^* = \emptyset, \ \zeta^* = -\infty, N = 1, \ \text{OPT} = 1, \text{ and } z^1 = \infty$. Go to Step 1.

Step 1 (Node management): For all $k \in \{1, \ldots, N\}$ such that $z^k \le \zeta^*$ or $l^k \not\le u^k$, remove node k. Update N as the number of remaining nodes.

---

if $N = 0$ **then**
  if $x^* \neq \emptyset$ **then**
    if OPT $=1$ **then**
      1(a) return (x^*, y^*, ζ^*) is an optimal solution to BILP (1)-(5).
    else
      1(b) return (x^*, y^*, ζ^*) is a heuristically optimal solution to BILP (1)-(5).
    end
  else
    if OPT $=1$ **then**
      1(c) return BILP (1)-(5) is infeasible.
    else
      1(d) return BILP (1)-(5) is heuristically infeasible.
    end
  end
else
  1(e) select a node k from $\{1, \ldots, N\}$, set $\left(\hat{l} = l^k, \hat{u} = u^k, \hat{w} = w^k\right)$, remove node k, reorder the remaining nodes from 1 to $N-1$, reduce N by 1, and go to Step 2.
end

Step 2 (Relaxation): Solve $\mathcal{R}(\hat{l}, \hat{u}, \hat{w})$.

if $\mathcal{R}(\hat{l}, \hat{u}, \hat{w})$ is infeasible **then**
  2(a) go to Step 1.
else
  let (x^R, y^R) denote an optimal solution to $\mathcal{R}(\hat{l}, \hat{u}, \hat{w})$.
  if $c^T x^R + d_1^T y^R \leq \zeta^*$ **then**
    2(b) go to Step 1.
  else
    2(c) go to Step 3.
  end
end

Step 3 (Lower level): Solve $\mathcal{L}(x^R)$
Let y^L denote an optimal solution to $\mathcal{L}(x^R)$.

if $d_2^T y^R = d_2^T y^L$ **then**
  3(a) update $(x^* = x^R, y^* = y^R, \zeta^* = c^T x^R + d_1^T y^R)$ and go to Step 1.
else
  if $\mathcal{Q}(x^R, y^L)$ is optimal (denote the optimal solution as x^Q), and $c^T x^Q + d_1^T y^L > \zeta^*$,
    then
    update $x^* = x^Q, y^* = y^L, \zeta^* = c^T x^Q + d_1^T y^L$
    if $c^T x^R + d_1^T y^R \leq \zeta^*$ **then**
      3(b) go to Step 1.
    end
  end
  if OPT =1 and $\mathcal{U}(x^R, y^L)$ has an optimal solution of 0 **then**
    update OPT =1.
  else
    update OPT = 0.
  end
  3(c) go to Step 4.
end

Step 4 ($[A_2x^R, b_2 - B_2y^L]$)

if $\exists p \in \{1,2,\dots,m_2\}$ s.t. $(A_2x^R)_p = \hat{l}_p \le \hat{u}_p \le (b_2 - B_2y^L)_p$ **then**

  4(a) go to Step 1.

else

  if $\exists j \in \{1,2,\dots,m_2\}$ s.t. $(A_2x^R)_j = \hat{l}_j \le (b_2 - B_2y^L)_j \le \hat{u}_j - 1$, **then**

    find all the subscript $q_1 < q_2 < \cdots < q_H$ such that $(A_2x^R)_{q_h} = \hat{l}_{q_h} \le (b_2 - B_2y^L)_{q_h} \le \hat{u}_{q_h} - 1, h = 1,\dots,H$.

  else

    let $q_1 = m_2 + 1,\ H = 1$.

  end

  if $\exists j \in \{1,2,\dots,m_2\}$ s.t. $\hat{l}_j + 1 \le (A_2x^R)_j \le \hat{u}_j \le (b_2 - B_2y^L)_j$, **then**

    find all the subscript $s_1 < s_2 < \cdots < s_T$ such that $\hat{l}_{s_t} + 1 \le (A_2x^R)_{s_t} \le \hat{u}_{s_t} \le (b_2 - B_2y^L)_{s_t}, t = 1,\dots,T$, and count the number of $\{s_1, s_2, \dots, s_T\}$, such that $s_t \le q_1$, and denote the number as r.

  else

    let $s_1 = m_2 + 1,\ r = 0$.

  end

  4(b) go to Step 5

end

Step 5 (An enhanced branching rule): Create $(q_1 - r)$ new node problems, increase N by $(q_1 - r)$, and go to Step 1. For $k = 1,\dots,q_1,\ k \ne s_t,\ t = 1,\dots,r$, node $(N+k)$ is characterized by $(l^{N+k}, u^{N+k}, w^{N+k}, z^{N+k})$, which is defined as

$$l_j^{N+k} = \begin{cases} \left(b_2 - B_2y^L\right)_j + 1 & \text{if } j = k, q_1, \dots, q_H, \\ \hat{l}_j & \text{otherwise;} \end{cases}$$

$$u_j^{N+k} = \begin{cases} \min\left\{\left(b_2 - B_2y^L\right)_j, \hat{u}_j\right\} & \text{if } j = 1,\dots,k-1, j \notin \{s_1,\dots,s_T\}, \\ \left(A_2x^R\right)_j - 1 & \text{if } j = s_1,\dots,s_T, \\ \hat{u}_j & \text{otherwise;} \end{cases}$$

$$w^{N+k} = \begin{cases} d_2^Ty^L & \text{if } k = m_2 + 1, \\ \hat{w} & \text{otherwise;} \end{cases}$$

$$z^{N+k} = c^Tx^R + d_1^Ty^R.$$

Next, we establish the finite termination and the correctness of our algorithm. Since we assume the variables to be discrete and bounded, the finite termination of our enhanced branch-and-bound algorithm is naturally guaranteed.

Theorem 1. *The output of our enhanced branch-and-bound algorithm is correct.*

Proof. We show the correctness of the algorithm step by step.

Steps 1(a), 1(b), 1(c), 1(d), 1(e), 2(a), 2(b), 2(c), 3(a), and 3(c) are standard procedures in a branch-and-bound algorithm.

Step 3(b) determines that (x^Q, y^L) is a bilevel optimal solution to $\mathcal{B}(\hat{l}, \hat{u}, \hat{w})$ because it is an optimal solution to $\mathcal{R}(\hat{l}, \hat{u}, \hat{w})$ and y^L is an optimal solution to $\mathcal{L}(x^R)$.

Step 4(a) determines that the feasible region of the current node contains no better bilevel feasible solutions, based on the discussion of relationship I in Section 4.

Step 4(b) calculates the number of the remaining new node problems after carving out a set of $A_2x \in [A_2x^R, b_2 - B_2y^L]$ of the feasible region of the current node in Step 5.

Step 5 carves out a set (obtained from Step 4) of $A_2x \in [A_2x^R, b_2 - B_2y^L]$ out from the current node and create $(q_1 - r)$ new branches based on the enhanced branching rule presented in Section 4. □

5. Computational results

In this section, we demonstrate the efficiency of our algorithm in solving BILP problem instances. We implement it in Matlab using CPLEX 12.9 as the ILP solver and run it on a large number of general BILP instances. All computational experiments are conducted on a desktop computer with 2.29 GHz CPU and 8 GB of RAM. Computing times reported in the following are in wall-clock seconds and the time limit for each run is set to be 3600 wall-clock seconds.

5.1. Testbed

We consider three sets of general BILP instances with no special structure. These instances, tested in previous studies, are termed WANGXU, DENEGRE and MIPLIB, respectively, in the literature. We also consider one set of randomly generated BILP instances to better illustrate the superior performance of our algorithm. We term this set of instances, WANGXU-LARGE, in our computational study.

- Instances of class WANGXU have been proposed in Wang and Xu (2017). They are based on the BMILP instances from

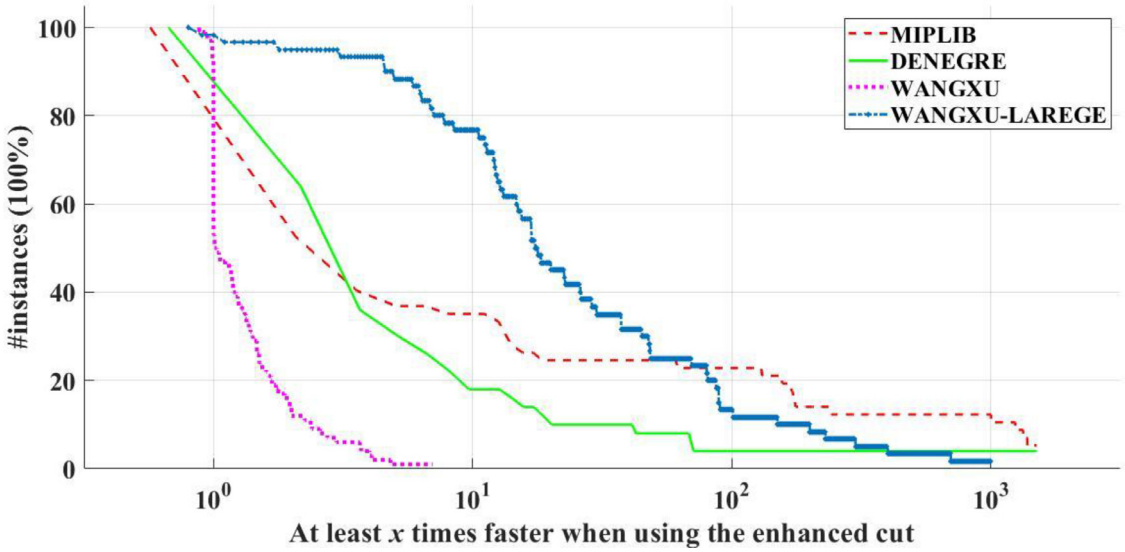

Fig. 1. Speed-ups achieved by the enhanced branching for three instance sets.

Xu and Wang (2014) requiring all variables to be integers. For these instances, one has $n_1 = n_2 \in \{10, 60, 110, \ldots, 460\}$ and $m_1 = m_2 = 0.4n_1$. Elements of the upper- (lower-) level matrices are real numbers (integers) uniformly distributed within a certain range: A_1, B_1, A_2 and B_2 are within $[0, 10]$; c, d_1 and d_2 are within $[-50, 50]$; b_1 is within $[30, 130]$; and b_2 is within $[10, 110]$. The bound on variables x and y is set to be $[0, 10]$. For each (n_1, m_1) pair, 10 instances are generated. The class WANGXU contains 100 instances totally. Among all the algorithms tested on WANGXU, the algorithm "WaterM-II" proposed by Wang and Xu (2017) has the best performance, as shown in Wang and Xu (2017).

- Since our enhanced branching idea is proposed to slow down the creation of new node problems by the benchmark branching rule, we expect that the enhanced branching rule can highlight its advantage when solving BILP instances with a medium- to large-sized lower-level problem. We create a new class WANGXU-LARGE of randomly generate BILP instances with a relatively large-sized lower-level problem but a small-sized upper-level problem by following the same procedure used for class WANGXU. For this new class, we have $n_1 = m_1 \in \{5, 6, \ldots, 10\}$, $n_2 \in \{500, 600, \ldots, 1000\}$ and $m_2 = 0.4n_2$. We set the bound of the decision variables to be $[-10, 10]$. We generate 10 instances for each (n_1, n_2) pair.
- Instances of class DENEGRE have been proposed in DeNegre (2011). For these instances, one has $n_1 \in \{5, 10, 15\}$, while the lower-level variable dimension n_2 is such that $n_1 + n_2 = 15$ or 20. There are $m_2 = 20$ lower-level constraints and no constraints in the upper-level problem. All coefficients are integers in the range $[-50, 50]$. The class DENEGRE contains 50 instances totally. Among all the algorithms tested on DENEGRE, the algorithm "MIX++" proposed by Fischetti et al. (2017a) has the best performance, as shown in Fischetti et al. (2017a).
- Instances of class MIPLIB have been introduced in Fischetti et al. (2016)) and are available at Fischetti, Ljubić, Monaci and Sinnl (2017b). They are based on instances of MILPLIB 3.0 (Bixby, Ceria, McZeal & Savelsbergh, 1998) containing only binary variables. These instances have been transformed into bilevel problems by considering the first Y% (rounded up) variables as lower-level variables, with $Y \in \{10, 50, 90\}$ and the remaining ones as upper-level variables. The objective function is used as the upper-level objective $c^T x + d_1^T y$ and the lower-level objective is set to be $d^T y = -d_1^T y$. All the constraints of these instances are defined to be lower-level constraints. The class MIPLIB contains 57 instances with up to about 80,000 relaxation-problem variables and 5000 lower-level constraints, making them much larger (and often also much more difficult to solve) than instances of the other classes. Among all the algorithms tested on MIPLIB, the algorithms "SEP2" proposed by Fischetti et al. (2016)) and "MIX++" proposed by Fischetti et al. (2017a) have the best performance, as shown in Fischetti et al. (2017a).

5.2. Computational analysis of the enhanced branching idea

In this subsection, we computationally evaluate the effect of our enhanced branching idea on the benchmark branching rule, proposed by Xu and Wang (2014). We denote the algorithm employing the benchmark branching rule and that employing the enhanced branching rule as "XW" and "Alg-E", respectively. We compare XW with Alg-E on three instance sets to directly test the performance of our enhanced branching idea.

We illustrate the importance of the enhanced branching idea through the cumulative speedup chart of Fig. 1. The chart shows the speedup values over the benchmark branching rule (i.e., XW) on four instance sets described in Section 5.1. The reported speedup ratio is calculated as $(t(\text{XW})+t_s)/(t(\text{Alg-E})+ t_s)$, where t(XW) and t(Alg-E) denote the computing time (in seconds) of algorithms XW and Alg-E, respectively. The time shift t_s is set to 1 second to reduce the importance of instances that are easy in the comparison. For a given instance set, each point (x, y) in this chart indicates that y% of instances in the corresponding class have a speedup ratio of at least x. Notice that the values on x-axis are given in log-scale. We observe that different degree of speed-up is achieved for the four instance sets. The most significant speedup is achieved for MIPLIB and WANGXU-LARGE, where the former is the most challenging set of benchmark instances in our study. Note that both sets contain BILP instances with a relatively complex lower-level problem. For about 35% of instances in MIPLIB, resp., 76% of instances in WANGXU-LARGE, a speedup of at least one order of magnitude is achieved; for about 25% of instances in MIPLIB, resp., 11% of instances in WANGXU-LARGE, the actual speedup is of two orders of magnitude or even higher, thanks to the use of the enhanced branching idea. For the instance set DENEGRE, a speedup

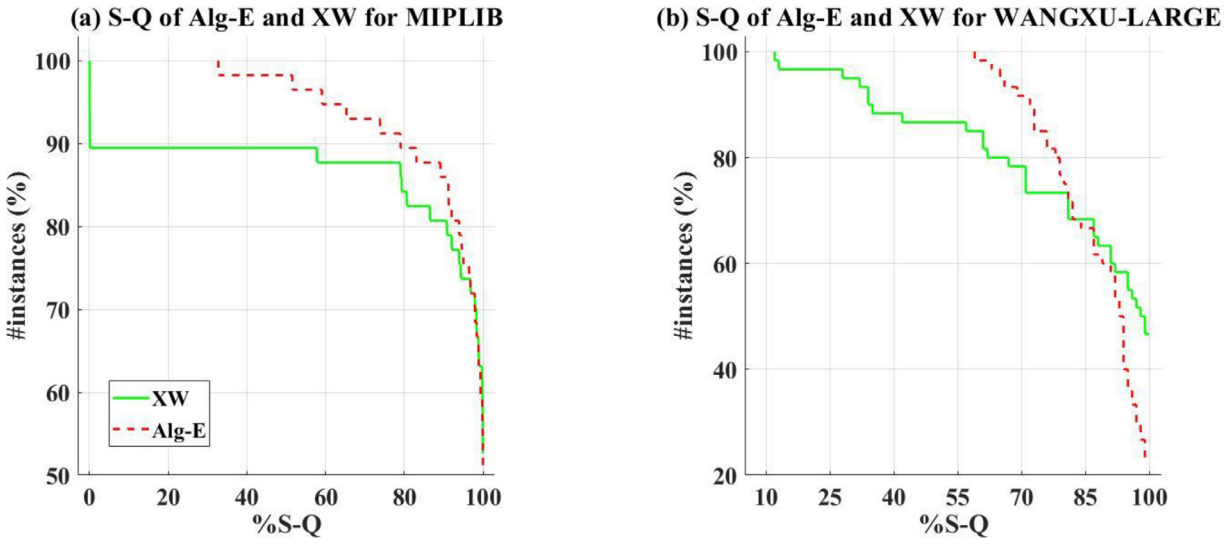

Fig. 2. Solution quality of Alg-E and XW for MIPLIB and WANGXU-LARGE.

of at least one order of magnitude is achieved for nearly 20% instances. For the instance set WANGXU, a slight speedup can be achieved since all the instances can be solved within several minutes. In few cases, a small slowdown is observed-this usually happens for instances that can be solved within a few seconds, in which case turning on the enhanced branching idea causes an unnecessary overhead.

To further analyze the effect of the enhanced branching, we next illustrate the solution quality of Alg-E and XW on three instance sets. The percentage value of solution quality is calculated as:

$$\left(1-\frac{\zeta^*-\zeta^S}{|\zeta^*|+10^{-10}}\right)\times\ 100\%,$$

where ζ^S is the solution value of the objective function output by the tested algorithm and ζ^* is the optimal objective function value. For the instance set MIPLIB, we denote ζ^* as the best solution value ever found for the instances that have not been solved to optimality so far. There two situations where the output solution may not be optimal so the percentage value of solution quality need to be calculated: on one hand, as mentioned in Section 4.2, Alg-E can achieve a bilevel feasible solution which may not be optimal, since all the benchmark instances are randomly generated and the optimal solution to lower-level problem may not be unique; On the other hand, if an algorithm cannot finish solving an instance within one hour, we terminate the algorithm prematurely and output the best bilevel feasible solution found so far.

Fig. 2(a) and (b) shows the cumulative chart for the percentage value of solution quality (denoted as "S-Q") obtained within one hour of XW and Alg-E for MIPLIB and WANGXU-LARGE respectively. For visualization, S-Q values smaller than 0 are set to 0; similarly, if an algorithm cannot find a feasible solution within one hour, we also set S-Q value to 0. For each instance set, each point (x, y) in this chart indicates that y% of all instances in the corresponding class have a percentage value of solution quality of at least x%. In particular, the rightmost point indicates the percentage of instances solved to optimality (or for which the best solution value ever found is obtained for the unsolved instances in MIPLIB) by the corresponding algorithm. In Fig. 2(a), a significant solution quality improvement is achieved by the enhanced branching idea for MIPLIB. For example, the S-Q is at least 32% for almost 90% of instances if XW is used, while it grows to 100% of instances when using Alg-E. Similarly, the S-Q is at least 80% for about 84% of instances if XW is used, while it grows to almost 90% of instances when using Alg-E. In Fig. 2(b), XW can solve nearly 50% instances to optimality for WANGXU-LARGE within one hour, but XW can only achieve an S-Q value of 33% for 90% instances while Alg-E can achieve an S-Q value of 73% for 90% instances. Actually, XW can solve all the instances of size $(n_1, n_2) \in \{(5500), (6600), (7700)\}$ to optimality. However, for instances of size $(n_1, n_2) \in \{(8800), (9900), (10, 1000)\}$, XW achieves rather disappointing solution quality within one hour since XW needs up to 2.5 hour to solve these instances to optimality. Whereas Alg-E can solve 92% of instances to an S-Q value of at least 70%, and solve all the instances to an S-Q value of at least 60%.We will show the details of the results for WANGXU-LARGE in the following subsection (in Table 2).

Table 1
Results for instance set WANGXU.

	XW	WaterM-I	WaterM-II	Alg-E	
n1-n2	Time	Time	Time	Time	S-Q
10–10	1	0	1	0	98.84%
60–60	11	2	10	2	91.99%
110–110	24	10	17	6	97.23%
160–160	31	13	20	14	95.75%
210–210	126	40	21	26	94.41%
260–260	185	57	31	66	94.02%
310–310	314	117	37	130	96.70%
360–360	147	110	44	46	100.00%
410–410	308	290	79	155	99.27%
460–460	395	268	95	159	99.30%
Average	**154**	**91**	**36**	**60**	**96.75%**

For DENEGRE and WANGXU, XW can solve all the instances in the two set to optimality within one hour, while Alg-E can solve 43% of instances in DENEGRE and 70% of instances in WANGXU to optimality. Specifically, for DENEGRE, Alg-E can solve 80% of instances to an S-Q value of at least 92%, while for WANGXU, Alg-E can solve 80% of instances to an S-Q value of at least 94%.

Overall, these experiments show that Alg-E adopting the enhanced branching rule has some unavoidable overhead and imperfection on solution quality for relatively small-sized instances but achieves overall speedup on most instances. In particular, our en-

Table 2
Results for instance set WANGXU-LARGE.

n1-n2	WaterM-II		XW		Alg-E	
	Time	S-Q	Time	S-Q	Time	S-Q
5–500	3600	79%	259	100%	22	91%
6–600	3600	80%	531	100%	52	85%
7–700	3600	82%	1121	91%	139	92%
8–800	3600	76%	2666	87%	159	85%
9–900	3600	47%	3600	73%	254	89%
10–1000	3600	69%	3600	49%	411	91%
Average	**3600**	**72%**	**1963**	**83%**	**173**	**89%**

hanced branching rule shows it advantage of speedup more obviously on large-sized instances with complex lower-level problem. For the instance set MIPLIB, Alg-E achieves both significant speedup and solution quality improvement for XW.

5.3. Comparison with state-of-the-art approaches from literature

In this subsection, we compare our algorithm Alg-E against state-of-the-art approaches from literature, on the same instance classes tested before.

Instance Set WANGXU. This class of instances is proposed by Wang and Xu (2017), where the computational results by a series of algorithms are reported, among which three algorithms: "XW" (Xu & Wang, 2014), "WaterM-I" and "WaterM-II" (Wang & Xu, 2017) are well-performed. Their results were obtained on "a desktop computer with 2.4 GHz" which is similar to our hardware. In Table 1, we report, for Alg-E, the average computing time (in seconds) and percentage value of solution quality (denoted as "S-Q") for each set of 10 instances. As XW, WaterM-I and WaterM-II can solve all the instances to optimality within the time limit, for these algorithms we only report the required computing time (in

Table 3
Results for instance set DENEGRE.

Instance	Opti-V	SEP1				MIX++		Alg-E				
		BestSol	Time	S-Q	Nodes	Time	Nodes	BestSol	Time	S-Q	Nodes	OPT
20–15–10–1	−388	−388	0	100.00%	50	1	21	−376	1	96.91%	55	0
20–15–10–2	−398	−398	17	100.00%	77,323	8	279	−398	2	100.00%	272	0
20–15–10–3	−42	−42	0	100.00%	2201	2	54	−42	1	100.00%	136	1
20–15–10–4	−729	−729	0	100.00%	185	1	56	−729	1	100.00%	61	0
20–15–10–5	−281	−281	0	100.00%	83	0	20	−281	0	100.00%	30	0
20–15–10–6	−246	−246	0	100.00%	233	16	205	−246	1	100.00%	162	0
20–15–10–7	−260	−260	0	100.00%	108	0	0	−260	0	100.00%	22	0
20–15–10–8	−293	−293	0	100.00%	114	0	22	−278	2	94.88%	134	0
20–15–10–9	−635	−635	0	100.00%	1061	1	16	−635	4	100.00%	401	0
20–15–10–10	−206	−206	0	100.00%	628	0	10	−196	0	95.15%	37	0
20–20–5–1	−548	−548	1	100.00%	6981	0	21	−545	0	99.45%	37	0
20–20–5–2	−568	−568	1558	100.00%	6053,523	0	49	−546	15	96.13%	688	0
20–20–5–3	−477	−477	0	100.00%	53	0	50	−477	0	100.00%	1	0
20–20–5–4	−753	−753	0	100.00%	142	0	71	−753	0	100.00%	1	0
20–20–5–5	−392	−392	0	100.00%	51	0	31	−392	0	100.00%	1	1
20–20–5–6	−1033	−1033	5	100.00%	79,502	0	92	−1018	1	98.55%	56	0
20–20–5–7	−547	−547	0	100.00%	80	0	16	−547	1	100.00%	42	0
20–20–5–8	−936	−936	0	100.00%	69	0	91	−936	0	100.00%	1	0
20–20–5–9	−868	−868	0	100.00%	112	0	62	−860	2	99.08%	104	0
20–20–5–10	−340	−340	0	100.00%	45	0	38	−330	1	97.06%	39	0
20–20–5–11	−426	−426	0	100.00%	9	0	11	−426	0	100.00%	1	0
20–20–5–12	−854	−854	0	100.00%	43	0	21	−854	0	100.00%	1	0
20–20–5–13	−514	−514	116	100.00%	947,138	0	11	−493	3	95.91%	241	0
20–20–5–14	−923	−923	0	100.00%	109	0	58	−923	0	100.00%	1	0
20–20–5–15	−617	−617	157	100.00%	1031,098	1	197	−617	2	100.00%	163	1
20–20–5–16	−833	−833	0	100.00%	2535	0	44	−833	2	100.00%	78	0
20–20–5–17	−895	−895	0	100.00%	3580	0	19	−859	1	95.98%	36	0
20–20–5–18	−356	−356	0	100.00%	2	0	0	−356	0	100.00%	1	0
20–20–5–19	−431	−431	3	100.00%	25,762	0	95	−426	2	98.84%	93	0
20–20–5–20	−438	−438	1	100.00%	3918	0	32	−432	1	98.63%	81	0
20–20–10–1	−359	−359	494	100.00%	1805,080	2	81	−347	6	96.66%	598	0
20–20–10–2	−659	−659	0	100.00%	939	1	17	−659	4	100.00%	286	1
20–20–10–3	−618	−618	1	100.00%	9456	0	52	−571	4	92.39%	269	0
20–20–10–4	−604	−604	3600	100.00%	7479,668	1	51	−592	15	98.01%	1347	0
20–20–10–5	−972	−972	0	100.00%	20	0	13	−972	0	100.00%	1	0
20–20–10–6	−731	−707	3600	96.72%	6244,669	10	511	−648	42	88.65%	2018	0
20–20–10–7	−683	−683	2788	100.00%	7420,465	0	54	−655	3	95.90%	242	0
20–20–10–8	−667	−667	3	100.00%	8116	15	232	−599	6	89.81%	523	0
20–20–10–9	−256	−256	4	100.00%	42,945	0	71	−195	4	76.17%	307	0
20–20–10–10	−441	−441	73	100.00%	256,927	2927	8068	−326	29	73.92%	2179	0
20–20–15–1	−450	−420	3600	93.33%	4313,453	0	16	−450	2	100.00%	217	0
20–20–15–2	−645	−645	3600	100.00%	14,175,981	0	6	−598	3	92.71%	371	0
20–20–15–3	−579	−579	838	100.00%	1420,792	3	43	−518	2	89.46%	246	0
20–20–15–4	−441	−441	3600	100.00%	5448,638	5	131	−441	3	100.00%	293	1
20–20–15–5	−271	−271	3600	100.00%	6169,959	1392	5466	−3	6	1.11%	732	0
20–20–15–6	−263	−263	3260	100.00%	5955,753	50	483	−197	1	74.90%	62	0
20–20–15–7	−471	−471	246	100.00%	787,848	0	23	−471	1	100.00%	126	0
20–20–15–8	−360	−360	3600	100.00%	11,797,237	0	3	−236	1	65.56%	159	0
20–20–15–9	−584	−584	1	100.00%	2027	0	8	−563	0	96.40%	57	0
20–20–15–10	−251	−251	0	100.00%	400	0	14	−118	1	47.01%	129	0
Average	–	–	**695**	**99.80%**	**1631,542**	**89**	**341**	–	**4**	**92.90%**	**263**	**–**

Table 4
Results for instance set MIPLIB

	SEP2				MIX++				Alg-E				
Instance	BestSol	Time	Nodes	S-Q	BestSol	Time	Nodes	S-Q	BestSol	Time	Nodes	S-Q	OPT
air03-0.1	382,822	3,600	146,125	98.44%	379,800	3,600	92,677	99.24%	387,656	47	191	97.16%	0
air03-0.5	505,172	3,600	85,478	100.00%	512,698	3,600	76,005	98.51%	637,254	65	251	73.85%	0
air03-0.9	823,130	3,600	44,697	93.11%	770,100	3,600	42,757	100.00%	900,420	52	158	83.08%	0
air04-0.1	56,563	3,600	55,921	99.71%	56,399	3,600	61,419	100.00%	57,029	2,624	3,224	98.88%	0
air04-0.5	60,131	3,600	35,826	99.91%	60,076	3,600	33,459	100.00%	63,333	3,600	2,574	94.58%	0
air04-0.9	84,993	3,600	3,752	84.77%	73,759	3,600	6,658	100.00%	89,235	3,600	181	79.02%	0
air05-0.1	26,801	3,600	101,047	99.16%	26,577	401	10,168	100.00%	27,515	516	1,285	96.47%	0
air05-0.5	32,497	3,600	92,234	96.14%	31,290	3,600	75,980	100.00%	44,082	925	1,656	59.12%	0
air05-0.9	44,567	3,600	82,050	90.12%	40,558	3,600	63,300	100.00%	60,217	1,487	2,358	51.53%	0
cap6000-0.1	–	3,600	1,980	–	-1,967,015	587	48,281	100.00%	-1,966,874	2,083	2,177	99.99%	0
cap6000-0.5	–	3,600	1,481	–	–	3,600	1,115	–	-1,634,335	2,136	2,177	100.00%	0
cap6000-0.9	-259,599	3,600	9,709	100.00%	–	3,600	328	–	-364,988	3,600	3,013	100.00%	0
enigma-0.1	0	0	990	100.00%	0	0	739	100.00%	0	0	1	100.00%	0
enigma-0.5	0	4	13,842	100.00%	0	6	10,531	100.00%	0	0	1	100.00%	0
enigma-0.9	0	46	2,670	100.00%	0	186	2,966	100.00%	0	0	1	100.00%	0
fast0507-0.1	12,562	3,600	604	99.38%	12,484	2	0	100.00%	12,484	1,383	508	100.00%	1
fast0507-0.5	61,516	3,600	7,767	99.87%	61,439	2	0	100.00%	61,439	1,169	508	100.00%	1
fast0507-0.9	109,916	8	2	100.00%	109,916	1	0	100.00%	109,916	1,419	508	100.00%	1
l152lav-0.1	4,722	2	367	100.00%	4,722	2	363	100.00%	4,722	1	1	100.00%	0
l152lav-0.5	4,866	3,600	311,915	100.00%	4,868	3,600	258,223	99.96%	4,966	25	177	97.94%	0
l152lav-0.9	5,090	3,600	211,309	99.65%	5,072	3,600	171,722	100.00%	5,518	35	227	91.21%	0
lseu-0.1	1,120	0	15	100.00%	1,120	0	19	100.00%	1,120	1	24	100.00%	0
lseu-0.5	2,525	3,600	13,333	90.83%	2,313	3,600	12,840	100.00%	2,563	1	75	89.19%	0
lseu-0.9	5,838	24	299	100.00%	5,838	65	357	100.00%	5,838	1	51	100.00%	0
mitre-0.1	122,310	3,600	20,791	99.94%	122,235	3,600	41,872	100.00%	122,250	3,600	2,062	99.99%	0
mitre-0.5	146,730	3,600	15,611	100.00%	–	3,600	19,004	–	147,030	3,600	1,970	99.80%	0
mitre-0.9	168,885	3,600	13,066	100.00%	–	3,600	10,099	–	169,215	3,600	2,027	99.80%	0
mod010-0.1	6,554	8	739	100.00%	6,554	4	9	100.00%	6,554	16	166	100.00%	0
mod010-0.5	6,692	3,600	117,241	98.88%	6,618	3,600	164,755	100.00%	6,828	28	265	96.83%	0
mod010-0.9	7,448	3,600	158,667	98.74%	7,355	3,600	111,883	100.00%	7,997	26	235	91.27%	0
nw04-0.1	17,066	820	2,884	100.00%	17,066	1,140	2,842	100.00%	17,928	70	54	94.95%	0
nw04-0.5	23,914	3,600	18,519	100.00%	24,100	3,600	8,472	99.22%	40,000	516	81	32.73%	0
nw04-0.9	43,374	3,600	12,282	100.00%	52,290	3,600	6,631	79.44%	58,422	144	52	65.31%	0
p0033-0.1	3,089	0	0	100.00%	3,089	0	0	100.00%	3,089	0	1	100.00%	0
p0033-0.5	3,095	0	2	100.00%	3,095	0	0	100.00%	3,095	0	17	100.00%	0
p0033-0.9	4,679	0	7	100.00%	4,679	0	6	100.00%	4,679	0	29	100.00%	0
p0201-0.1	12,465	3,600	5,092	98.04%	12,555	3,600	5,837	97.30%	12,225	1	134	100.00%	0
p0201-0.5	13,650	3,600	649,100	99.89%	13,635	1,113	71,052	100.00%	13,850	2	442	98.42%	0
p0201-0.9	15,025	1	150	100.00%	15,025	1	157	100.00%	15,025	1	170	100.00%	0
p0282-0.1	260,785	3,600	371,989	100.00%	260,781	4	272	100.00%	260,781	6	625	100.00%	0
p0282-0.5	273,069	3,600	998,732	99.85%	272,659	3,600	120,899	100.00%	274,353	40	3,490	99.38%	0
p0282-0.9	627,411	3,600	2,075,980	97.95%	616,034	3,600	175,290	99.81%	614,837	3	648	100.00%	0
p0548-0.1	11,301	3,600	54,071	97.74%	11,051	3,600	102,504	100.00%	11,174	4	637	98.89%	0
p0548-0.5	22,197	3,600	5,121	97.91%	–	3,600	11,943	–	21,742	18	1,692	100.00%	0
p0548-0.9	49,235	3,600	293,986	100.00%	49,509	3,600	17,003	99.44%	49,537	23	2,196	99.39%	0
p2756-0.1	14,444	3,600	36,718	87.70%	12,862	3,600	37,599	100.00%	12,879	1,774	10,054	99.87%	0
p2756-0.5	23,565	3,600	58,203	100.00%	25,384	3,600	18,777	92.28%	24,989	3,600	424	93.96%	0
p2756-0.9	35,087	3,600	13,687	95.65%	33,623	3,600	9,263	100.00%	36,309	3,600	262	92.01%	0
seymour-0.1	486	3,600	231	97.90%	476	3,600	48,178	100.00%	476	3,600	3,495	100.00%	0
seymour-0.5	836	3,600	564	96.41%	807	2	18	100.00%	807	3,600	2,627	100.00%	0
seymour-0.9	1,251	9	2	100.00%	1,251	1	0	100.00%	1,251	3,600	2,948	100.00%	0
stein27-0.1	18	22	983	100.00%	18	0	528	100.00%	18	2	119	100.00%	0
stein27-0.5	19	7	336	100.00%	19	0	5	100.00%	19	1	119	100.00%	0
stein27-0.9	24	0	0	100.00%	24	0	0	100.00%	24	1	119	100.00%	0
stein45-0.1	30	1,899	12,549	100.00%	30	3	2,999	100.00%	30	13	332	100.00%	1
stein45-0.5	32	658	18,613	100.00%	32	0	14	100.00%	32	9	332	100.00%	1
stein45-0.9	40	0	0	100.00%	40	0	0	100.00%	40	9	332	100.00%	1
Average	**–**	**2,398**	**108,409**	**95.06%**	**–**	**1,956**	**34,348**	**90.63%**	**–**	**987**	**1,044**	**94.29%**	–

seconds). In the last row, the average value of computing time, and S-Q for each algorithm is reported.

We point out that Alg-E can solve most instances to optimality with the average percentage value of solution quality of 96.75%. For the computing time, Alg-E performs better than XW and WaterM-I, but is inferior to WaterM-II.

Instance Set WANGXU-LARGE. In Table 2, we report results for the set WANGXU-LARGE where, similarly to the set WANGXU, we compare our algorithm Alg-E with XW and WaterM-II (we omit WaterM-I since it has been proved to be inferior to WaterM-II). To calculate the percentage value of solution quality, each instance in this set is solved to optimality by XW with up to 2.5 hours. The results show that Alg-E can efficiently speed up the algorithm XW with an average "S-Q" value of 89%, which is also better than that of XW. In particular, Alg-E significantly outperforms WaterM-II on both computing time and solution quality.

Instance Set DENEGRE. For this set of 50 bilevel instances introduced by DeNegre (2011), Fischetti et al. (2017a) reported the computational results by two algorithms "SEP1" (Fischetti et al., 2016) and "MIX++". Their results were obtained "on a cluster consisting of Intel Xeon E5–2670v2 with 2.5 GHz and 12 GB of RAM", which is therefore 2–5 times faster than our hardware. In Table 3,

we report, for SEP1 and Alg-E, and for each instance, the best obtained feasible solution (denoted as "BestSol"), the associated percentage value of solution quality (denoted as "S-Q"), the computing time (in seconds), and the total number of branch-and-bound nodes. For Alg-E, we also report the indicator "OPT" which indicates the exact optimality of the output if OPT = 1, and heuristic of the output if OPT = 0. As MIX++ can solve all the instances to optimality within one hour, for this algorithm we only report the required computing time and the number of nodes. The optimal solution value of each instance is given in the column "Opti-V". In the last row, the average value of computing time, number of nodes, and S-Q for each algorithm is reported.

We point out that Alg-E can finish solving all instances in a fraction of a second to 42 seconds with an average value of S-Q of 92.90%, where five instances are indicated to be exactly solved to optimality by "OPT = 1". Alg-E needs fewer nodes than SEP1 for all but two instances. Alg-E also needs fewer nodes and shorter computing time on average than SEP1 and MIX++. Alg-E performs unsatisfactorily on two instances of this class: "20–20–15–5" and "20–20–15–10", with S-Q value of 1.11% and 47.01%. These two instances contain 15 variables and 20 constraints in the lower-level problem, which results in a large optimal solution set. When the enhanced branching idea is performed, some well-performed bilevel feasible solutions including the optimal solution are carved out, which causes poor performance of Alg-E.

Instance Set MIPLIB. Table 4 compares Alg-E with the two best-performing algorithms from the literature; namely "SEP2" by Fischetti et al. (2016)) and "MIX++" by Fischetti et al. (2017a), on the very hard MIPLIB class. Recall that this class contains some instances with up to 80,000 relaxation-problem variables, hence in many cases, the optimal solutions are still unknown. For these cases, we calculate the percentage value of solution quality based on the best obtained feasible solution value. For the three algorithms and for each instance, Table 4 reports the best obtained feasible solution, the computing time (in seconds), the number of branch-and-bound nodes, and the percentage value of solution quality (denoted as "S-Q"). In the last row, the average value of computing time, number of nodes, and S-Q for each algorithm is reported (if an algorithm cannot find a feasible solution within the time limit, the S-Q is calculated as 0).

We point out that Alg-E solves 25 instances to optimality, where six instances are indicated to be exactly solved to optimality by "OPT = 1", and obtains feasible solution for all the rest instances of this class. In particular, among the 30 instances in MIPLIB that have not been solved to optimality, Alg-E provides the best feasible solution for four instances (i.e., "cap6000–0.1", "cap6000–0.9", "p0282–0.9", "p0548–0.5"). Similar to the results for DENEGRE, the indicator "OPT" of the exactness of the output solution may be insensitive due to the sufficient (not necessary) condition of the optimality checking mechanism on the uniqueness of lower-level optimal solution. Table 4 shows that Alg-E needs the fewest nodes and shortest computing time on average, while achieves an average percentage value of solution quality of 94.29%. Alg-E performs badly on three instances of this class: "air05–0.5", "air05–0.9" and "nw04–0.5", with S-Q value of 59.12%, 51.53% and 32.73%. These instances contain a large number of variables and a relatively small number of constraints in the lower-level problem. For example, "nw04–0.5" contains 43,741 variables and 36 constraints in the lower-level problem. As explained in DENEGRE, such cases may result in a large set of optimal solution of the lower-level problem, which induces inferior performance with our algorithm.

Consequently, our algorithm based on the enhanced branching rule clearly shows its superiority on general large-sized BILP instances with a relatively complex lower-level problem, such as instances of MIPLIB and WANGXU-LARGE. For these instances, the enhanced branching idea can significantly reduce the size of the branching tree, while this may cause a cost of sacrificing the optimality. We can expect that when the existing algorithms cannot finish solving relatively large-sized BILP instances within a restricted computation time, our algorithm can provide a promising bilevel feasible solution within reasonable time.

For those interested in conducting comparative studies, we have uploaded our test instances and the source codes to https://person.zju.edu.cn/wangmingzheng#781824.

6. Conclusions

In this paper, we present an enhanced branch-and-bound algorithm for a class of BILP problems, where both the upper-level and the lower-level variables are bounded. We introduce an enhanced branching idea and propose an enhanced branching rule based on a benchmark branching rule presented in Xu and Wang (2014). Our algorithm may discard bilevel feasible solutions if the lower-level problem is not uniquely optimal, which may lead to sub-optimality in BILP. Nevertheless, we provide a reasonable global optimality checking mechanism which is sufficient but not necessary for BILP, adapted from a well-established sufficient-and-necessary condition on the solution uniqueness of linear programming. Our computational results show that the enhanced branching rule can achieve considerable speedup for the benchmark branching rule while the output solution can achieve satisfactory solution quality. We compare our algorithm with state-of-the-art algorithms from the literature on a testbed of general BILP instances with up to 80,000 relaxation-problem variables and 5000 lower-level constraints. In particular, our algorithm can achieve superiority on both algorithm speedup and solution quality for large-sized BILP instances with relatively complex lower-level problem.

In the future, we will investigate how our global optimality checking mechanism can be weaken to a sufficient and necessary condition. We will also improve our enhanced branching idea so as to promote the solution quality or even guarantee the exactness of the algorithm. In addition, it is worth further study on adapting the algorithm for large-scale real-world instances such as those arising in the bilevel bidding problem in electricity markets. Finally, it is interesting to investigate the actual implementation of state-of-the-art BILP algorithms for real-world instances generated by algebraic modeling tools.

Acknowledgements

The authors thank the associate editor and the three anonymous referees for their valuable comments that have substantially improved the paper. This research was supported by the Key Program of the NSFC under the Grant 71931009, the General Program of the NSFC under the Grant 71671023, National Science Foundation of US under the Grant 1761022, the Foundation for Innovative Research Groups of NSFC under the Grant 71421001 and the key projects of of International Cooperation and Exchanges NSFC under the Grant 72010107002.

Appendix. An application on a bilevel facility location problem using our algorithm in algebraic formulation

To demonstrate how our algorithm can be used in an algebraic formulation setting, we address a bilevel facility location problem (Zare et al., 2019) as an example, since this problem is modeled as a BILP problem. The problem can be described as follows.

A firm that produces a set of products given by $\mathcal{G} = \{1, .., G\}$ can place new facilities at the locations given by $I = \{1, \ldots, q\}$. The leader chooses the facilities placement, while the follower must determine the number of each product's demand that each facility processes. The firm incurs a cost of $a_i^{(1)}$ for each facility opened at

location $i \in I$, and incurs an opportunity cost of $a_i^{(2)}$ for each unused production capacity of any plant at location $i \in I$ after it is opened. The follower faces a cost of $c_i^{(1)}$ for using a unit of capacity at a facility at location $i \in I$, and a cost of $c_{ig}^{(2)}$ associated with the transportation of $g \in \mathcal{G}$ from a facility at location $i \in I$.

Let x_i be the number of facilities to open at location i, and let y_{ig} be the number of demand for product g that the plants at location i process. $\mathcal{Q}$ denotes the maximum number of facilities that can be opened at any given location. D_g denotes the demand for g, and r_{ig} is the units of capacity needed to make product g at a facility at location i, and C_i is the capacity of a plant at location i, then the bilevel facility location problem (BFLP) can be modeled as:

BFLP:

$$\min_x \ \zeta = \sum_{i\in I} a_i^{(1)} x_i + \sum_{i\in I} a_i^{(2)} \left(C_i x_i - \sum_{g\in\mathcal{G}} r_{ig} y_{ig} \right),$$

$$\text{s.t. } 0 \le x_i \le \mathcal{Q}, \ i \in I,$$

$$x_i \in \mathbb{Z}, \ i \in I,$$

$$y \in \underset{\hat{y}}{\operatorname{argmin}} \ \{ \sum_{i\in I} \sum_{g\in\mathcal{G}} \left(c_i^{(1)} r_{ig} + c_{ig}^{(2)} \right) \hat{y}_{ig} :$$

$$\text{s.t. } \sum_{i\in I} \hat{y}_{ig} \ge D_g, \ g \in \mathcal{G},$$

$$\sum_{g\in\mathcal{G}} r_{ig} \hat{y}_{ig} \le C_i x_i, \ i \in I,$$

$$\sum_{g\in\mathcal{G}} \hat{y}_{ig} \le \sum_{g\in\mathcal{G}} D_g x_i, \ i \in I,$$

$$\hat{y}_{ig} \ge 0, \ \hat{y}_{ig} \in \mathbb{Z}, \ i \in I, \ g \in \mathcal{G}. \}$$

Before implementing the algorithm, we should construct the following model.

The first model is the relaxation problem of BFLP, denoted as $\mathcal{R}$(BFLP), which contains both the upper-level and the lower-level constraints.

$\mathcal{R}$(**BFLP**) (the relaxation problem of BFLP):

$$\min_x \ \sum_{i\in I} a_i^{(1)} x_i + \sum_{i\in I} a_i^{(2)} \left(C_i x_i - \sum_{g\in\mathcal{G}} r_{ig} y_{ig} \right),$$

$$\text{s.t. } 0 \le x_i \le \mathcal{Q}, \ i \in I,$$

$$x_i \in \mathbb{Z}, \ i \in I,$$

$$\sum_{i\in I} y_{ig} \ge D_g, \ g \in \mathcal{G},$$

$$\sum_{g\in\mathcal{G}} r_{ig} y_{ig} \le C_i x_i, \ i \in I,$$

$$\sum_{g\in\mathcal{G}} y_{ig} \le \sum_{g\in\mathcal{G}} D_g x_i, \ i \in I,$$

$$y_{ig} \ge 0, \ y_{ig} \in \mathbb{Z}, \ i \in I, \ g \in \mathcal{G}.$$

The second model is the node problem of BFLP, denoted as $\mathcal{R}(\hat{l}, \hat{u}, \hat{w})$, which adds bound constraints to the relaxation problem of BFLP.

$\mathcal{R}(\hat{\boldsymbol{l}}, \hat{\boldsymbol{u}}, \hat{\boldsymbol{w}})$ (the node problem of BFLP)

$$\min_x \ \sum_{i\in I} a_i^{(1)} x_i + \sum_{i\in I} a_i^{(2)} \left(C_i x_i - \sum_{g\in\mathcal{G}} r_{ig} y_{ig} \right),$$

$$\text{s.t. } 0 \le x_i \le \mathcal{Q}, \ i \in I,$$

$$x_i \in \mathbb{Z}, \ i \in I,$$

$$\sum_{i\in I} y_{ig} \ge D_g, \ g \in \mathcal{G},$$

$$\sum_{g\in\mathcal{G}} r_{ig} y_{ig} \le C_i x_i, \ i \in I,$$

$$\sum_{g\in\mathcal{G}} y_{ig} \le \sum_{g\in\mathcal{G}} D_g x_i, \ i \in I,$$

$$y_{ig} \ge 0, \ y_{ig} \in \mathbb{Z}, \ i \in I, \ g \in \mathcal{G}.$$

$$\hat{l}_i \le C_i x_i \le \hat{u}_i, \ i \in I,$$

$$\hat{l}_{I+i} \le \sum_{g\in\mathcal{G}} D_g x_i \le \hat{u}_{I+i}, \ i \in I,$$

$$\sum_{i\in I} \sum_{g\in\mathcal{G}} \left(c_i^{(1)} r_{ig} + c_{ig}^{(2)} \right) \hat{y}_{ig} \le \hat{w}.$$

The third model, denoted as $\mathcal{Q}(x^R, y^L)$, uses the leader's objective function as the objective function. The constraints contain all the upper-level and lower-level constraints, together with a bound constraint on the related part of the upper-level decision variables. All the lower-level decision variables equal to y^L.

$\mathcal{Q}\left(\boldsymbol{x^R}, \boldsymbol{y^L}\right)$

$$\min_x \ \sum_{i\in I} a_i^{(1)} x_i + \sum_{i\in I} a_i^{(2)} \left(C_i x_i - \sum_{g\in\mathcal{G}} r_{ig} y_{ig}^L \right),$$

$$\text{s.t. } 0 \le x_i \le \mathcal{Q}, \ i \in I,$$

$$x_i \in \mathbb{Z}, \ i \in I,$$

$$C_i x_i \le C_i x_i^R, \ i \in I,$$

$$\sum_{g\in\mathcal{G}} D_g x_i \le \sum_{g\in\mathcal{G}} D_g x_i^R, \ i \in I,$$

$$\sum_{g\in\mathcal{G}} r_{ig} y_{ig}^L \le C_i x_i, \ i \in I,$$

$$\sum_{g\in\mathcal{G}} y_{ig}^L \le \sum_{g\in\mathcal{G}} D_g x_i, \ i \in I.$$

The fourth model, denoted as $\mathcal{U}(x^R, y^L)$, is used to propose the optimality checking mechanism, which uses the sufficient and necessary condition for linear programming to be uniquely optimal as constraints.

$\mathcal{U}\left(\boldsymbol{x^R}, \boldsymbol{y^L}\right)$

$$\max_y \ y^2,$$

$$\text{s.t.} \sum_{i\in I}\sum_{g\in\mathcal{G}}\left(c_i^{(1)}r_{ig}+c_{ig}^{(2)}\right)y_{ig}=0,$$

$$\sum_{i\in I}\hat{y}_{ig}\geq 0,\ g\in J,$$

$$\sum_{g\in\mathcal{G}}\hat{y}_{ig}\leq 0,\ i\in J,$$

$$\sum_{g\in\mathcal{G}}r_{ig}\hat{y}_{ig}\leq 0,\ i\in J,$$

$$J=\left\{j|\sum_{i\in I}\hat{y}_{ij}=D_j \text{ or } \sum_{g\in\mathcal{G}}r_{jg}\hat{y}_{jg}=C_jx_j \text{ or } \sum_{g\in\mathcal{G}}\hat{y}_{jg}=\sum_{g\in\mathcal{G}}D_gx_j\right\}.$$

We are now ready to use our branch-and-bound algorithm to solve the BFLP model. The detailed steps can be demonstrated as follows.

Initialization. Denote $x^*=\emptyset$, $y^*=\emptyset$, $\zeta^*=+\infty$, $N=1$, OPT $=1$, $z^1=\infty$, and $l_i^1=0$, $i=1,\ldots,2q$, $u_i^1=\mathcal{Q}C_i$, $i=1,\ldots,q$, $u_i^1=\mathcal{Q}\sum_{g\in\mathcal{G}}D_g$, $i=q+1,\ldots,2q$.

Step 1. For all $k\in\{1,\ldots,N\}$ such that $z^k\geq\zeta^*$ or $l^k\nleq u^k$, remove node k. Update N as the number of remaining nodes.

---

if $N=0$ **then**

  if $x^*\neq\emptyset$ **then**

    if OPT $=1$ **then**

      1(a) return (x^*,y^*,ζ^*) is an optimal solution to BFLP.

    else

      1(b) return (x^*,y^*,ζ^*) is a heuristically optimal solution to BFLP.

    end

  else

    if OPT $=1$ **then**

      1(c) return BFLP is infeasible.

    else

      1(d) return BFLP is heuristically infeasible.

    end

  end

else

  1(e) select a node k from $\{1,\ldots,N\}$, set $\left(\hat{l}=l^k,\hat{u}=u^k,\hat{w}=w^k\right)$, remove node k, reorder the remaining nodes from 1 to $N-1$, reduce N by 1, and go to Step 2.

end

Step 2. Solve $\mathcal{R}(\hat{l}, \hat{u}, \hat{w})$.

if $\mathcal{R}(\hat{l}, \hat{u}, \hat{w})$ is infeasible **then**

  2(a) go to Step 1.

else

  let (x^R, y^R) denote an optimal solution to $\mathcal{R}(\hat{l}, \hat{u}, \hat{w})$.

  if $\sum_{i\in I} a_i^{(1)} x_i^R + \sum_{i\in I} a_i^{(2)} \left(C_i x_i^R - \sum_{g\in\mathcal{G}} r_{ig} y_{ig}^R \right) \geq \zeta^*$ **then**

    2(b) go to Step 1.

  else

    2(c) go to Step 3.

  end

end

Step 3 (Lower level) Solve $\mathcal{L}(x^R)$
Let y^L denote an optimal solution to $\mathcal{L}(x^R)$.

if $\sum_{i\in I}\sum_{g\in\mathcal{G}} \left(c_i^{(1)} r_{ig} + c_{ig}^{(2)} \right) y_{ig}^R = \sum_{i\in I}\sum_{g\in\mathcal{G}} \left(c_i^{(1)} r_{ig} + c_{ig}^{(2)} \right) y_{ig}^L$ **then**

  3(a) update $\left(x^* = x^R, y^* = y^R, \zeta^* = \sum_{i\in I} a_i^{(1)} x_i^R + \sum_{i\in I} a_i^{(2)} \left(C_i x_i^R - \sum_{g\in\mathcal{G}} r_{ig} y_{ig}^R \right) \right)$ and go to Step 1.

else

  if $\mathcal{Q}(x^R, y^L)$ is optimal (denote the optimal solution as x^Q), and $\sum_{i\in I} a_i^{(1)} x_i^Q + \sum_{i\in I} a_i^{(2)} \left(C_i x_i^Q - \sum_{g\in\mathcal{G}} r_{ig} y_{ig}^L \right) < \zeta^*$, **then**

    update $x^* = x^Q, y^* = y^L, \zeta^* = \sum_{i\in I} a_i^{(1)} x_i^Q + \sum_{i\in I} a_i^{(2)} \left(C_i x_i^Q - \sum_{g\in\mathcal{G}} r_{ig} y_{ig}^L \right)$

    if $\sum_{i\in I} a_i^{(1)} x_i^R + \sum_{i\in I} a_i^{(2)} \left(C_i x_i^R - \sum_{g\in\mathcal{G}} r_{ig} y_{ig}^R \right) \geq \zeta^*$ **then**

      3(b) go to Step 1.

    end

  end

  if OPT =1 and $\mathcal{U}(x^R, y^L)$ has an optimal solution of 0 **then**

    update OPT =1.

  else

    update OPT = 0.

  end

  3(c) go to Step 4.

end

Step 4.

if $\exists p \in \{1,2,\dots,q\}$ s.t. $\sum_{g\in\mathcal{G}} r_{pg} y^L_{pg} = \hat{l}_p \le \hat{u}_p \le C_p x^R_p$ or $\exists p \in \{q+1, q+2, \dots, 2q\}$ s.t. $\sum_{g\in\mathcal{G}} y^L_{p-q,g} = \hat{l}_p \le \hat{u}_p \le \sum_{g\in\mathcal{G}} r_{p-q,g} x^R_{p-q}$ **then**

4(a) go to Step 1.

else

if $\exists p \in \{1,2,\dots,q\}$ s.t. $\sum_{g\in\mathcal{G}} r_{pg} y^L_{pg} = \hat{l}_p \le C_p x^R_p \le \hat{u}_p - 1$ or $\exists p \in \{q+1, q+2, \dots, 2q\}$ s.t. $\sum_{g\in\mathcal{G}} y^L_{p-q,g} = \hat{l}_p \le \sum_{g\in\mathcal{G}} r_{p-q,g} x^R_{p-q} \le \hat{u}_p - 1$, **then**

find all the subscript $v_1 < v_2 < \cdots < v_H$ such that $\sum_{g\in\mathcal{G}} r_{v_h,g} y^L_{v_h,g} = \hat{l}_{v_h} \le C_{v_h} x^R_{v_h} \le \hat{u}_{v_h} - 1$ and $\sum_{g\in\mathcal{G}} y^L_{v_h-q,g} = \hat{l}_{v_h} \le \sum_{g\in\mathcal{G}} r_{v_h-q,g} x^R_{v_h-q} \le \hat{u}_{v_h} - 1$, $h = 1, \dots, H$.

else

let $v_1 = 2q$, $H = 1$.

end

if $\exists j \in \{1,2,\dots,q\}$ s.t. $\hat{l}_j + 1 \le \sum_{g\in\mathcal{G}} r_{jg} y^L_{jg} \le \hat{u}_j \le C_j x^R_j$ or $\exists j \in \{q+1, q+2, \dots, 2q\}$ s.t. $\hat{l}_j + 1 \le \sum_{g\in\mathcal{G}} y^L_{j-q,g} \le \hat{u}_j \le \sum_{g\in\mathcal{G}} r_{j-q,g} x^R_{j-q}$, **then**

find all the subscript $s_1 < s_2 < \cdots < s_T$ such that $\hat{l}_{s_t} + 1 \le \sum_{g\in\mathcal{G}} r_{s_t,g} y^L_{s_t,g} \le \hat{u}_{s_t} \le C_{s_t} x^R_{s_t}$, and $\hat{l}_{s_t} + 1 \le \sum_{g\in\mathcal{G}} y^L_{s_t-q,g} \le \hat{u}_{s_t} \le \sum_{g\in\mathcal{G}} r_{s_t-q,g} x^R_{s_t-q}$, $t = 1, \dots, T$, and count the number of $\{s_1, s_2, \dots, s_T\}$, such that $s_t \le v_1$, and denote the number as r.

else

let $s_1 = 2q + 1$, $r = 0$.

end

4(b) go to Step 5

end

Step 5. Create $(v_1 - r)$ new node problems, increase N by $(v_1 - r)$, and go to Step 1. For $k = 1, \dots, v_1$, $k \ne s_t$, $t = 1, \dots, r$, node $(N + k)$ is characterized by $(l^{N+k}, u^{N+k}, w^{N+k}, z^{N+k})$, which is defined as

$$l^{N+k}_j = \begin{cases} \sum\limits_{g\in\mathcal{G}} r_{jg} y^L_{jg} + 1 & \text{if } j = k, v_1, \dots, v_H, \text{ and } j \le q, \\ \sum\limits_{g\in\mathcal{G}} y^L_{j-q,g} & \text{if } j = k, v_1, \dots, v_H, \text{ and } j > q; \\ \hat{l}_j & \text{otherwise;} \end{cases}$$

$$u^{N+k}_j = \begin{cases} \min\left\{\sum_{g\in\mathcal{G}} r_{jg} y^L_{jg}, \hat{u}_j\right\} & \text{if } j = 1, \dots, k-1, j \notin \{s_1, \dots, s_T\}, \text{ and } j \le q, \\ \min\left\{\sum_{g\in\mathcal{G}} y^L_{j-q,g}, \hat{u}_j\right\} & \text{if } j = 1, \dots, k-1, j \notin \{s_1, \dots, s_T\}, \text{and } j > q, \\ C_j x^R_j - 1 & \text{if } j = s_1, \dots, s_T, \text{ and } j \le q, \\ \sum_{g\in\mathcal{G}} r_{j-q,g} x^R_{j-q} & \text{if } j = s_1, \dots, s_T, \text{ and } j > q, \\ \hat{u}_j & \text{otherwise;} \end{cases}$$

$$w^{N+k} = \begin{cases} \sum\limits_{i\in I} \sum\limits_{g\in\mathcal{G}} \left(c^{(1)}_i r_{ig} + c^{(2)}_{ig}\right) y^L_{ig} & \text{if } k = 2q + 1, \\ \hat{w} & \text{otherwise;} \end{cases}$$

$$z^{N+k} = \sum_{i\in I} a^{(1)}_i x^R_i + \sum_{i\in I} a^{(2)}_i \left(C_i x^R_i - \sum_{g\in\mathcal{G}} r_{ig} y^L_{ig}\right).$$

References

Avraamidou, S., & Pistikopoulos, E. N. (2019). A Multi-Parametric optimization approach for bilevel mixed-integer linear and quadratic programming problems. *Computers and Chemical Engineering, 125*, 98–113.

Bard, J. F., & Moore, J. T. (1992). An algorithm for the discrete bilevel programming problem. *Naval Research Logistics (NRL), 39*(3), 419–435.

Bixby, R. E., Ceria, S., McZeal, C. M., & Savelsbergh, M. W. P. (1998). An updated mixed integer programming library: MIPLIB 3.0. *Optima, 58*, 12–15.

Brotcorne, L., Labbé, M., Marcotte, P., & Savard, G. (2001). A bilevel model for toll optimization on a multicommodity transportation network. *Transportation science, 35*(4), 345–358.

Cao, D., & Chen, M. (2006). Capacitated plant selection in a decentralized manufacturing environment: A bilevel optimization approach. *European Journal of Operational Research, 169*(1), 97–110.

Caprara, A., Carvalho, M., Lodi, A., & Woeginger, G. J. (2016). Bilevel knapsack with interdiction constraints. *INFORMS Journal on Computing, 28*(2), 319–333.

Caramia, M., & Mari, R. (2015). Enhanced exact algorithms for discrete bilevel linear problems. *Optimization Letters, 9*(7), 1447–1468.

Caramia, M., & Mari, R. (2016). A decomposition approach to solve a bilevel capacitated facility location problem with equity constraints. *Optimization Letters, 10*(5), 997–1019.

Ceylan, H., & Bell, M. G. H. (2004). Traffic signal timing optimization based on genetic algorithm approach, including drivers' routing. *Transportation Research Part B: Methodological, 38*(4), 329–342.

Croce, F. D., & Scatamacchia, R. (2019). Lower Bounds and a New Exact Approach for the Bilevel Knapsack with Interdiction Constraints. *Integer Programming and Combinatorial Optimization*, 155–167.

Croce, F. D., & Scatamacchia, R. (2020). An exact approach for the bilevel knapsack problem with interdiction constraints and extensions. *Mathematical Programming*, 1–33.

Dempe, S., Kalashnikov, V., Pérez-Valdés, G. A., & Kalashnykova, N. I. (2011). Natural gas bilevel cash-out problem: Convergence of a penalty function method. *European Journal of Operational Research, 215*(3), 532–538.
Dempe, S., Kalashnikov, V., & RíÔs-Mercado, R. Z. (2005). Discrete bilevel programming: Application to a natural gas cash-out problem. *European Journal of Operational Research, 166*(2), 469–488.
Dempe, S., Mordukhovich, B. S., & Zemkoho, A. B. (2014). Necessary optimality conditions in pessimistic bilevel programming. *Optimization, 63*(4), 505–533.
DeNegre, S. (2011). *Interdiction and discrete bilevel linear programming* PhD thesis. Bethlehem, PA: Lehigh University.
DeNegre, S. T., & Ralphs, T. K. (2009). A branch-and-cut algorithm for integer bilevel linear programs. *Operations research and cyber-infrastructure* (pp. 65–78). Boston, MA: Springer.
Domínguez, L. F., & Pistikopoulos, E. N. (2010). Multiparametric programming based algorithms for pure integer and mixed-integer bilevel programming problems. *Computers and Chemical Engineering, 34*(12), 2097–2106.
Faísca, N. P., Dua, V., Rustem, B., Saraiva, P. M., & Pistikopoulos, E. N. (2007). Parametric global optimisation for bilevel programming. *Journal of Global Optimization, 38*(4), 609–623.
Fischetti, M., Ljubić, I., Monaci, M., & Sinnl, M. (2016). Intersection cuts for bilevel optimization. In Q. Louveaux, & M. Skutella (Eds.), *Proceedings of the 18th international conference integer programming combinatorial optimization, IPCO* (pp. 77–88). ChamSwitzerland: Springer International.
Fischetti, M., Ljubić, I., Monaci, M., & Sinnl, M. (2017a). A new general-purpose algorithm for mixed-integer bilevel linear programs. *Operations Research, 65*(6), 1615–1637.
Fischetti, M., Ljubić, I., Monaci, M., & Sinnl, M. (2017b). Instances and solver software for mixed-integer bilevel linear problems. Accessed March 2017. https://msinnl.github.io/pages/bilevel.html
Fischetti, M., Ljubić, I., Monaci, M., & Sinnl, M. (2018). On the use of intersection cuts for bilevel optimization. *Mathematical Programming, 172*(1–2), 77–103.
Florensa, C., Garcia-Herreros, P., Misra, P., Arslan, E., Mehta, S., & Grossmann, I. E. (2017). Capacity planning with competitive decision-makers: Trilevel MILP formulation, degeneracy, and solution approaches. *European Journal of Operational Research, 262*(2), 449–463.
Garcia-Herreros, P., Zhang, L., Misra, P., Arslan, E., Mehta, S., & Grossmann, I. E. (2016). Mixed-integer bilevel optimization for capacity planning with rational markets. *Computers and Chemical Engineering, 86*, 33–47.
Hemmati, M., & Smith, J. C. (2016). A mixed-integer bilevel programming approach for a competitive prioritized set covering problem. *Discrete Optimization, 20*, 105–134.
Jeroslow, R. G. (1985). The polynomial hierarchy and a simple model for competitive analysis. *Mathematical Programming, 32*(2), 146–164.
Kalashnikov, V. V., Pérez, G. A., & Kalashnykova, N. I. (2010). A linearization approach to solve the natural gas cash-out bilevel problem. *Annals of Operations Research, 181*(1), 423–442.
Kalashnikov, V. V., & Ríos-Mercado, R. Z. (2006). A natural gas cash-out problem: A bilevel programming framework and a penalty function method. *Optimization and Engineering, 7*(4), 403–420.
Labbé, M., Marcotte, P., & Savard, G. A. (1988). Bilevel model of taxation and its application to optimal highway pricing. *Management science, 44*(12-part-1), 1608–1622.
Lozano, L., & Smith, J. C. (2016). A backward sampling framework for interdiction problems with fortification. *INFORMS Journal on Computing, 29*(1), 123–139.
Lozano, L., & Smith, J. C. (2017). A value-function-based exact approach for the bilevel mixed-integer programming problem. *Operations Research, 65*(3), 768–786.
Mangasarian, O. L. (1979). Uniqueness of solution in linear programming. *Linear Algebra and Its Applications, 25*(none), 151–162.
Moore, J. T., & Bard, J. F. (1990). The mixed integer linear bilevel programming problem. *Operations research, 38*(5), 911–921.
Saharidis, G. K., & Ierapetritou, M. G. (2009). Resolution method for mixed integer bi-level linear problems based on decomposition technique. *Journal of Global Optimization, 44*(1), 29–51.
Tahernejad, S., Ralphs, T.K., .& DeNegre, S.T. (.2016). A branch-and-cut algorithm for mixed integer bilevel linear optimization problems and its implementation. COR@ L Laboratory Technical Report 16T-015-R3, Lehigh University.
Tang, Y., Richard, J. P. P., & Smith, J. C. (2016). A class of algorithms for mixed-integer bilevel min–max optimization. *Journal of Global Optimization, 66*(2), 225–262.
Vicente, L., Savard, G., & Judice, J. (1996). Discrete linear bilevel programming problem. *Journal of Optimization Theory and Applications, 89*(3), 597–614.
Wang, L., & Xu, P. (2017). The watermelon algorithm for the bilevel integer linear programming problem. *SIAM Journal on Optimization, 27*(3), 1403–1430.
Wiesemann, W., Tsoukalas, A., Kleniati, P. M., & Rustem, B. (2013). Pessimistic bilevel optimization. *SIAM Journal on Optimization, 23*(1), 353–380.
Xu, P., & Wang, L. (2014). An exact algorithm for the bilevel mixed integer linear programming problem under three simplifying assumptions. *Computers and Operations Research, 41*, 309–318.
Yue, D., Gao, J., Zeng, B., & You, F. (2019). A projection-based reformulation and decomposition algorithm for global optimization of a class of mixed integer bilevel linear programs. *Journal of Global Optimization, 73*(1), 27–57.
Zare, M. H., Borrero, J. S., Zeng, B., & Prokopyev, O. A. (2019). A note on linearized reformulations for a class of bilevel linear integer problems. *Annals of Operations Research, 272*(1–2), 99–117.
Zeng, B., & An, Y. (2014). Solving bilevel mixed integer program by reformulations and decomposition. *Optimization Online*, 1–34.
Zhang, J., & Özaltın, O. Y. (2017). A branch-and-cut algorithm for discrete bilevel linear programs. *Optimization Online*.