SIAM J. OPTIM.
Vol. 27, No. 3, pp. 1403–1430

THE WATERMELON ALGORITHM FOR THE BILEVEL INTEGER LINEAR PROGRAMMING PROBLEM*

LIZHI WANG† AND PAN XU‡

Abstract. This paper presents an exact algorithm for the bilevel integer linear programming (BILP) problem. The proposed algorithm, which we call the watermelon algorithm, uses a multiway disjunction cut to remove bilevel infeasible solutions from the search space, which was motivated by how watermelon seeds can be carved out by a scoop. Serving as the scoop, a polyhedron is designed to enclose as many bilevel infeasible solutions as possible, and then the complement of this polyhedron is applied to the search space as a multiway disjunction cut in a branch-and-bound framework. We have proved that the watermelon algorithm is able to solve all BILP instances finitely and correctly, providing either a global optimal solution or a certificate of infeasibility or unboundedness. Computational experiment results on two sets of small- to medium-sized instances suggest that the watermelon algorithm could be significantly more efficient than previous branch-and-bound based BILP algorithms.

Key words. bilevel optimization, cutting plane, branch and bound

AMS subject classifications. 90C05, 90C10, 90C99

DOI. 10.1137/15M1051592

1. Introduction.

We present an exact algorithm, which we call the watermelon algorithm, for the bilevel integer linear programming (BILP) problem defined as follows:

$$\max_{x,y} \zeta = c^\top x + d_1^\top y \tag{1.1}$$

$$\text{s.t. } A_1 x + B_1 y \le b_1, \tag{1.2}$$

$$x \in \mathbb{Z}^{n_1}, \tag{1.3}$$

$$y \in \arg\max_{\tilde{y}} \{d_2^\top \tilde{y} : A_2 x + B_2 \tilde{y} \le b_2; \tilde{y} \in \mathbb{Z}^{n_2}\}, \tag{1.4}$$

where $A_1 \in \mathbb{Q}^{m_1 \times n_1}$, $A_2 \in \mathbb{Z}^{m_2 \times n_1}$, $B_1 \in \mathbb{Q}^{m_1 \times n_2}$, $B_2 \in \mathbb{Z}^{m_2 \times n_2}$, $b_1 \in \mathbb{Q}^{m_1}$, $b_2 \in \mathbb{Z}^{m_2}$, $c \in \mathbb{Q}^{n_1}$, $d_1 \in \mathbb{Q}^{n_2}$, and $d_2 \in \mathbb{Z}^{n_2}$ are rational or integer parameters. With four lower-level parameters (A_2, B_2, b_2, and d_2) being restricted to integers rather than rational numbers, this definition has no loss of generality, since rational parameters can be equivalently converted to integers by multiplication.

BILP problems arise in many real world situations where two entities make decisions in a hierarchical manner. The lower level controls variable y and solves a parametric integer linear program (ILP) whose right-hand side is influenced by the upper level. The upper level directly controls variable x and anticipates and indirectly controls the lower level's optimal solution y in response to x. As formulation (1.1)–(1.4) indicates, when the lower level possesses multiple optimal solutions, the upper level gets to pick. This is commonly known as the optimistic formulation. In contrast, under the pessimistic formulation, when the lower level possesses multiple optimal

*Received by the editors December 8, 2015; accepted for publication (in revised form) February 23, 2017; published electronically July 26, 2017.
http://www.siam.org/journals/siopt/27-3/M105159.html
Funding: This research was partially supported by the National Science Foundation under grant EFRI-0835989.
†Iowa State University, Ames, IA 50011 (lzwang@iastate.edu).
‡University of Maryland, College Park, MD 20742 (panxu@cs.umd.edu).

solutions, its decision maker will pick the one that either violates the upper level constraint or otherwise makes the least contribution to the upper level objective function. A more detailed discussion regarding these two formulations can be found in [45].

BILP and related bilevel discrete optimization models have found a wide spectrum of real world applications, including large steel structure design [40], blended composite structures [47], competitive location or capacity planning [15, 29, 18], natural gas cash-out [10], network design [17], taxation policy [21, 48], robust unit commitment [4, 24], production planning [32], traffic calming [38], and critical infrastructure protection [6, 7, 31, 41, 46]. In these and many other applications, discrete variables are indispensable to model logical decisions, mutually exclusive choices, fixed costs, etc.

2. Literature review. Studies on bilevel optimization algorithms date back to the 1980s [2, 5]. Earlier work focused primarily on continuous bilevel linear problems. Without the integrality requirement, especially on the lower level decision variables, constraint (1.4) can be replaced by two constraints: $B_2^\top \mu = d_2$ and $0 \le b_2 - A_2 x - B_2 y \perp \mu \ge 0$, where μ is the dual variable of constraint $A_2 x + B_2 \tilde{y} \le b_2$. The resulting formulation is a linear program with linear complementary constraints, which has been well studied and is solvable by a variety of algorithms, including branch-and-bound [3, 20], branch-and-cut [1], Benders decomposition [23], parametric [14], semi-infinite [42], active-set [25], and simplex [37] approaches.

No sooner had research effort been extended to discrete bilevel optimization problems than ill-conditioned cases were discovered [35]: If the upper level contains continuous variables and the lower level contains discrete ones, then the supremum of the bilevel problem may not be attainable even if it is finite. Köppe et al. [28] presented such an example and, under a few simplifying assumptions, proved the existence of an algorithm that is able to solve such problems to either global optimality or ϵ-optimality if the supremum is unattainable.

The majority of the work on discrete bilevel linear optimization focused on the well-defined yet notoriously challenging case of the bilevel mixed integer linear program (BMILP) with pure integer variables at the upper level and either mixed integers or pure integers at the lower level. Moore and Bard [35] invented the first branch-and-bound algorithm for BMILP and proved its finite termination and correctness under a few simplifying assumptions. A branch-and-cut algorithm was proposed by DeNegre and Ralphs [11], who redesigned the relaxation and used cutting planes to produce tighter bounds. Mitsos [34] presented another algorithm for a more general bilevel mixed nonlinear program, in which no branching is performed; rather, the upper bound (for a maximization problem) is iteratively updated by solving a mixed-integer nonlinear program. Another approach was proposed by Gümüş and Floudas [19], which reformulates the lower level mixed-integer problem as a continuous one via its vertex polyhedral convex hull representation and then replaces it with its KKT conditions. As a result, the bilevel problem is converted into a much larger single level mixed integer nonlinear program. Domínguez and Pistikopoulo [13] introduced two multiparametric algorithms for BILP and BMILP. Both algorithms embed multiparametric solutions in the upper level problem to form a set of mixed integer nonlinear programming problems, which are then solved to global optimality using standard fixed-point optimization methods. Genetic algorithms [30, 36] have also been investigated. In a previous study [45], we designed an exact algorithm for BMILP with an additional simplifying assumption that the upper level variables have known finite bounds. More recently, Caprara et al. [8] proposed an exact algorithm for solving the bilevel knapsack problem with interdiction constraints. The algorithm

uses a new method to enumerate solutions and introduces a strong cut for the upper level by exploiting the structural properties of the problem. Hemmati and Smith [22] formulated a competitive set covering problem as a BMILP and developed a cutting-plane algorithm "based on inequalities that support the convex hull of feasible solutions." Caramia and Mari [9] proposed two algorithms for BILP. The first one is a cutting plane method, in which a nonlinear valid inequality was introduced and then approximated with a linear cut by solving an auxiliary problem. The second one is a branch-and-cut algorithm motivated by some geometrical properties of the bilevel linear problem. Fischetti et al. [16] presented intersection cuts and informed no-good cuts for BMILP, which could be used to enhance the performance of existing branch-and-cut algorithms.

Inspired by these previous works, the watermelon algorithm presents a new approach that integrates branch-and-bound and cutting plane techniques. It follows a commonly used algorithmic framework: iteratively solving the relaxation and then removing its optimal solution if it is bilevel infeasible. The novelty lies in the way the relaxed solution is removed from the search space. We design a polyhedron to enclose not only the relaxation solution but also many other bilevel infeasible solutions, and then remove this polyhedron from the search space by applying a multiway disjunction cut, which is carried out in a branch-and-bound framework.

3. Prerequisite. We make several definitions and clarifications to facilitate the introduction of the algorithm. We define the set $\overline{\mathbb{R}} = \mathbb{R} \cup \{+\infty\} \cup \{-\infty\}$ as the extended real number line including positive and negative infinity. We use a vector with a subscript j to refer to the jth element of the vector. For any given $x \in \mathbb{Z}^{n_1}$, we use $\mathcal{L}(x)$ to denote the lower level problem:

$$\max_{\tilde{y}} \{d_2^\top \tilde{y} : A_2 x + B_2 \tilde{y} \leq b_2; \tilde{y} \in \mathbb{Z}^{n_2}\}.$$

For any given $x \in \mathbb{Z}^{n_1}$, we define $\Psi_0(x)$ as the set of optimal solutions to $\mathcal{L}(x)$:

$$\Psi_0(x) = \arg\max_{\tilde{y}} \{d_2^\top \tilde{y} : A_2 x + B_2 \tilde{y} \leq b_2; \tilde{y} \in \mathbb{Z}^{n_2}\}.$$

If $\mathcal{L}(x)$ is infeasible or unbounded, then an optimal solution does not exist in the real number space, so $\Psi_0(x)$ is an empty set. We define Ψ_1 as the set of bilevel feasible solutions to the BILP (1.1)–(1.4):

$$\Psi_1 = \{(x, y) : x \in \mathbb{Z}^{n_1}; y \in \Psi_0(x); A_1 x + B_1 y \leq b_1\}.$$

As such, $\Psi_0(x)$ and Ψ_1 are subsets of $\mathbb{Z}^{n_2}$ and $\mathbb{Z}^{n_1} \times \mathbb{Z}^{n_2}$, respectively. By definition, the BILP (1.1)–(1.4) can be written as

$$\max_{x,y} \{c^\top x + d_1^\top y : (x, y) \in \Psi_1\}. \tag{3.1}$$

A solution (x, y) is called lower level optimal if $y \in \Psi_0(x)$. A solution (x, y) is called bilevel feasible if $(x, y) \in \Psi_1$. 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 $c^\top x^* + d_1^\top y^* \geq c^\top x^0 + d_1^\top y^0$ for any other bilevel feasible solution $(x^0, y^0) \in \Psi_1$. For all BILP instances conforming to the definition of (1.1)–(1.4), there are only three possible outcomes: optimal, infeasible, or unbounded. A BILP instance is called optimal if a finite bilevel optimal solution exists (uniquely or not). A BILP instance is called infeasible if no bilevel feasible solution exists. A BILP

instance is called unbounded if for any real number K there exists a bilevel feasible solution (x^K, y^K) such that $c^\top x^K + d_1^\top y^K \geq K$, implying that y^K is a (finite) optimal solution to $\mathcal{L}(x^K)$.

The following ILP is a relaxation of BILP (1.1)–(1.4), which was referred to as the high point problem in [35]:

$$\max_{x,y} \zeta = c^\top x + d_1^\top y \tag{3.2}$$

$$\text{s.t. } A_1 x + B_1 y \leq b_1, \tag{3.3}$$

$$A_2 x + B_2 y \leq b_2, \tag{3.4}$$

$$x_i \in \mathbb{Z} \ \ \forall i \in I, \ y_j \in \mathbb{Z} \ \forall j \in J. \tag{3.5}$$

Depending on the outcomes of the BILP (1.1)–(1.4), its relaxation (3.2)–(3.5) and, when relevant, the existence of known finite bounds on decision variables, there are seven types of BILP instances. One example of each type is provided as follows, which will be referenced in the forthcoming sections. In particular, Examples 2 and 3 differ in the knowledge of upper bounds on x and y.

Example 1. The following BILP is infeasible because constraints $y = 1$ in (3.7) and $y = 2$ in (3.8) are incompatible. The relaxation is also infeasible.

$$\max_{x,y} x - y \tag{3.6}$$

$$\text{s.t. } x \leq 1, \ y = 1, \ x \in \mathbb{Z}, \tag{3.7}$$

$$y \in \arg\max_{\tilde{y}} \{0 : \tilde{y} = 2\}. \tag{3.8}$$

Example 2. The following BILP instance is adopted from [45]. The BILP is infeasible, because no matter what positive integer value x takes, the lower level imposes $y = x$, which is incompatible with the upper level constraints $x \geq 1$ and $y \geq 2x$. The relaxation has an optimal solution, but decision variables do not have finite upper bounds:

$$\begin{aligned} &\max_{x,y} x - y \\ &\text{s.t. } x \geq 1, \ y \geq 2x, \ x \in \mathbb{Z}, \\ &\quad y \in \arg\max_{\tilde{y}} \{-\tilde{y} : \tilde{y} \geq x; \tilde{y} \geq 0; \tilde{y} \in \mathbb{Z}\}. \end{aligned}$$

Example 3. The following BILP instance is Example 2 with finite upper bounds on x and y. This BILP is infeasible and its relaxation has an optimal solution:

$$\begin{aligned} &\max_{x,y} x - y \\ &\text{s.t. } 1 \leq x \leq 10, \ y \geq 2x, \ x \in \mathbb{Z}, \\ &\quad y \in \arg\max_{\tilde{y}} \{-\tilde{y} : \tilde{y} \geq x; 0 \leq \tilde{y} \leq 10; \tilde{y} \in \mathbb{Z}\}. \end{aligned}$$

Example 4. This BILP instance is from [35] (without the redundant constraint $y \geq 0$) and can be defined by the following parameters. The bilevel optimal solution is $(x^* = 2, y^* = 2)$. The relaxation has an optimal solution

$$A_1 = -1, \quad B_1 = 0, \quad b_1 = 0, \quad c = 1, \quad d_1 = 10, \quad d_2 = -1,$$

$$A_2 = \begin{bmatrix} -5 \\ 1 \\ 2 \\ -2 \end{bmatrix}, \quad B_2 = \begin{bmatrix} 4 \\ 2 \\ -1 \\ -10 \end{bmatrix}, \quad b_2 = \begin{bmatrix} 6 \\ 10 \\ 15 \\ -15 \end{bmatrix}.$$

Example 5. The following BILP is infeasible because the lower level is unbounded for all x; thus, a solution (x, y) that satisfies Constraint (3.11) does not exist. The relaxation is unbounded:

$$\max_{x,y} \zeta = x + y \tag{3.9}$$

$$\text{s.t. } -2 \le x \le 2,\ x \in \mathbb{Z}, \tag{3.10}$$

$$y \in \arg\max_{\tilde{y}} \{\tilde{y} : 3x - 2\tilde{y} \le 0; -3x - 2\tilde{y} \le 0; \tilde{y} \in \mathbb{Z}\}. \tag{3.11}$$

Example 6. The following BILP has an optimal solution ($x^* = 2, y^* = 3$). The relaxation is unbounded:

$$\begin{aligned} &\max_{x,y} x + y \\ &\text{s.t. } -2 \le x \le 2,\ x \in \mathbb{Z}, \\ &\quad y \in \arg\max_{\tilde{y}} \{-\tilde{y} : 3x - 2\tilde{y} \le 0; -3x - 2\tilde{y} \le 0; \tilde{y} \in \mathbb{Z}\}. \end{aligned}$$

Example 7. The following BILP is unbounded, because for any integer $K \ge 0$, $(x = 0, y = K)$ is a bilevel feasible solution with $\zeta = K$. The relaxation is unbounded:

$$\begin{aligned} &\max_{x,y} \zeta = x + y \\ &\text{s.t. } -2 \le x \le 2,\ x \in \mathbb{Z}, \\ &\quad y \in \arg\max_{\tilde{y}} \{0 : 3x - 2\tilde{y} \le 0; -3x - 2\tilde{y} \le 0; \tilde{y} \in \mathbb{Z}\}. \end{aligned}$$

An infeasible BILP instance could result from three possibilities. First, as demonstrated in Example 1, lower and upper level constraints are over-restrictive: $\{(x, y) : A_1x + B_1y \le b_1; A_2x + B_2y \le b_2\} = \emptyset$. Second, as demonstrated in Examples 2 and 3, the lower level optimality and upper level feasibility constraints are incompatible: $\Psi_0(x) \cap \{y : A_1x + B_1y \le b_1\} = \emptyset\ \forall x$. Third, as demonstrated in Example 5, the lower level is unbounded. Since the lower level ILP is defined in the real number space, which does not include infinity, an unbounded lower level means that the set $\Psi_0(x)$ is empty for any x and that the BILP is infeasible. The following lemma proves this point.

LEMMA 3.1. *If there exists an $x \in \mathbb{Z}^{n_1}$ such that $\mathcal{L}(x)$ is unbounded, then the BILP* (1.1)–(1.4) *is infeasible.*

Proof. If $\mathcal{L}(x)$ is unbounded for some x, then $\mathcal{L}(x)$ must possess a ray Δy such that $\Delta y \in \mathbb{Z}^{n_2}$, $B_2\Delta y \le 0$, and $d_2^\top \Delta y \ge 1$, which means that $\Psi_0(x) = \emptyset$. Since Δy is independent of x, it exists for all x, and $\Psi_0(x) = \emptyset$ for all x. Therefore, the BILP (1.1)–(1.4) is infeasible. ∎

4. Motivation. The watermelon algorithm takes advantage of the bilevel structure of BILP by iteratively using a polyhedron to remove bilevel infeasible solutions from the search space. This idea was motivated by how watermelon seeds can be carved out by a scoop, which is why the algorithm was metaphorically named after the fruit.

To facilitate the presentation of the algorithm, we coined figurative names to refer to several mathematical terms. We refer to the polyhedron $\mathcal{W}_0 = \{(x, y) : A_1x + B_1y \le b_1; A_2x + B_2y \le b_2\}$ as *the watermelon*. This polyhedron was called the inducible region in [35]. For a given set of parameters $l, u \in \overline{\mathbb{R}}^{m_2}$, we refer to $\mathcal{W}(l, u) = \{(x, y) : A_1x + B_1y \le b_1; l \le A_2x + B_2y \le u\}$ as *a slice of watermelon* or

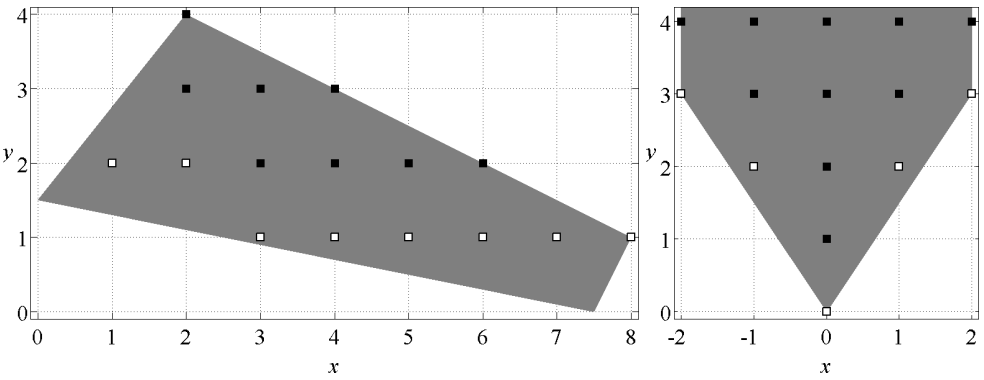

FIG. 1. *Illustration of definitions of the watermelon (shaded polyhedra), bilevel feasible solutions (white squares), and watermelon seeds (black squares) for Examples* 4 *(left) and* 6 *(right). The watermelon for Example* 6 *is unbounded from above.*

a watermelon slice, which is also a polyhedron obtained by "chopping the head and tail off of" the watermelon if $u \leq b_2$ (which is always the case in the algorithm). As a special case, we have $\mathcal{W}_0 = \mathcal{W}(\{-\infty\}^{m_2}, b_2)$. We refer to a subset of the watermelon or a watermelon slice as the *watermelon flesh*. We define $\Psi_2 = \mathcal{W}_0 \cap (\mathbb{Z}^{n_1} \times \mathbb{Z}^{n_2}) \setminus \Psi_1$ and refer to this set as *watermelon seeds*, which are integer points in the watermelon that are bilevel infeasible. As such, Ψ_1 and Ψ_2 are a partition of the set $\mathcal{W}_0 \cap (\mathbb{Z}^{n_1} \times \mathbb{Z}^{n_2})$. A solution $(x^0, y^0) \in \mathcal{W}_0 \cap (\mathbb{Z}^{n_1} \times \mathbb{Z}^{n_2})$ is a watermelon seed if and only if $y^0 \notin \Psi_0(x^0)$. We refer to a polyhedron that does not contain bilevel feasible solutions as a *scoop*. The term scoop is also used as a verb, meaning to exclude a solution or a set of solutions from the search space.

We illustrate the definitions of $\mathcal{W}_0$, Ψ_1, and Ψ_2 on Examples 4 and 6 in Figure 1. For both examples, the shaded polyhedra are the watermelon $\mathcal{W}_0$, the white squares are the set of bilevel feasible solutions Ψ_1, and the black squares are the set of watermelon seeds Ψ_2.

Using the aforementioned definitions, we can equivalently reformulate the BILP (3.1) as

$$\max_{x,y} \left\{ c^\top x + d_1^\top y : (x, y) \in \mathcal{W}_0 \cap (\mathbb{Z}^{n_1} \times \mathbb{Z}^{n_2}) \setminus \Psi_2 \right\}. \tag{4.1}$$

Similarly, we can equivalently reformulate (3.2)–(3.5) as the following ILP:

$$\max_{x,y} \left\{ c^\top x + d_1^\top y : (x, y) \in \mathcal{W}_0 \cap (\mathbb{Z}^{n_1} \times \mathbb{Z}^{n_2}) \right\}, \tag{4.2}$$

which is also a relaxation of the BILP (4.1).

The following lemma presents an implication of a watermelon seed, which is important for the development of the watermelon algorithm.

LEMMA 4.1. *If (x^0, y^0) is a watermelon seed, then there exists a $\Delta y^0 \in \mathbb{Z}^{n_2}$ such that $A_2 x^0 + B_2(y^0 + \Delta y^0) \leq b_2$ and $d_2^\top \Delta y^0 \geq 1$.*

Proof. If (x^0, y^0) is a watermelon seed, then y^0 is by definition a feasible but nonoptimal solution to $\mathcal{L}(x^0)$. Therefore, there must exist a better feasible solution $y^1 \in \mathbb{Z}^{n_2}$ satisfying $A_2 x^0 + B_2 y^1 \leq b_2$ and $d_2^\top y^1 - d_2^\top y^0 \geq 1$. The right-hand side of the latter inequality is due to the integrality of d_2 and y. As such, we can construct the desirable Δy^0 as $\Delta y^0 = y^1 - y^0$. ∎

For a given set of parameters $l, u \in \overline{\mathbb{R}}^{m_2}$, we use $\mathcal{B}(l,u)$ to denote the parametric BILP

$$\max_{x,y} \left\{ c^\top x + d_1^\top y : (x,y) \in \mathcal{W}(l,u) \cap (\mathbb{Z}^{n_1} \times \mathbb{Z}^{n_2}) \setminus \Psi_2 \right\},$$

and we use $\mathcal{R}(l,u)$ to denote its relaxation

$$\max_{x,y} \left\{ c^\top x + d_1^\top y : (x,y) \in \mathcal{W}(l,u) \cap (\mathbb{Z}^{n_1} \times \mathbb{Z}^{n_2}) \right\}.$$

As special cases, BILP (4.1) and its relaxation (4.2) are equivalent to $\mathcal{B}(\{-\infty\}^{m_2}, b_2)$ and $\mathcal{R}(\{-\infty\}^{m_2}, b_2)$, respectively.

The BILP differs from its ILP relaxation in the additional requirement that all seeds be removed from the watermelon. This interpretation inspired the following algorithmic strategy. We first solve the relaxation. If the solution turns out to be bilevel feasible, then it is also bilevel optimal; otherwise, it must be a watermelon seed, and then we scoop it out and solve the updated relaxation iteratively. This strategy is elaborated upon as follows.

Initialization: Define $\mathcal{G}$ as the feasible region of the relaxation and initialize it as $\mathcal{G} = \mathcal{W}_0 \cap (\mathbb{Z}^{n_1} \times \mathbb{Z}^{n_2})$. Go to the main loop.

Main loop: Solve the following relaxation:

$$\max_{x,y} \left\{ c^\top x + d_1^\top y : (x,y) \in \mathcal{G} \right\}. \tag{4.3}$$

In each iteration, $\mathcal{C}^0$ is a scoop that encloses the seed $(x^{\mathrm{R}}, y^{\mathrm{R}})$ and possibly other seeds but no bilevel feasible solutions. In the following three subsections, we discuss three challenges in implementing the aforementioned algorithmic strategy: the construction of the scoop, solving the relaxation, and the possibility of unbounded relaxation.

---

if (4.3) *is infeasible* **then**
  return: BILP is infeasible.
else if (4.3) *is unbounded* **then**
  additional investigation is needed.
else
  let $(x^{\mathrm{R}}, y^{\mathrm{R}})$ be an optimal solution to (4.3).
  if $(x^R, y^R) \in \Psi_1$ **then**
    return: $(x^{\mathrm{R}}, y^{\mathrm{R}})$ is an optimal solution to BILP (4.1).
  else
    construct a polyhedron $\mathcal{C}^0$ such that

$$(x^{\mathrm{R}}, y^{\mathrm{R}}) \in \mathcal{C}^0 \text{ and } \mathcal{C}^0 \cap \Psi_1 = \emptyset, \tag{4.4}$$

    update the feasible region of the relaxation as

$$\mathcal{G} \leftarrow \mathcal{G} \setminus \mathcal{C}^0, \tag{4.5}$$

    and repeat the main loop.
  end
end

4.1. The scoop construction problem. The scoop construction problem is essentially a separation problem, which separates a given watermelon seed $(x^{\mathrm{R}}, y^{\mathrm{R}})$ from the set of bilevel feasible solutions Ψ_1 by a polyhedron, as defined in (4.4). In single level discrete optimization, a hyperplane is commonly used for the separation of the relaxation solution and feasible solutions. For BILP, however, we may have $(x^{\mathrm{R}}, y^{\mathrm{R}}) \notin \Psi_1$ and $(x^{\mathrm{R}}, y^{\mathrm{R}}) \in \mathrm{conv}(\Psi_1)$; therefore, the separation may not be achievable by a hyperplane. For example, as can be seen in Figure 1, seed $(x = 0, y = 1)$ in Example 6 is enclosed by the convex hull of bilevel feasible solutions.

In the watermelon algorithm, the scoop is defined by the lower level constraints with modified right hand sides so that they make a shrinking translational movement inward:

$$\mathcal{C}(t) = \{(x, y) : A_2 x + B_2 y \le b_2 - t\}. \tag{4.6}$$

Here, $t \in \mathbb{R}^{m_2}_+$ is a design parameter, which specifies the distances between the facets of the scoop and their parallel counterparts of the watermelon. For a given seed $(x^{\mathrm{R}}, y^{\mathrm{R}})$, the following auxiliary ILP, referred to as the *type I scoop problem* and denoted as $\mathcal{S}^{\mathrm{I}}(x^{\mathrm{R}}, y^{\mathrm{R}})$, is designed to determine t:

$$\min_{\Delta y, t} \sum_{j=1}^{m_2} t_j \tag{4.7}$$

$$\text{s.t. } d_2^\top \Delta y \ge 1, \tag{4.8}$$

$$B_2 \Delta y \le t, \tag{4.9}$$

$$A_2 x^{\mathrm{R}} + B_2 y^{\mathrm{R}} \le b_2 - t, \tag{4.10}$$

$$t \in \mathbb{Z}^{m_2}_+;\ \Delta y \in \mathbb{Z}^{n_2}. \tag{4.11}$$

The scoop obtained from $\mathcal{S}^{\mathrm{I}}(x^{\mathrm{R}}, y^{\mathrm{R}})$ has at least two desirable features. First, it is able to separate seed $(x^{\mathrm{R}}, y^{\mathrm{R}})$ from bilevel feasible solutions. Second, it is designed to enclose not only the target seed $(x^{\mathrm{R}}, y^{\mathrm{R}})$ but also many surrounding ones. The following two lemmas explain the first feature.

LEMMA 4.2. *For any $(x^R, y^R) \in \mathcal{W}^0 \cap (\mathbb{Z}^{n_1} \times \mathbb{Z}^{n_2})$, (x^R, y^R) is a watermelon seed if and only if $\mathcal{S}^I(x^R, y^R)$ has a feasible solution.*

Proof. The "only if" direction: If $(x^{\mathrm{R}}, y^{\mathrm{R}})$ is a watermelon seed, then by Lemma 4.1 there exists a Δy^0 such that $A_2 x^{\mathrm{R}} + B_2(y^{\mathrm{R}} + \Delta y^0) \le b_2$ and $d_2^\top \Delta y^0 \ge 1$. It is easy to check that $(\Delta y = \Delta y^0, t = \max\{B_2 \Delta y^0, 0\})$ is a feasible solution to $\mathcal{S}^{\mathrm{I}}(x^{\mathrm{R}}, y^{\mathrm{R}})$.

The "if" direction: Let $(\Delta y^1, t^1)$ be a feasible solution to $\mathcal{S}^{\mathrm{I}}(x^{\mathrm{R}}, y^{\mathrm{R}})$. Then it is easy to check that $y^{\mathrm{R}} + \Delta y^1$ dominates y^{R} in $\mathcal{L}(x^{\mathrm{R}})$; thus, $(x^{\mathrm{R}}, y^{\mathrm{R}})$ is by definition a watermelon seed. ∎

LEMMA 4.3. *If there exists $(x^R, y^R) \in \mathcal{W}^0 \cap (\mathbb{Z}^{n_1} \times \mathbb{Z}^{n_2})$ such that $(\Delta y, t)$ is a feasible solution to $\mathcal{S}^I(x^R, y^R)$, then no bilevel feasible solution is enclosed inside the scoop $\mathcal{C}(t)$.*

Proof. We prove that any solution $(x^0 \in \mathbb{Z}^{n_1}, y^0 \in \mathbb{Z}^{n_2})$ enclosed inside the scoop $\mathcal{C}(t)$ is bilevel infeasible. By the definition of $\mathcal{S}^{\mathrm{I}}(x^{\mathrm{R}}, y^{\mathrm{R}})$ and the fact that (x^0, y^0) is enclosed inside the scoop, solution $(x^0, y^0 + \Delta y)$ satisfies the following conditions:

$$\begin{aligned} &A_2 x^0 + B_2(y^0 + \Delta y) \le b_2, \\ &y^0 + \Delta y \in \mathbb{Z}^{n_2}, \\ &d_2^\top (y^0 + \Delta y) \ge d_2^\top y^0 + 1, \end{aligned}$$

which means that $(x^0, y^0 + \Delta y)$ dominates (x^0, y^0) in $\mathcal{L}(x^0)$, so $y^0 \notin \Psi_0(x^0)$, and thus (x^0, y^0) is bilevel infeasible. ∎

The second desirable feature of the type I scoop problem is because the total distance between the facets of the scoop and those of the watermelon is being minimized. This objective is intuitively aligned with the maximization of the volume of the scoop and thus the number of seeds enclosed.

We illustrate the type I scoop problem by computing a scoop to carve out the seed $(x^{\mathrm{R}} = 2, y^{\mathrm{R}} = 4)$ for Example 4. We define t_1, t_2, t_3, and t_4 for the four facets of the watermelon $-5x + 4y \le 6$, $x + 2y \le 10$, $2x - y \le 15$, and $-2x - 10y \le -15$, respectively. Then the type I scoop problem $\mathcal{S}^{\mathrm{I}}(x^{\mathrm{R}}, y^{\mathrm{R}})$ becomes

$$\begin{aligned}
\min_{\Delta y, t}\ & t_1 + t_2 + t_3 + t_4 \\
\text{s.t. } & -\Delta y \ge 1, \\
& 4\Delta y \le t_1, \\
& 2\Delta y \le t_2, \\
& -\Delta y \le t_3, \\
& -10\Delta y \le t_4, \\
& 6 \le 6 - t_1, \\
& 10 \le 10 - t_2, \\
& 0 \le 15 - t_3, \\
& -44 \le -15 - t_4, \\
& t_1, t_2, t_3, t_4 \in \mathbb{Z}_+;\ \Delta y \in \mathbb{Z}.
\end{aligned}$$

An optimal solution to $\mathcal{S}^{\mathrm{I}}(x^{\mathrm{R}}, y^{\mathrm{R}})$ is $(t_1^* = 0, t_2^* = 0, t_3^* = 1, t_4^* = 10)$, which defines the scoop

$$\mathcal{C}(t^*) = \{(x, y) : -5x + 4y \le 6; x + 2y \le 10; 2x - y \le 14; -2x - 10y \le -25\}. \tag{4.12}$$

Figure 2 shows how this scoop (in lighter shade) is able to enclose all the seeds. It can be expected that, in the next iteration, the relaxation with the scoop of flesh carved out of the watermelon will yield the bilevel optimal solution.

Motivated by the above example, the question arises whether for all BILP instances an ideal convex scoop can be found that is able to enclose all the seeds. Unfortunately, the following example shows that such an ideal scoop may not exist: some BILP instances may require more than one scoop to enclose all of the seeds no matter what convex scoop is used.

Example 8. This example cannot be solved by the watermelon algorithm in one iteration.

$$\begin{aligned}
\max_{x, y_1, y_2}\ & x - y_1 - y_2 \\
\text{s.t. } & 1 \le x \le 3,\ x \in \mathbb{Z}, \\
& (y_1, y_2) \in \arg\max_{\tilde{y}_1, \tilde{y}_2} \{\tilde{y}_1 + \tilde{y}_2 : 2\tilde{y}_1 \le x + 1; 2\tilde{y}_2 \le x + 1; \tilde{y}_1, \tilde{y}_2 \in \mathbb{Z}\}.
\end{aligned}$$

We point out three observations about Example 8. First, $(x = 1, y_1 = 0, y_2 = 1)$ and $(x = 3, y_1 = 2, y_2 = 1)$ are two watermelon seeds, because they are, respectively, dominated by $(x = 1, y_1 = 1, y_2 = 1)$ and $(x = 3, y_1 = 2, y_2 = 2)$ in the lower

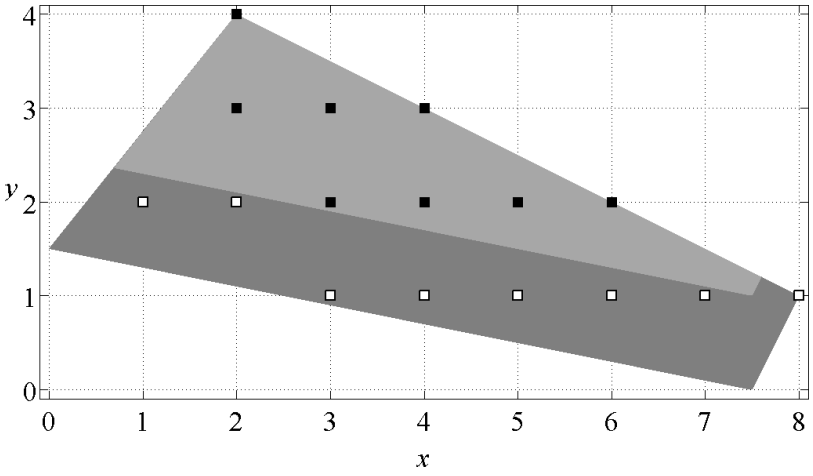

FIG. 2. *Illustration of scoop (in lighter shade) designed by the type* I *scoop problem for Example* 4.

level. Second, $(x = 2, y_1 = 1, y_2 = 1)$ is a bilevel feasible (indeed, optimal) solution. Third, the bilevel feasible solution lies on the line segment between the two seeds. As a result, no single convex scoop can enclose both seeds without also containing the bilevel feasible solution and violating the requirement (4.4) for a scoop. Therefore, this example requires solving at least two iterations of the type I scoop problem.

4.2. Solving the relaxation. The way the feasible region of the relaxation is updated in (4.5) could introduce additional nonconvexity (besides the integrality requirement), which makes the relaxation hard to solve. The following lemma explains that when a scoop of flesh defined in (4.6) is carved out of a slice of watermelon, all bilevel feasible solutions can be partitioned into m_2 smaller slices of watermelon.

LEMMA 4.4. *Suppose* (x^R, y^R) *is a seed enclosed by the slice of watermelon* $\mathcal{W}(l, u)$ *and* $(\Delta y, t)$ *is a feasible solution to* $\mathcal{S}^I(x^R, y^R)$. *Then we have*

$$\begin{aligned}\mathcal{W}(l,u) \cap (\mathbb{Z}^{n_1} \times \mathbb{Z}^{n_2}) \setminus \mathcal{C}(t) &= \mathcal{W}(l,u) \cap (\mathbb{Z}^{n_1} \times \mathbb{Z}^{n_2}) \setminus \mathcal{W}(\{-\infty\}^{m_2}, b_2 - t) \\ &= \bigcup_{k=1}^{m_2} \mathcal{W}(\tilde{l}^k, \tilde{u}^k) \cap (\mathbb{Z}^{n_1} \times \mathbb{Z}^{n_2}),\end{aligned}$$

where for all $k \in \{1, \ldots, m_2\}$ *and* $j \in \{1, \ldots, m_2\}$, $\tilde{l}^k$ *and* $\tilde{u}^k$ *are defined as*

$$\tilde{l}_j^k = \begin{cases} (b_2 - t)_j + 1 & \text{if } j = k, \\ l_j & \text{otherwise}; \end{cases} \tag{4.13}$$

$$\tilde{u}_j^k = \begin{cases} \min\{u_j, (b_2 - t)_j\} & \text{if } j \le k - 1, \\ u_j & \text{otherwise.} \end{cases} \tag{4.14}$$

Proof. After the scoop of flesh is carved out of the watermelon, the remaining flesh must violate at least one of the following m_2 scoop constraints: $(A_2 x + B_2 y)_j \le (b_2 - t)_j \ \forall j \in \{1, \ldots, m_2\}$. The parameters $\tilde{l}^k$ and $\tilde{u}^k$ are determined in such a way that the slice $\mathcal{W}(\tilde{l}^k, \tilde{u}^k)$ is a subset of $\mathcal{W}(l, u)$ that satisfies the first $(k-1)$ scoop constraints and violates the kth one. As such, all bilevel feasible solutions within $\mathcal{W}(l, u)$ are partitioned into the m_2 smaller slices. When the jth scoop constraint is violated, the lower bound $\tilde{l}_j^k$ is strengthened to $(b_2 - t)_j + 1$ due to the integrality

requirements of A_2, B_2, b_2, x, and y. It is a stronger lower bound than l_j because $b_2 - t + 1 > b_2 - t \geq A_2 x^{\mathrm{R}} + B_2 y^{\mathrm{R}} \geq l$. The second inequality is because of constraint (4.10), and the last one is because $(x^{\mathrm{R}}, y^{\mathrm{R}}) \in \mathcal{W}(l, u)$. ∎

Lemma 4.4 suggests that removing a scoop of flesh from a watermelon slice is equivalent to creating m_2 sub-slices by cutting the original slice $2m_2$ times, with two cuts for each subslice. For all $j \in \{1, \ldots, m_2\}$, the jth subslice is created by the cut $(A_2 x + B_2 y)_j \geq (b_2 - t)_j + 1$; then, another cut $(A_2 x + B_2 y)_j > (b_2 - t)_j$ chops off a piece of scrap from the remainder of the slice. This process is carried out sequentially until all m_2 subslices are created.

We illustrate this multiway disjunction branching scheme on Example 4. Suppose the following scoop is used to carve out seed $(x = 3, y = 2)$:

$$\mathcal{C} = \{(x, y) : -5x + 4y \leq 0; x + 2y \leq 8; 2x - y \leq 7; -2x - 10y \leq -25\}. \tag{4.15}$$

The following four subslices are created using Lemma 4.4:

$$\begin{aligned}
\mathcal{W}_1 &= \{(x, y) : 1 \leq -5x + 4y \leq 6; x + 2y \leq 10; 2x - y \leq 15; -2x - 10y \leq -15\},\\
\mathcal{W}_2 &= \{(x, y) : -5x + 4y \leq 0; 9 \leq x + 2y \leq 10; 2x - y \leq 15; -2x - 10y \leq -15\},\\
\mathcal{W}_3 &= \{(x, y) : -5x + 4y \leq 0; x + 2y \leq 8; 8 \leq 2x - y \leq 15; -2x - 10y \leq -15\}, \text{ and}\\
\mathcal{W}_4 &= \{(x, y) : -5x + 4y \leq 0; x + 2y \leq 8; 2x - y \leq 7; -24 \leq -2x - 10y \leq -15\}.
\end{aligned}$$

Figures 3 and 4 illustrate how sub-slices $\mathcal{W}_1$ and $\mathcal{W}_2$ are created, respectively. Figure 5 shows all four subslices.

Scoop (4.15) is used here only to illustrate the multiway disjunction approach. In the watermelon algorithm, for the given seed $(x = 3, y = 2)$, the type I scoop problem would yield the more efficient scoop (4.12), which is able to carve out all the seeds and create only two new nonempty watermelon slices (one of which is a line segment), as shown in Figure 6.

4.3. Unbounded relaxation. When the relaxation $\mathcal{R}(l, u)$ is unbounded, it possesses an infinite sequence of solutions in $\mathcal{W}(l, u) \cap (\mathbb{Z}^{n_1} \times \mathbb{Z}^{n_2})$ whose objective values measured in (1.1) asymptotically approach infinity. The unboundedness of

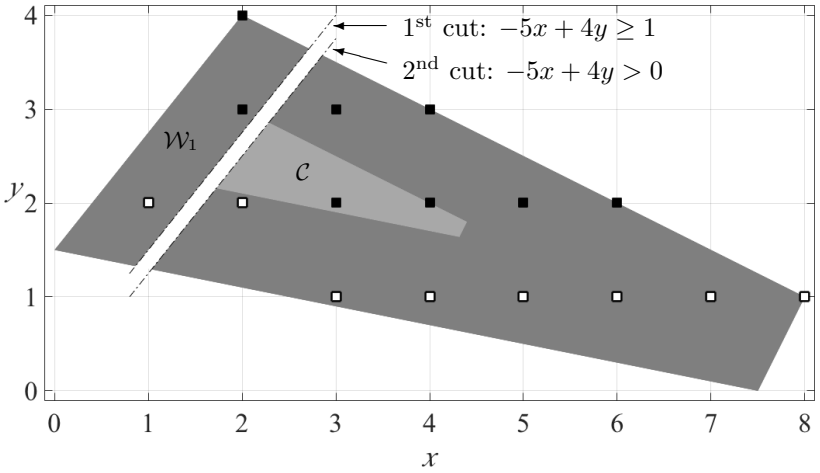

FIG. 3. *Creation of* $\mathcal{W}_1$ *using scoop* (4.15) *for Example* 4.

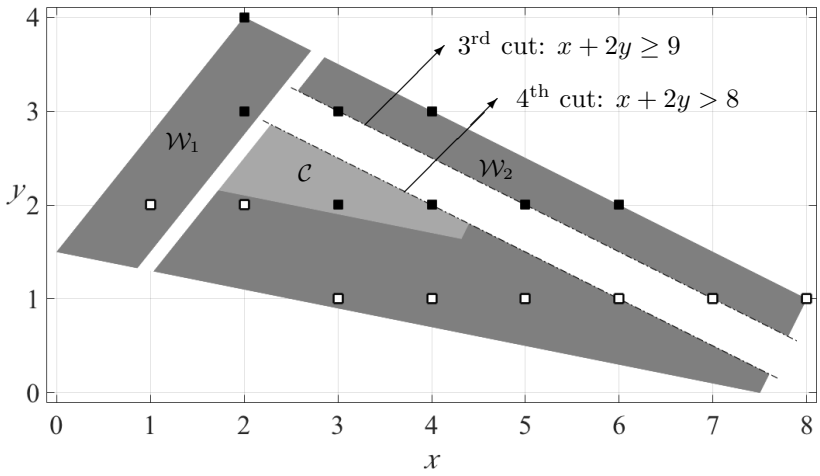

FIG. 4. *Creation of $\mathcal{W}_2$ using scoop* (4.15) *for Example* 4.

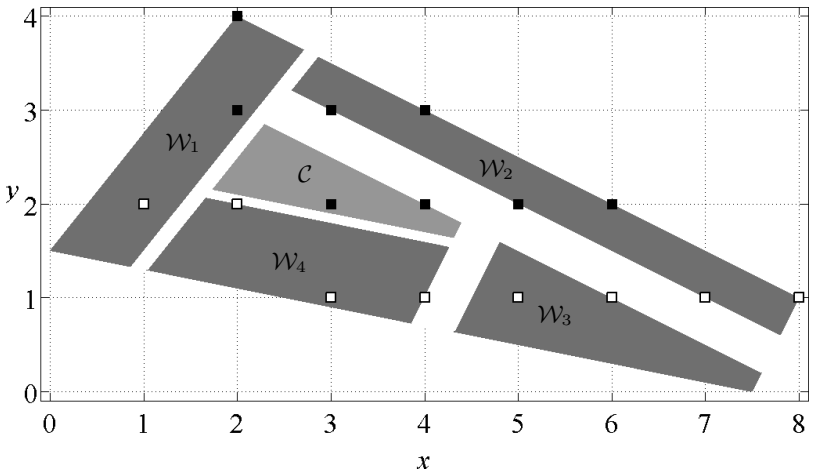

FIG. 5. *Illustration of a multiway disjunction cut using scoop* (4.15) *for Example* 4.

$\mathcal{R}(l,u)$ does not ascertain that of $\mathcal{B}(l,u)$. In fact, with $\mathcal{R}(l,u)$ being unbounded, $\mathcal{B}(l,u)$ may be infeasible, unbounded, or optimal, depending on how the infinite sequence of solutions are divided between Ψ_1 and Ψ_2. If the unboundedness of $\mathcal{R}(l,u)$ is caused by bilevel feasible solutions in Ψ_1, then $\mathcal{B}(l,u)$ is unbounded. If, on the other hand, watermelon seeds in Ψ_2 are solely responsible for the unboundedness of $\mathcal{R}(l,u)$, then $\mathcal{B}(l,u)$ is either optimal or infeasible, depending on the existence of a bilevel feasible solution.

Consider Examples 5, 6, and 7 defined in section 3. All three instances in these examples share the same relaxation, $\max_{x,y}\{x + y : -2 \leq x \leq 2; 2y \geq 3x; 2y \geq -3x; x, y \in \mathbb{Z}\}$, which is unbounded. The outcomes of these BILP instances, however, span all three possibilities.

The watermelon algorithm reveals the true property of $\mathcal{B}(l,u)$ behind its unbounded relaxation $\mathcal{R}(l,u)$ by solving the following auxiliary ILP, which is referred to as the *type II scoop problem* and denoted as $\mathcal{S}^{\mathrm{II}}(l,u)$:

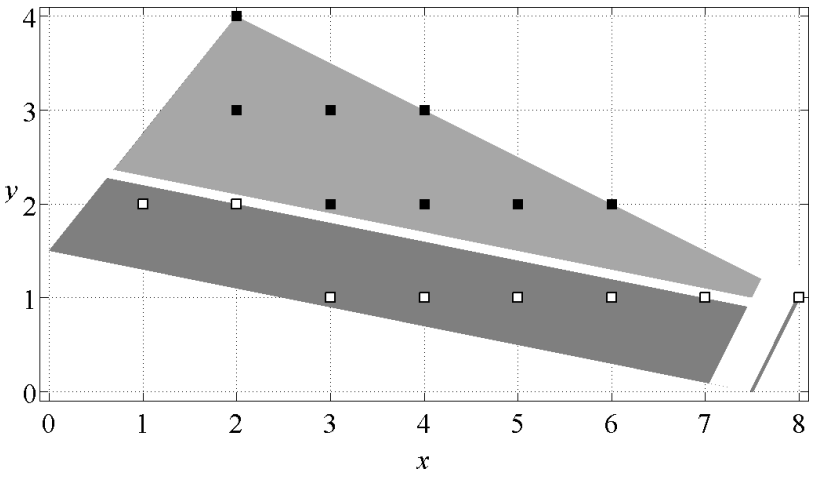

FIG. 6. *Illustration of multiway disjunction using scoop* (4.12) *for Example* 4.

$$\begin{aligned}
\min_{x,y,\Delta y,t} \ & \sum_{j=1}^{m_2} t_j \\
\text{s.t. } & d_2^\top \Delta y \geq 1, \\
& B_2 \Delta y \leq t, \\
& A_1 x + B_1 y \leq b_1, \\
& A_2 x + B_2 y \leq b_2 - t, \\
& l \leq A_2 x + B_2 y \leq u, \\
& t \in \mathbb{Z}_+^{m_2};\ x \in \mathbb{Z}^{n_1};\ y, \Delta y \in \mathbb{Z}^{n_2}.
\end{aligned}$$

Unlike the type I scoop problem, which designs a scoop to enclose a given seed, the type II scoop problem must search for a seed inside a given slice of watermelon first and then design a scoop to enclose it. The following three lemmas explain how the result from $\mathcal{S}^{\mathrm{II}}(l,u)$ reveals the property of $\mathcal{B}(l,u)$. Lemma 4.5 shows that the existence of a seed in $\mathcal{W}(l,u)$ is a necessary and sufficient condition for $\mathcal{S}^{\mathrm{II}}(l,u)$ to have an optimal solution.

LEMMA 4.5. *For any* $(x^0,y^0) \in \mathcal{W}(l,u) \cap (\mathbb{Z}^{n_1} \times \mathbb{Z}^{n_2})$, (x^0,y^0) *is a watermelon seed if and only if there exists* $(\Delta y,t)$ *such that* $(x^0,y^0,\Delta y,t)$ *is a feasible solution to* $\mathcal{S}^{II}(l,u)$.

Proof. The "only if" direction: if (x^0,y^0) is a watermelon seed, by Lemma 4.2, $\mathcal{S}^{\mathrm{I}}(x^0,y^0)$ has a feasible solution, which we denote as $(\Delta y,t)$. It is easy to check that $(x^0,y^0,\Delta y,t)$ is also a feasible solution to $\mathcal{S}^{\mathrm{II}}(l,u)$.

The "if" direction: if $(x^0,y^0,\Delta y,t)$ is a feasible solution to $\mathcal{S}^{\mathrm{II}}(l,u)$, then it is easy to check that $(\Delta y,t)$ is a feasible solution to $\mathcal{S}^{\mathrm{I}}(x^0,y^0)$. By Lemma 4.2, (x^0,y^0) is a watermelon seed. ∎

As a corollary of Lemma 4.5, the following lemma shows that if $\mathcal{S}^{\mathrm{II}}(l,u)$ is infeasible, then the watermelon slice $\mathcal{W}(l,u)$ does not contain any seed, so bilevel feasible solutions are solely responsible for the unboundedness of $\mathcal{R}(l,u)$, and thus $\mathcal{B}(l,u)$ is also unbounded.

LEMMA 4.6. *If there exist* $l, u \in \overline{\mathbb{R}}^{m_2}$ *such that* $\mathcal{R}(l,u)$ *is unbounded and* $\mathcal{S}^{II}(l,u)$ *infeasible, then* $\mathcal{B}(l,u)$ *is unbounded.*

Proof. If $\mathcal{R}(l,u)$ is unbounded, then for any real number K there exists $(x^{\mathrm{R}}, y^{\mathrm{R}}) \in \mathcal{W}(l,u) \cap (\mathbb{Z}^{n_1} \times \mathbb{Z}^{n_2})$ such that $c^\top x^{\mathrm{R}} + d_1^\top y^{\mathrm{R}} \geq K$. If $\mathcal{S}^{\mathrm{II}}(l,u)$ is infeasible, then $\mathcal{S}^{\mathrm{I}}(x^{\mathrm{R}}, y^{\mathrm{R}})$ is also infeasible. By Lemma 4.2, $(x^{\mathrm{R}}, y^{\mathrm{R}})$ is not a seed, and thus is bilevel feasible. Therefore, for any real number K, there exists a bilevel feasible solution $(x^{\mathrm{R}}, y^{\mathrm{R}})$ such that $c^\top x^{\mathrm{R}} + d_1^\top y^{\mathrm{R}} \geq K$, which establishes the unboundedness of $\mathcal{B}(l,u)$. ∎

The next lemma confirms that the scoop generated from $\mathcal{S}^{\mathrm{II}}(l,u)$ satisfies the requirement (4.4).

LEMMA 4.7. *If there exist* $l, u \in \overline{\mathbb{R}}^{m_2}$ *such that* $(x, y, \Delta y, t)$ *is a feasible solution to* $\mathcal{S}^{II}(l,u)$*, then no bilevel feasible solution exists in the scoop* $\mathcal{C}(t)$*.*

Proof. Similar to the proof of Lemma 4.3. ∎

Lemmas 4.5–4.7 motivated the following strategy to deal with unbounded relaxation. Once $\mathcal{R}(l,u)$ is found unbounded, we iteratively use $\mathcal{S}^{\mathrm{II}}(l,u)$ to search for seeds and generate scoops to remove them from the watermelon slice $\mathcal{W}(l,u)$ until one of two possible terminating conditions is met: either $\mathcal{S}^{\mathrm{II}}(l,u)$ is infeasible, indicating that $\mathcal{B}(l,u)$ is unbounded, or $\mathcal{R}(l,u)$ is no longer unbounded. The finite termination of this strategy is proved in section 6. The next lemma explains the implication of a scoop $\mathcal{C}(t)$ with $t = 0$.

LEMMA 4.8. *If there exist* $l, u \in \overline{\mathbb{R}}^{m_2}$ *such that the optimal objective value of* $\mathcal{S}^{II}(l,u)$ *is zero, then the BILP is infeasible.*

Proof. The optimal objective value of $\mathcal{S}^{\mathrm{II}}(l,u)$ being zero means that $t = 0$ is a feasible solution, then the scoop $\mathcal{C}(0)$ encloses the entire watermelon, which, by Lemma 4.7, does not contain any bilevel feasible solution. ∎

We note that if Lemma 4.8 becomes applicable, it will apply at the root node in the algorithm, since the parameters l and u will only get more restrictive in subsequent iterations, which will increase the optimal objective value. We also point out that the converse statement of Lemma 4.8 is not true. Consider the following Example 9, which is illustrated in Figure 7.

Example 9. The relaxation at the root is unbounded, but the BILP is infeasible, because the lower level imposes $y = x$, which is incompatible with the upper level constraints (4.18) and (4.19). The optimal solution to the type II scoop problem $\mathcal{S}^{\mathrm{II}}(\{-\infty\}^{m_2}, b_2)$ yields $(t_1^* = 1, t_2^* = 0)$, with the objective value being 1. The scoop defined by this solution encloses the entire watermelon, but it does leave a gap from the lower level constraint $-y \leq -x$. Since the parameters $(\{-\infty\}^{m_2}, b_2)$ in the type II scoop problem are as unrestrictive as $(\{-\infty\}^{m_2}, \{+\infty\}^{m_2})$, there do not exist $l, u \in \overline{\mathbb{R}}^{m_2}$ that will result in a smaller optimal objective value:

$$\max_{x,y} \ \zeta = x + y \tag{4.16}$$

$$\text{s.t. } 2x - y \leq 0, \tag{4.17}$$

$$-x \leq -1, \ x \in \mathbb{Z}, \tag{4.18}$$

$$-y \leq 0, \ y \in \mathbb{Z}, \tag{4.19}$$

$$y \in \arg\max_{\tilde{y}} \{-\tilde{y} : -\tilde{y} \leq -x; \tilde{y} \leq 3x; \tilde{y} \in \mathbb{Z}\}. \tag{4.20}$$

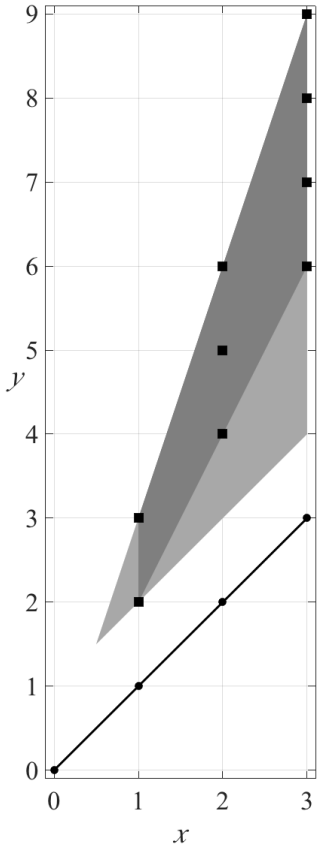

FIG. 7. *Illustration of Example* 9. *Both the watermelon and the scoop are unbounded from northeastern. The dots on the ray from* $(0,0)$ *through* $(3,3)$ *to infinity are the optimal solutions to the lower level, but they all violate upper level constraints.*

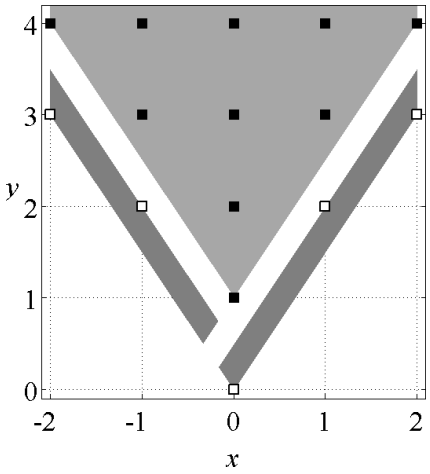

FIG. 8. *Illustration of scoop designed by type* II *scoop problem and multiway disjunction for Example* 6. *Both the watermelon and the scoop are unbounded from above.*

We illustrate the application of the type II scoop problem on Example 6, whose relaxation is unbounded. We define t_1 and t_2 for the two facets of the watermelon $3x - 2y \leq 0$ and $-3x - 2y \leq 0$, respectively. Then the type II scoop problem becomes

$$\begin{aligned} \min_{x,y,\Delta y,t} \ & t_1 + t_2 \\ \text{s.t.} \ & -\Delta y \geq 1, \\ & -2\Delta y \leq t_1, \\ & -2\Delta y \leq t_2, \\ & -2 \leq x \leq 2, \\ & 3x - 2y \leq -t_1, \\ & -3x - 2y \leq -t_2, \\ & t_1, t_2 \in \mathbb{Z}_+; \ x, y, \Delta y \in \mathbb{Z}. \end{aligned}$$

An optimal solution to this type II scoop problem is $(x^* = 0, y^* = 1, \Delta y^* = -1, t_1^* = 2, t_2^* = 2)$, which defines the scoop $\mathcal{C}(t^*) = \{(x, y) : 3x - 2y \leq -2; -3x - 2y \leq -2\}$. Figure 8 shows how this scoop (unbounded from above) is able to enclose all of the infinitely many seeds. The figure also shows the creation of two (bounded) subslices using the multiway disjunction cut approach described in section 4.2.

5. The watermelon algorithm for BILP. We are now ready to present the watermelon algorithm for BILP, which takes the set of BILP parameters $(A_1, A_2, B_1, B_2, b_1, b_2, c, d_1, d_2)$ as input and outputs the optimal solution (x^*, y^*, ζ^*) to the BILP (1.1)–(1.4). The notations of $(x^* = \emptyset, y^* = \emptyset, \zeta^* = -\infty)$ and $(x^* = \emptyset, y^* = \emptyset, \zeta^* = +\infty)$ are used as the outputs for infeasible and unbounded instances, respectively. Steps of the watermelon algorithm are summarized below and diagrammed in Figure 9. In the algorithm, a node is defined as a parametric BILP $\mathcal{B}(l, u)$, characterized by $l, u \in \overline{\mathbb{R}}^{m_2}$; a parameter z^j is used to record the objective value of the relaxation for node j's parent node for bounding purposes; and parameter N keeps track of the number of active nodes in the branch-and-bound tree.

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

Step 0 (Initialization): Create node 1, $\mathcal{B}(l^1, u^1)$, with $l^1 = \{-\infty\}^{m_2}, u^1 = b_2$. Initialize $x^* = \emptyset, y^* = \emptyset, \zeta^* = -\infty, N = 1$, and $z^1 = \infty$. Go to Step 1.

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

---

if $N = 0$ **then**
  if $x^* \neq \emptyset$ **then**
    1(a) return: (x^*, y^*, ζ^*) is an optimal solution to the BILP (1.1)–(1.4).
  else
    1(b) return: BILP (1.1)–(1.4) is infeasible.
  end
else
  1(c) select a node k from $\{1, \ldots, N\}$, set $\hat{l} = l^k$ and $\hat{u} = u^k$, discard 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 $\mathcal{R}(\hat{l}, \hat{u})$.

---

if $\mathcal{R}(\hat{l}, \hat{u})$ *is infeasible* **then**
  2(a) go to Step 1.
else if $\mathcal{R}(\hat{l}, \hat{u})$ *is unbounded* **then**
  2(b) go to Step 6.
else
  let $(x^{\mathrm{R}}, y^{\mathrm{R}})$ denote an optimal solution to $\mathcal{R}(\hat{l}, \hat{u})$.
  if $c^\top x^R + d_1^\top y^R \le \zeta^*$ **then**
    2(c) go to Step 1.
  else
    2(d) go to Step 3.
  end
end

Step 3 (Lower level): Solve $\mathcal{L}(x^{\mathrm{R}})$.

---

if $\mathcal{L}(x^R)$ *is unbounded* **then**
  3(a) return: BILP (1.1)–(1.4) is infeasible.
else
  let y^{L} denote an optimal solution to $\mathcal{L}(x^{\mathrm{R}})$.
  if $d_2^\top y^R = d_2^\top y^L$ **then**
    3(b) update $(x^* = x^{\mathrm{R}}, y^* = y^{\mathrm{R}}, \zeta^* = c^\top x^{\mathrm{R}} + d_1^\top y^{\mathrm{R}})$ and go to Step 1.
  else
    if $A_1 x^R + B_1 y^L \le b_1$ *and* $c^\top x^R + d_1^\top y^L > \zeta^*$ **then**
      update $(x^* = x^{\mathrm{R}}, y^* = y^{\mathrm{L}}, \zeta^* = c^\top x^{\mathrm{R}} + d_1^\top y^{\mathrm{L}})$.
      if $d_1^\top y^R = d_1^\top y^L$ **then**
        3(c) go to Step 1.
      end
    end
    3(d) go to Step 4.
  end
end

Step 4 (Type I scoop): Solve $\mathcal{S}^{\mathrm{I}}(x^{\mathrm{R}}, y^{\mathrm{R}})$ and let $(\Delta y^1, t)$ denote an optimal solution. Go to Step 5.

Step 5 (Multiway disjunction): Create m_2 new nodes, $\mathcal{B}(l^{N+k}, u^{N+k})$ $\forall k \in \{1, \ldots, m_2\}$. For all $k \in \{1, \ldots, m_2\}$ and $j \in \{1, \ldots, m_2\}$, the parameters l^{N+k}, u^{N+k}, and z^{N+k} are defined as

$$l_j^{N+k} = \begin{cases} (b_2 - t)_j + 1 & \text{if } j = k, \\ \hat{l}_j & \text{otherwise;} \end{cases} \tag{5.1}$$

$$u_j^{N+k} = \begin{cases} \min\{\hat{u}_j, (b_2 - t)_j\} & \text{if } j \le k-1, \\ \hat{u}_j & \text{otherwise;} \end{cases} \tag{5.2}$$

$$z^{N+k} = \begin{cases} c^\top x^{\mathrm{R}} + d_1^\top y^{\mathrm{R}} & \text{if Step 4 was the previous step,} \\ +\infty & \text{otherwise.} \end{cases} \tag{5.3}$$

Increase N by m_2. Go to Step 1.

Step 6 (Type II scoop): Solve $\mathcal{S}^{\mathrm{II}}(\hat{l}, \hat{u})$.

---

if $\mathcal{S}^{II}(\hat{l}, \hat{u})$ *is infeasible* **then**
  6(a) return: BILP (1.1)–(1.4) is unbounded.
else
  let $(x^2, y^2, \Delta y^2, t)$ be an optimal solution to $\mathcal{S}^{\mathrm{II}}(\hat{l}, \hat{u})$.
  if $t = 0$ **then**
    6(b) return: BILP (1.1)–(1.4) is infeasible.
  else
    6(c) go to Step 5.
  end
end

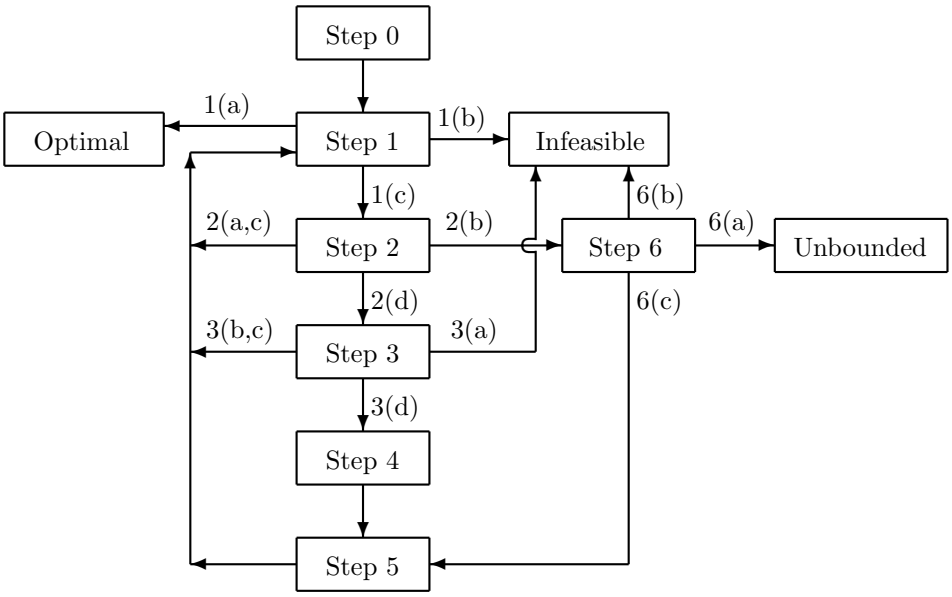

FIG. 9. *Diagram of the watermelon algorithm.*

6. Proofs of finite termination and correctness. In this section, we prove the finite termination and correctness of the watermelon algorithm. Consider the following parametric ILP, denoted as $\mathcal{F}(u)$ with $u \in \mathbb{R}^{m_2}$ being a given parameter:

$$f(u) = \min_{\Delta y, t} \left\{ \sum_{j=1}^{m_2} t_j : d_2^\top \Delta y \geq 1; B_2 \Delta y \leq t; 0 \leq t \leq u; t \in \mathbb{Z}^{m_2}; \Delta y \in \mathbb{Z}^{n_2} \right\},$$

where $f(u)$ denotes the optimal objective value of $\mathcal{F}(u)$. Define $\mathcal{U}$ as the set of integer u such that $\mathcal{F}(u)$ possesses a feasible solution:

$$\mathcal{U} = \left\{ u \in \mathbb{Z}_+^{m_2} : d_2^\top \Delta y \geq 1; B_2 \Delta y \leq t; 0 \leq t \leq u; \text{ for some } t \in \mathbb{Z}^{m_2} \text{ and } \Delta y \in \mathbb{Z}^{n_2} \right\}.$$

The following lemma presents two important properties of $f(u)$.

LEMMA 6.1. *$f(u)$ is a monotonically decreasing function of $u \in \mathcal{U}$, and it is uniformly bounded on $\mathcal{U}$.*

Proof. Proof of monotonicity: For any $u^1 \leq u^2 \in \mathcal{U}$, we have $f(u^1) \geq f(u^2)$ because an optimal solution to $\mathcal{F}(u^1)$ is feasible to $\mathcal{F}(u^2)$, but the reverse statement is not true.

Proof of boundedness: By definition, for all $u \in \mathcal{U}$, $f(u) \geq 0$. Since $\mathcal{U} \subseteq \mathbb{Z}_+^{m_2}$, according to Dickson's lemma [12], it has a finite minimal set, say S, with respect to point-wise partial order. Let M denote the uniform upper bound of $\{f(u) : u \in S\}$, which is a finite number. For any $u \in \mathcal{U} \setminus S$, since it is not minimal, there exists $u' \in S$ such that $u' \leq u$, which implies that $f(u) \leq f(u') \leq M$. Therefore, M is also the uniform upper bound of $\{f(u) : u \in \mathcal{U}\}$. ∎

As a corollary of Lemma 6.1, the following lemma establishes the uniform boundedness of $\mathcal{S}^{\mathrm{I}}(x, y)$ and $\mathcal{S}^{\mathrm{II}}(l, u)$.

LEMMA 6.2. *If feasible, $\mathcal{S}^{I}(x, y)$ and $\mathcal{S}^{II}(l, u)$ are uniformly bounded for all (x, y) and (l, u).*

Proof. By definition, $\mathcal{S}^{\mathrm{I}}(x, y)$ is equivalent to $\mathcal{F}(b_2 - A_2x - B_2y)$ and $\mathcal{S}^{\mathrm{II}}(l, u)$ is equivalent to $\min_{x,y}\{f(b_2 - A_2x - B_2y) : (x, y) \in \mathcal{V}(l, u)\}$, where $\mathcal{V}(l, u) = \{(x, y) : A_1x + B_1y \leq b_1; l \leq A_2x + B_2y \leq u; x \in \mathbb{Z}^{n_1}; y \in \mathbb{Z}^{n_2}\}$. By Lemma 6.1, these two scoop problems are uniformly bounded if feasible. ∎

Now we establish the finite termination and correctness of the watermelon algorithm in the following two theorems.

THEOREM 6.3. *For any input of BILP instance defined in* (1.1)–(1.4)*, the watermelon algorithm terminates finitely.*

Proof. In the watermelon algorithm, $\mathcal{S}^{\mathrm{I}}(x, y)$ is always fed with a seed (x, y), which ensures the feasibility of $\mathcal{S}^{\mathrm{I}}(x, y)$, according to Lemma 4.5. Since the algorithm terminates in Step 6 whenever $\mathcal{S}^{\mathrm{II}}(l, u)$ is infeasible for some (l, u), we only consider the situation when $\mathcal{S}^{\mathrm{II}}(l, u)$ is feasible. By Lemma 6.2, both $\mathcal{S}^{\mathrm{I}}(x, y)$ and $\mathcal{S}^{\mathrm{II}}(l, u)$ are bounded. Let K denote an upper bound for both $\mathcal{S}^{\mathrm{I}}(x, y)$ and $\mathcal{S}^{\mathrm{II}}(l, u)$. Then for all $j \in \{1, \ldots, m_2\}$, the parameters l_j and u_j that could possibly be used to characterize a node only lie within $\{-\infty, (b_2)_j - K, (b_2)_j - K + 1, \ldots, (b_2)_j - 1, (b_2)_j, (b_2)_j + 1\}$, which is a finite set. Moreover, the multiway disjunction in Step 5 ensures that no two nodes have the same bounds. Therefore, the branch-and-bound tree generated in the watermelon algorithm only contains a finite number of possible nodes, and thus it takes a finite number of iterations for the watermelon algorithm to explore the entire tree. ∎

THEOREM 6.4. *When the watermelon algorithm terminates, the output is correct.*

Proof. We show the correctness of all decision points in the algorithm steps.

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

Step 2(b) directs an unbounded relaxation to Step 6 for further investigation.

Step 3(a) concludes that the BILP is infeasible, due to Lemma 3.1.

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

Step 4 calculates a scoop of watermelon flesh that will be carved out of the feasible region of the current node in Step 5. The correctness of this scoop is guaranteed by Lemmas 4.2 and 4.3.

Step 5 carves a scoop (obtained either from Step 4 or Step 6) of watermelon flesh out of the current node and creates m_2 new branches using results from Lemma 4.4.

Step 6(a) is due to Lemma 4.6.

Step 6(b) is due to Lemma 4.8.

Step 6(c) calculates a scoop of watermelon flesh that will be carved out of the feasible region of the current node in Step 5. The correctness of this scoop is guaranteed by Lemmas 4.5 and 4.7. ∎

7. An alternative type I scoop problem. The purpose of the scoop was to enclose as many watermelon seeds as possible. However, determining the exact number of integer solutions inside a polyhedron is NP-hard (with ILP feasibility as a special case), so we did not directly maximize the number of enclosed seeds. Instead, we minimize $\sum_j t_j$, the total distance between the facets of the scoop and those of the watermelon. Although this objective function is intuitively aligned with the maximization of the scoop volume and thus the number of enclosed seeds, it is somewhat arbitrary. In fact, if a positive factor α_k was multiplied on both sides of the kth constraint in the lower level, then this would equivalently change the weight of t_k from 1 to α_k in the objective function. This observation means that the objective function of the scoop problem may take different definitions, and there may not be one definition that is the most effective for all instances. One could also experiment with nonlinear objective functions, such as the weighted L_2 norm.

In this section, we present an alternative objective function for the type I scoop problem in order to slow down the creation of new nodes. Recall that the variable t in the type I scoop problem indicates the gap between the facets of the scoop and those of the watermelon. If $t_j = 0$, then the jth new node is an empty one, because the bounds computed from (4.13) and (4.14) would not satisfy $l^j \le u^j$. Immediately following the creation of the m_2 new nodes, in Step 1 of the next iteration, all empty nodes will be discarded. As such, we can slow down the creation of new nodes by minimizing the number of positive elements in vector t in the type I scoop problem, which determines the number of nonempty new nodes. For example, if $t^1 = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}^\top$ and $t^2 = \begin{bmatrix} 3 & 4 & 0 & 0 \end{bmatrix}^\top$ are two candidate parameters for the scoop, then t^1 would be preferred by the original type I scoop problem but the alternative version would favor t^2 since it contains fewer positive elements.

We present the alternative type I scoop problem as follows:

$$\min_{\Delta y, t, w} \sum_{j=1}^{m_2} w_j \tag{7.1}$$

$$\text{s.t. } d_2^\top \Delta y \ge 1, \tag{7.2}$$

$$B_2 \Delta y \le t, \tag{7.3}$$

$$0 \le t_j \le (b_2 - A_2 x^{\mathrm{R}} - B_2 y^{\mathrm{R}})_j w_j, \ \forall j \in \{1, \dots, m_2\}, \tag{7.4}$$

$$t \in \mathbb{Z}^{m_2}, \ \Delta y \in \mathbb{Z}^{n_2}, \ w \in \mathbb{B}^{m_2}. \tag{7.5}$$

Here, w is a new binary variable indicating whether ($w_j = 1$) or not ($w_j = 0$) the integer variable t_j is strictly positive. The type II scoop problem is not modified due to the lack of a known upper bound for t. In contrast, the given seed in the type I scoop problem provides a convenient upper bound in (7.4).

8. Comparison with previous algorithms. The major difference between the watermelon algorithm and previous branch-and-bound based BILP algorithms lies in how a relaxation solution is removed from the search space.

In [35], the bilevel linear program (BLP) is iteratively solved as the relaxation, which relaxes the integrality requirements at both levels in (1.1)–(1.4). However, the BLP yields neither a lower bound nor an upper bound of the BILP; thus very conservative branch-and-bound rules are used in [35]. For example, even if the relaxation solution turns out to be integral, the node cannot be immediately fathomed.

In [11], the relaxation (3.2)–(3.4) without any integrality requirements is used to provide an upper bound for the BILP. If the relaxation yields a fractional solution, then branch-and-bound is applied. If the relaxation yields an integral but bilevel infeasible solution, then cutting planes are generated to remove it.

In [45], when the relaxation solution is found to be bilevel infeasible, an optimal solution to the lower level, y^{L}, is used to construct a linear cut for the relaxation:

$$d_2^\top y \geq d_2^\top y^{\mathrm{L}}. \tag{8.1}$$

However, this cut is only valid when x satisfies

$$A_2 x \leq b_2 - B_2 y^{\mathrm{L}}. \tag{8.2}$$

Therefore, the search space is divided into $m_2 + 1$ pieces depending on which of the m_2 constraints in (8.2) is violated, and then the linear cut (8.1) is applied to the one piece that satisfies all the constraints in (8.2). That algorithm and the watermelon algorithm both use a branch-and-bound framework and create multiple new nodes in each iteration.

The two types of scoop problems, which are the key original contribution of this paper, distinguish the watermelon algorithm from [45] and all other previous algorithms. The type I scoop problem designs a polyhedral scoop to remove the relaxation solution and many other seeds, whereas [45] uses a linear cut. Intuitively, the scoop is more efficient than the linear cut in removing seeds, because the scoop has a flexible shape specifically designed to reach out to many seeds in the corners, whereas the linear cut is only valid for one out of $m_2 + 1$ pieces of the search space. The type II scoop problem enables the watermelon algorithm to solve instances like Example 2 that do not have known bounds for the upper level decision variables; these instances are not solvable by any previous algorithm that we are aware of.

Table 1 summarizes the capability of previous algorithms to solve the seven examples from section 3. A checkmark indicates that the algorithm is able to solve the example, whereas the lack of a checkmark means that the example violates one or more simplifying assumptions that the algorithm needs to work. In [11, 30, 35], it is assumed that the feasible region of relaxation (3.2)–(3.5) is nonempty and bounded. Therefore, these algorithms were not designed to solve Examples 1, 2, 5, 6, or 7. In [13] and [19], it is assumed that the lower level variables have known finite bounds. In [28, 34, 36], it is assumed that the upper level, the lower level, and the BILP itself are all bounded. Therefore, these algorithms were not designed to solve Examples 2, 5, 6, or 7. In [45], it is assumed that the upper level variables have known finite bounds. Therefore, this algorithm was not designed to solve Example 2.

In terms of comparison on computational performance, most previous studies on BILP algorithms either did not report their computational performance or did so on small instances. For example, no computational performance of any algorithm was reported in [28]; test instances used in [13], [19], and [30] had no more than a dozen variables and constraints; the largest test instances in [11] and [34] had 34 and 12 variables, respectively; the algorithms in [35] were used to solve instances with up to 40 variables and 16 constraints, and the best version of the algorithm took

TABLE 1
Capability of previous algorithms to solve Examples 1 *to* 7.

Relaxation	BILP	Example	Previous Algorithms								
			[11]	[13]	[19]	[28]	[30]	[34]	[35]	[36]	[45]
infeasible	infeasible	1		✓	✓	✓		✓		✓	✓
optimal	infeasible	2									
		3	✓	✓	✓	✓	✓	✓	✓	✓	✓
	optimal	4	✓	✓	✓	✓	✓	✓	✓	✓	✓
unbounded	infeasible	5									✓
	optimal	6									✓
	unbounded	7									✓

almost two minutes. Nishizaki and Sakawa [36] compared their genetic algorithm with the branch-and-bound algorithm from [35]. They found that the branch-and-bound algorithm was able to solve instances with up to 140 variables in more than 11 hours, and the genetic algorithm was able to solve instances with up to 200 variables in more than one hour. In [45], 100 randomly generated instances with up to 920 variables and 368 constraints (both levels combined) were used in the computational experiments, and the computation times for the largest instances ranged from 10 minutes to 4 hours. In the next section, we compare the watermelon algorithm with those from [11, 35, 45] using the same instances from [45].

9. Computational experiments. In this section, we demonstrate the capability of the watermelon algorithm to correctly and efficiently solve BILP instances. We were able to use the watermelon algorithm to correctly solve all Examples 1 to 7 within a few iterations. We also compared two versions of the watermelon algorithm with three other branch-and-bound based BILP algorithms in terms of their computational efficiency in solving BILP instances of the same type as Example 4. We denote the algorithms in [35], [11], and [45] as `MB`, `DR`, and `XW`, respectively; and we denote the watermelon algorithms with the original type I scoop problem (4.7)–(4.11) and with the alternative type I scoop problem (7.1)–(7.5) as `WaterM-I` and `WaterM-II`, respectively. We have made an effort to implement all the algorithms in the most efficient and consistent way that we could. We used the same data structure and solver settings for the algorithmic framework and only changed the subroutines for the relaxation, cutting plane, and branching scheme to reflect the differences in algorithm design. In our implementation of `MB`, after testing several options, we used the branch-and-bound algorithm for BLP from [3], which yielded the best tradeoff between computational speed and solution quality.

We made three adjustments to accommodate different assumptions made by these algorithms. First, both `MB` and `DR` assume that $B_1 = 0^{m_1 \times n_2}$. In our computational experiments, we first obtained an optimal solution (x^*, y^*) to the BILP (1.1)–(1.4) using `WaterM-I`, and then modified constraint (1.2) to $A_1 x \leq b_1 - B_1 y^*$ for `MB` and `DR` to solve. We realize that the modified BILP could potentially have a different optimal solution than (x^*, y^*), yet this modification makes it easier to find the first bilevel feasible solution. Second, `MB` and `DR` assume a bounded feasible region of the relaxation (3.2)–(3.5), and `XW` requires known finite bounds for the upper level variables as a given parameter. In our computational experiments, all test instances had bounded relaxations. We also fed `XW` with $X = \max\{10x^*, 100\}$ as the upper bound for x, where x^* is an optimal solution obtained using `WaterM-I`. Third, `DR` and `XW` assume (without loss of generality) that both upper and lower level variables are nonnegative, whereas the other three algorithms treat all variables as unrestricted unless otherwise stated

TABLE 2

Average computation time (in seconds) for ten groups of BILP instances. In each group, $n_2 = n_1$ and $m_2 = m_1$. In each cell, the three time values from top to bottom report how long it took to find the first bilevel feasible solution, to find the optimal solution, and to confirm its optimality (to a tolerance of 10^{-6}). Notations D, B, Z *denote depth first, breadth first, and largest-z first traversing strategies.*

Instance	MB			DR			XW			WaterM-I			WaterM-II		
	D	B	Z	D	B	Z	D	B	Z	D	B	Z	D	B	Z
$n_1 = 10$	1	3	1	>481	52	78	0	0	0	0	0	0	0	2	0
$m_1 = 14$	4	14	4	>526	62	116	1	0	0	0	0	0	1	2	1
	36	177	36	>555	120	156	1	1	1	0	0	0	2	2	1
$n_1 = 60$	>999	>977	>911	>876	26	41	0	0	0	0	0	0	0	0	0
$m_1 = 84$	>999	>999	>999	>999	>237	>442	8	6	6	2	1	2	5	8	6
	>999	>999	>999	>999	>388	>470	17	12	11	3	2	2	12	12	10
$n_1 = 110$	>999	>999	>999	>584	92	45	0	0	0	0	0	0	0	0	0
$m_1 = 154$	>999	>999	>999	>819	>285	>215	3	4	7	11	9	8	17	14	13
	>999	>999	>999	>830	>316	>253	25	24	24	13	11	10	26	23	17
$n_1 = 160$	>999	>999	>999	>588	24	78	1	1	1	1	1	1	1	1	1
$m_1 = 224$	>999	>999	>999	>699	111	106	23	12	10	8	9	10	43	18	18
	>999	>999	>999	>774	228	200	48	33	31	14	14	13	49	23	20
$n_1 = 210$	>999	>999	>999	>576	41	62	1	1	1	1	1	1	1	1	1
$m_1 = 294$	>999	>999	>999	>714	263	270	36	58	35	55	10	10	16	13	14
	>999	>999	>999	>807	473	311	138	142	126	65	40	40	26	23	21
$n_1 = 260$	>999	>999	>999	>766	82	37	2	1	1	2	2	2	2	2	2
$m_1 = 364$	>999	>999	>999	>996	205	113	164	93	100	68	32	26	21	18	15
	>999	>999	>999	>999	378	166	248	185	185	93	63	57	42	34	31
$n_1 = 310$	>999	>999	>999	>567	18	104	4	4	4	4	4	4	4	4	4
$m_1 = 434$	>999	>999	>999	>870	246	444	269	250	229	210	85	61	27	44	30
	>999	>999	>999	>999	456	553	429	363	314	293	136	117	45	48	37
$n_1 = 360$	>999	>999	>999	181	39	51	5	5	5	5	5	5	5	5	5
$m_1 = 504$	>999	>999	>999	503	185	236	133	55	91	21	62	63	286	10	12
	>999	>999	>999	648	423	418	191	147	147	111	110	110	297	44	44
$n_1 = 410$	>999	>999	>999	>555	39	141	6	6	6	6	6	6	6	6	6
$m_1 = 574$	>999	>999	>999	>999	>805	>855	171	176	170	67	86	121	153	64	62
	>999	>999	>999	>999	>999	>912	311	308	308	291	290	290	179	81	79
$n_1 = 460$	>999	>999	>999	>843	37	70	9	9	9	9	9	9	9	9	9
$m_1 = 644$	>999	>999	>999	>999	277	>999	271	263	214	529	199	157	76	74	91
	>999	>999	>999	>999	741	>999	473	394	395	680	268	268	95	79	95
average	>999	>999	>999	>602	45	71	3	3	3	3	3	3	3	3	3
	>999	>999	>999	>892	>268	>380	108	92	86	97	49	46	64	27	26
	>999	>999	>999	>978	>453	>449	188	161	154	156	93	91	77	37	36

in the constraints. In our computational experiments, we added $x \geq 0$ and $y \geq 0$ as additional constraints to MB, WaterM-I, and WaterM-II. The number of constraints of the instances reported in Table 2 included these nonnegativity constraints. For each algorithm, three traversing strategies were used to select a new node in Step 1(c): depth first, breadth first, and largest-z first, which are denoted in Table 2 as D, B, and Z, respectively. In particular, the largest-z first strategy means that node k is selected such that $k \in \arg\max_{j\in\{1,\ldots,N\}} z^j$, where z_j records the relaxation objective value of node j's parent node.

We used the same set of 100 randomly generated BMILP instances from [45], except that all variables at both levels are required to be pure integers. These 100 instances were divided into ten groups, each containing ten instances of the same dimension. All algorithms were implemented in Matlab using TOMLAB/CPLEX as the ILP solver. The computational experiments were executed on a desktop computer with a 3 GB RAM and a 2.4 GHz CPU. Computational results are summarized in Table 2. The computation times reported in Table 2 are averages over ten instances within each group. For each group of instances using each traversing strategy of each algorithm, three time values are reported from top to bottom: how long it took to find the first bilevel feasible solution, to find the optimal solution, and to confirm its optimality. The last row in the table reports the average performance over all 100 instances. All average times are rounded to the nearest second. When an algorithm could not finish solving an instance within 999 seconds, we terminated the program prematurely and used 999 seconds for the average time calculation with ">" as a prefix.

We make the following observations from our computational results. First, WaterM-II was less sensitive to the instance dimensions than WaterM-I, since WaterM-II was much faster than WaterM-I for larger instances but took a little more time for smaller ones. This is because the increased computational effort caused by additional binary variables in the alternative type I scoop problem is more likely to be compensated for when a more efficient scoop is applied to larger instances. Second, the largest-z first strategy was slightly better than the breadth first strategy and significantly better than the depth first strategy. Third, XW, WaterM-I, and WaterM-II with all three traversing strategies were able to find the first bilevel feasible solution almost equally quickly. Fourth, WaterM-II was several times faster than XW for BILP problems, although the latter is able to solve BMILP instances for which the former was not designed. Fifth, algorithms XW, WaterM-I, and WaterM-II outperformed DR, which also appeared to be more efficient than MB.

We also tested these five algorithms on 100 interdiction instances with knapsack constraints at the upper level from [33]. In Table 3, we report the computational times using the largest-z first traversing strategy for WaterM-I and WaterM-II. Figures 10 and 11 visualize the process for WaterM-I and WaterM-II to solve one specific instance K50S0W09.KNP. The other three algorithms took more than 999 seconds to solve the majority of the instances. These results further confirm the advantages of the proposed new algorithms over previous ones as well as the effectiveness of the alternative type I scoop problem.

10. Conclusions. In this paper, we tackle the notoriously hard BILP problem with an exact algorithm, the watermelon algorithm, which makes two major contributions to the literature. First, we used a multiway disjunction cut in a branch-and-bound framework to remove bilevel infeasible solutions from the search space. Our computational experiment results suggest that the watermelon algorithm could be

TABLE 3

Average computation time (in seconds) for five groups of interdiction instances with knapsack constraints at the upper level from [33]. *Each group consists of* 20 *instances of the same size with* $n_2 = n_1$ *and* $m_2 = m_1$. *In each cell, the three time values from top to bottom report how long it took to find the first bilevel feasible solution, to find the optimal solution, and to confirm its optimality (to a tolerance of* 10^{-6}*).*

Instance	WaterM-I	WaterM-II
$n_1 = 10$	0	0
$m_1 = 21$	1	1
	1	1
$n_1 = 20$	0	0
$m_1 = 41$	7	6
	11	9
$n_1 = 30$	0	0
$m_1 = 61$	45	27
	58	35
$n_1 = 40$	0	0
$m_1 = 81$	185	52
	>269	81
$n_1 = 50$	0	0
$m_1 = 101$	405	148
	>535	197
	0	0
average	>128	47
	>175	65

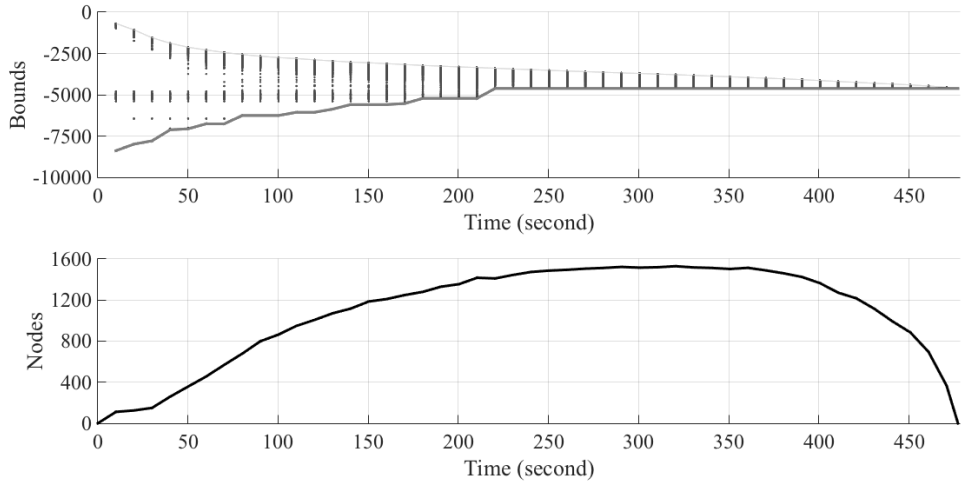

FIG. 10. *The process of solving* K5050W09.KNP *using* WaterM-I. *The top graph shows the upper bound, lower bound, and the z-values of active nodes in* 10*-second intervals, and the lower graph shows the number of active nodes.*

much more efficient than previous branch-and-bound based BILP algorithms. Second, this is the first exact BILP algorithm that does not rely on additional simplifying assumptions. We have proved that, for any BILP instance conforming to the definition, the watermelon algorithm guarantees to provide, in a finite number of iterations, either a global optimal solution or a certificate of infeasibility or unboundedness. It would be a meaningful followup study to extend our computational experiments to a more comprehensive comparison with more algorithms, such as those presented in

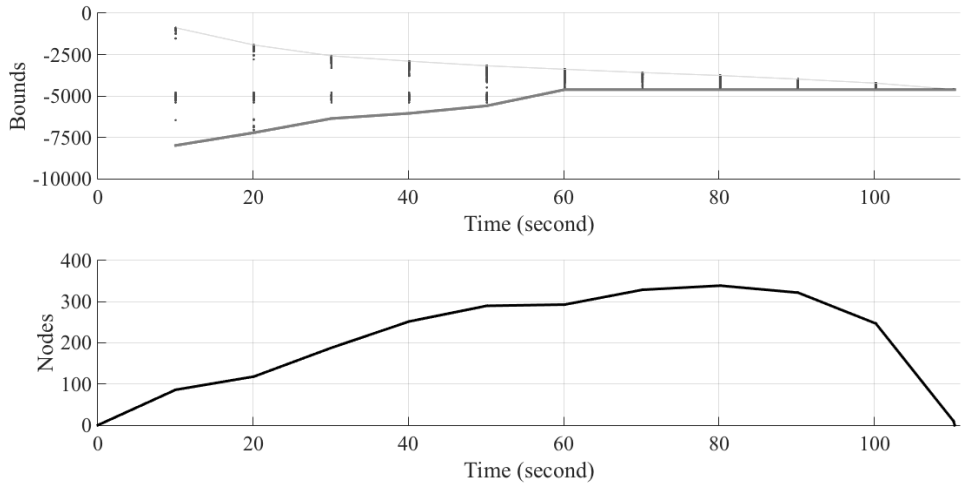

FIG. 11. *The process of solving* K5050W09.KNP *using* WaterM-II. *The top graph shows the upper bound, lower bound, and the z-values of active nodes in* 10*-second intervals, and the lower graph shows the number of active nodes.*

[9, 13, 14, 16, 26, 27, 39, 44, 43]. Future research directions include exploring different variations of the scoop problem and a more comprehensive comparison of their computational performances on larger BILP instances.

Acknowledgments. The authors are indebted to the editors and anonymous reviewers for their insightful and constructive feedback, which greatly improved the quality of this paper.

REFERENCES

[1] C. AUDET, J. HADDAD, AND G. SAVARD, *Disjunctive cuts for continuous linear bilevel programming*, Optim. Lett., 1 (2007), pp. 259–267.
[2] J. F. BARD AND J. FALK, *An explicit solution to the multi-level programming problem*, Comput. Oper. Res., 9 (1982), pp. 77–100.
[3] J. BARD AND J. MOORE, *A branch and bound algorithm for the bilevel programming problem*, SIAM J. Sci. Stat. Comput., 11 (1990), pp. 281–292.
[4] D. BERTSIMAS, E. LITVINOV, X. SUN, J. ZHAO, AND T. ZHENG, *Adaptive robust optimization for the security constrained unit commitment problem*, IEEE Trans. Power Syst., 28 (2012), pp. 52–63.
[5] W. F. BIALAS AND M. H. KARWAN, *On two-level optimization*, IEEE Trans. Autom. Control, AC-27, (1982), pp. 211–214.
[6] G. BROWN, M. CARLYLE, J. SALMERÓN, AND K. WOOD, *Defending critical infrastructure*, Interfaces, 36 (2006), pp. 530–544.
[7] P. CAPPANERA AND M. P. SCAPARRA, *Optimal allocation of protective resources in shortest-path networks*, Transp. Sci., 45 (2011), pp. 64–80.
[8] A. CAPRARA, M. CARVALHO, A. LODI, AND G. J. WOEGINGER, *Bilevel knapsack with interdiction constraints*, INFORMS J. Comput., 28 (2016), pp. 319–333.
[9] M. CARAMIA AND R. MARI, *Enhanced exact algorithms for discrete bilevel linear problems*, Optim. Lett., 9 (2015), pp. 1447–1468.
[10] S. DEMPE, V. KALASHNIKOV, AND R. Z. RÍOS-MERCADO, *Discrete bilevel programming: Application to a natural gas cash-out problem*, Eur. J. Oper. Res., 166 (2005), pp. 469–488.
[11] S. T. DENEGRE AND T. K. RALPHS, *A branch-and-cut algorithm for integer bilevel linear programs*, Oper. Res. Cyber Infrastruct., 47 (2009), pp. 65–78.
[12] L. E. DICKSON, *Finiteness of the odd perfect and primitive abundant numbers with n distinct prime factors*, Am. J. Math., 35 (1913), pp. 413–422.

[13] L. F. Domínguez and E. F. Pistikopoulos, *Multiparametric programming based algorithms for pure integer and mixed-integer bilevel programming problems*, Comput. Chem. Eng., 34 (2010), pp. 2097–2106.
[14] N. P. Faísca, V. Dua, B. Rustem, P. M. Saraiva, and E. N. Pistikopoulos, *Parametric global optimisation for bilevel programming*, J. Global Optim., 38 (2007), pp. 609–623.
[15] K. Fischer, *Sequential discrete p-facility models for competitive location planning*, Ann. Oper. Res., 111 (2002), pp. 253–270.
[16] M. Fischetti, I. Ljubić, M. Monaci, and M. Sinnl, *Intersection cuts for bilevel optimization*, in IPCO 2016 Proceedings, Mathematical Optimization Society, 2016.
[17] Z. Gao, J. Wu, and H. Sun, *Solution algorithm for the bi-level discrete network design problem*, Transp. Res., Part B, 39 (2005), pp. 479–495.
[18] P. Garcia-Herreros, L. Zhang, P. Misra, E. Arslan, S. Mehta, and I. E. Grossmann, *Mixed-integer bilevel optimization for capacity planning with rational markets*, Comput. Chem. Eng., 86 (2016), pp. 33–47.
[19] Z. H. Gümüş and C. A. Floudas, *Global optimization of mixed-integer bilevel programming problems*, Comput. Manag. Sci., 2 (2005), pp. 181–212.
[20] P. Hansen, B. Jaumard, and G. Savard, *New branch-and-bound rules for linear bilevel programming*, SIAM J. Sci. Stat. Comput., 13 (1992), pp. 1194–1217.
[21] Y. He, L. Wang, and J. Wang, *Comparing the effectiveness of cap-and-trade and carbon taxes policies in generation expansion planning*, Comput. Ind. Eng., 63 (2012), pp. 708–716.
[22] M. Hemmati and J. C. Smith, *A mixed-integer bilevel programming approach for a competitive prioritized set covering problem*, Discrete Optim., 20 (2016), pp. 105–134.
[23] J. Hu, J. E. Mitchell, J. S. Pang, K. P. Bennett, and G. Kunapuli, *On the global solution of linear programs with linear complementarity constraints*, SIAM J. Optim., 19 (2008), pp. 445–471.
[24] R. Jiang, J. Wang, and Y. Guan, *Robust unit commitment with wind power and pumped storage hydro*, IEEE Trans. Power Syst., 27 (2012), pp. 800–810.
[25] J. J. Júdice and A. M. Faustino, *A sequential LCP method for bilevel linear programming*, Ann. Oper. Res., 34 (1992), pp. 89–106.
[26] P.-M. Kleniati and C. S. Adjiman, *Branch-and-sandwich: A deterministic global optimization algorithm for optimistic bilevel programming problems. Part I: Theoretical development*, J. Global Optim., 60 (2014), pp. 425–458.
[27] P.-M. Kleniati and C. S. Adjiman, *Branch-and-sandwich: A deterministic global optimization algorithm for optimistic bilevel programming problems. Part II: Convergence analysis and numerical results*, J. Global Optim., 60 (2014), pp. 459–481.
[28] M. Köppe, M. Queyranne, and C. T. Ryan, *Parametric integer programming algorithm for bilevel mixed integer programs*, J. Optim. Theory Appl., 146 (2010), pp. 1–11.
[29] H. Küçükaydin, N. Aras, and I. K. Altinel, *A leader–follower game in competitive facility location*, Comput. Oper. Res., 39 (2012), pp. 437–448.
[30] H. Li, Y. Wang, and X. Li, *An evolutionary algorithm for nonlinear bilevel programming problems based on a new penalty method*, in Fourth International Conference on Natural Computation, IEEE Computer Society, Washington, 2008, pp. 531–535.
[31] C. Losada, M. P. Scaparra, and J. R. O'Hanley, *Optimizing system resilience: A facility protection model with recovery time*, Eur. J. Oper. Res., 217 (2012), pp. 519–530.
[32] Z. Lukač, K. Šorić, and V. V. Rosenzweig, *Production planning problem with sequence dependent setups as a bilevel programming problem*, Eur. J. Oper. Res., 187 (2008), pp. 1504–1512.
[33] *Detailed Bilevel Instances–MCDM*, http://coral.ise.lehigh.edu/data-sets/bilevel-instances/detailed-bilevel-instances-mcdm.
[34] A. Mitsos, *Global solution of nonlinear mixed-integer bilevel programs*, J. Global Optim., 47 (2010), pp. 557–582.
[35] J. Moore and J. Bard, *The mixed integer linear bilevel programming problem*, Oper. Res., 38 (1990), pp. 911–921.
[36] I. Nishizaki and M. Sakawa, *Computational methods through genetic algorithms for obtaining Stackelberg solutions to two-level integer programming problems*, Electron. Commun. Japan, 86 (2003), pp. 59–66.
[37] H. Önal, *A modified simplex approach for solving bilevellinear programming problems*, Eur. J. Oper. Res., 67 (1993), pp. 126–135.
[38] E. Rashidi, M. Parsafard, H. Medal, and X. Li, *Optimal traffic calming: A mixed-integer bi-level programming model for locating sidewalks and crosswalks in a multimodal transportation network to maximize pedestrians' safety and network usability*, Transp. Res., Part E, 91 (2016), pp. 33–50.

[39] G. K. D. Saharidis, A. J. Conejo, and G. Kozanidis, *Exact Solution Methodologies for Linear and (Mixed) Integer Bilevel Programming*, in Metaheuristics for Bi-level Optimization, Springer, New York, 2013, pp. 221–245.
[40] K. C. Sarma and H. Adeli, *Bilevel parallel genetic algorithms for optimization of large steel structures*, Comput. Aided Civil Infrastruct. Eng., 16 (2001), pp. 295–304.
[41] M. P. Scaparra and R. L. Church, *A bilevel mixed-integer program for critical infrastructure protection planning*, Comput. Oper. Res., 35 (2008), pp. 1905–1923.
[42] O. Stein and G. Still, *On generalized semi-infinite optimization and bilevel optimization*, Eur. J. Oper. Res., 142 (2002), pp. 444–462.
[43] U. P. Wen and A. D. Huang, *A simple tabu search method to solve the mixed-integer linear bilevel programming problem*, Eur. J. Oper. Res., 88 (1996), pp. 563–571.
[44] U. P. Wen and Y. H. Yang, *Algorithms for solving the mixed integer two-level linear programming problem*, Comput. Oper. Res., 17 (1990), pp. 133–142.
[45] P. Xu and L. Wang, *An exact algorithm for the bilevel mixed integer linear programming problem under three simplifying assumptions*, Comput. Oper. Res., 41 (2014), pp. 309–318.
[46] Y. Yao, T. Edmunds, D. Papageorgiou, and R. Alvarez, *Trilevel optimization in power network defense*, IEEE Trans. Syst. Man Cybern., 37 (2007), pp. 712–718.
[47] S. Zein and M. Bruyneel, *A bilevel integer programming method for blended composite structures*, Adv. Eng. Softw., 79 (2015), pp. 1–12.
[48] Y. Zhou, L. Wang, and J. D. McCalley, *Designing effective and efficient incentive policies for renewable energy in generation expansion planning*, Appl. Energy, 88 (2011), pp. 2201–2209.