Contents lists available at ScienceDirect

Computers and Chemical Engineering

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

A generalization of the Branch-and-Sandwich algorithm: From continuous to mixed-integer nonlinear bilevel problems

Polyxeni-M. Kleniati[1], Claire S. Adjiman*

Centre for Process Systems Engineering, Department of Chemical Engineering, Imperial College London, London SW7 2AZ, United Kingdom

ARTICLE INFO

Article history:
Received 29 January 2014
Received in revised form 6 June 2014
Accepted 10 June 2014
Available online 19 June 2014

Keywords:
Bilevel optimization
Deterministic global optimization
Branch-and-bound
Optimal value function

ABSTRACT

We propose a deterministic global optimization algorithm for mixed-integer nonlinear bilevel problems (MINBP) by generalizing the Branch-and-Sandwich algorithm (Kleniati and Adjiman, 2014a). Advances include the removal of regularity assumptions and the extension of the algorithm to mixed-integer problems. The proposed algorithm can solve very general MINBP problems to global optimality, including problems with inner equality constraints that depend on the inner and outer variables. Inner lower and inner upper bounding problems are constructed to bound the inner optimal value function and provide constant-bound cuts for the outer bounding problems. To remove the need for regularity, we introduce a robust counterpart approach for the inner upper bounding problem. Branching is allowed on all variables without distinction by keeping track of refined partitions of the inner space for every refined subdomain of the outer space. Finite ε-convergence to the global solution is proved. The algorithm is applied successfully to 10 mixed-integer literature problems.

1. Introduction

Many of the optimization problems relevant to chemical engineering are bilevel problems in nature (Grossmann and Biegler, 2004; Gümüş and Floudas, 2005; Floudas and Gounaris, 2009) and can thus be framed as two-person, hierarchical optimization problems having a second optimization problem as part of the constraints. Examples of such formulations include design under uncertainty (Floudas et al., 2001; Ryu et al., 2004), the estimation of parameters in thermodynamic models (Mitsos et al., 2009), the optimization of processes involving phase equilibrium (Kamath et al., 2010) and/or chemical equilibrium (Clark and Westerberg, 1990; Sahin et al., 1998), simultaneous process optimization and heat integration (Kamath et al., 2012), and strategies for product pricing and marketing in competitive markets (Lemonidis, 2008).

In this work, the following mixed-integer nonlinear bilevel problem is considered:

$$\begin{aligned}
\min_{x_i,x_c,y_i,y_c} \quad & F(x_i,x_c,y_i,y_c) \\
\text{s.t.} \quad & G(x_i,x_c,y_i,y_c) \le 0, \\
& (y_i,y_c) \in \arg\min_{y_i\in Y_I, y_c\in Y_C} \{f(x_i,x_c,y_i,y_c) \ \text{ s.t. } \ g(x_i,x_c,y_i,y_c)\le 0\}, \\
& x_i \in X_I \subset \mathbb{Z}^{n_1}, x_c \in X_C \subset \mathbb{R}^{n-n_1}, \\
& y_i \in Y_I \subset \mathbb{Z}^{m_1}, y_c \in Y_C \subset \mathbb{R}^{m-m_1},
\end{aligned} \tag{1}$$

where subscripts i and c stand for *integer* and *continuous*, respectively. Thus, the n-dimensional vector (x_i, x_c) denotes the outer (leader) variables, where the first n_1 components are integer and the rest are continuous. Similarly, the m-dimensional vector (y_i, y_c) denotes the inner (follower) variables, where the first m_1 components are integer and the rest are continuous. Functions $F, f : \mathbb{Z}^{n_1} \times \mathbb{R}^{n-n_1} \times \mathbb{Z}^{m_1} \times \mathbb{R}^{m-m_1} \to \mathbb{R}$, $G : \mathbb{Z}^{n_1} \times \mathbb{R}^{n-n_1} \times \mathbb{Z}^{m_1} \times \mathbb{R}^{m-m_1} \to \mathbb{R}^p$ and $g : \mathbb{Z}^{n_1} \times \mathbb{R}^{n-n_1} \times \mathbb{Z}^{m_1} \times \mathbb{R}^{m-m_1} \to \mathbb{R}^r$ denote the outer/inner objective and outer/inner constraint functions. The functions are assumed to be twice continuously differentiable when the integrality condition is relaxed. Finally, the host sets for the continuous variables are closed and bounded: $X_C = [x_c^{\mathrm{L}}, x_c^{\mathrm{U}}] \subset \mathbb{R}^{n-n_1}$ and $Y_C = [y_c^{\mathrm{L}}, y_c^{\mathrm{U}}] \subset \mathbb{R}^{m-m_1}$.

Remark 1. We adopt the so-called optimistic formulation of the bilevel problem that implies some cooperation between the leader and the follower. In particular, for different globally optimal solutions in the inner problem to which the follower is indifferent, we

* Corresponding author. Tel.: +44 20 7594 6638; fax: +44 20 7594 6606.
E-mail address: c.adjiman@imperial.ac.uk (C.S. Adjiman).

[1] Current address: Process Systems Enterprise, 26-28 Hammersmith Grove, London W6 7HA, United Kingdom.

http://dx.doi.org/10.1016/j.compchemeng.2014.06.004

optimize in favor of the leader's (outer) objective and constraints; hence, the outer minimization in (1) is with respect to the whole set of variables. For more details on the optimistic formulation and its alternative pessimistic formulation, the interested reader is referred to Bard (1998), Dempe (2002), Loridan and Morgan (1996), and Wiesemann et al. (2013).

Problem (1) has been tackled for specific classes of the participating functions. For instance, for mixed discrete and continuous decision variables and linear functions, the problem has been addressed in Moore and Bard (1990), Bard (1998) and, using parametric programming, in Faísca et al. (2007). Furthermore, nonlinear but convex outer and inner objective functions were considered in Edmunds and Bard (1992) and the purely discrete linear bilevel program was analyzed in Vicente et al. (1996). Advances in theory and algorithms for mixed-integer and purely discrete linear bilevel problems are also found in Dempe (2002), Dempe and Kalashnikov (2004), Dempe et al. (2005), Fanghänel and Dempe (2009), while advances for more general classes can be found in Gümüş and Floudas (2005) and Mitsos (2010). In particular, Gümüş and Floudas (2005) tackled first a class of mixed-integer nonlinear bilevel programs with polynomial terms in the inner discrete variables and linear terms in the inner continuous variables (at the lower level). More recently, Mitsos (2010) addressed general mixed-integer nonlinear bilevel programs, limited only by the absence of inner equality constraints that depend on the outer variables.

In the present article, we assume only twice differentiability of the functions when the integrality condition is relaxed and compactness of the host sets. We assume neither a special form nor convexity of the functions involved. Furthermore, we allow outer and inner equality constraints, but we subsume those within the set of inequality constraints both in the outer and the inner problems for the sake of a simpler presentation. The use of inner equality constraints that depend on the outer variables in the context of such a general formulation has, to the best of our knowledge, not been considered previously in the published literature. As in the general nonconvex case (Mitsos et al., 2008; Kleniati and Adjiman, 2014a), problem (1) can be written equivalently as follows (Mitsos, 2010; Dempe and Zemkoho, 2011):

$$\begin{aligned} \min_{x_i,x_c,y_i,y_c} \quad & F(x_i,x_c,y_i,y_c) \\ \text{s.t.} \quad & G(x_i,x_c,y_i,y_c) \le 0, \\ & g(x_i,x_c,y_i,y_c) \le 0, \\ & f(x_i,x_c,y_i,y_c) \le w(x_i,x_c), \\ & x_i \in X_I,\ x_c \in X_C, \\ & y_i \in Y_I,\ y_c \in Y_C, \end{aligned} \tag{2}$$

where $w(x_i, x_c)$ is the optimal value function of the inner (mixed-integer nonlinear) problem:

$$w(x_i,x_c) = \min_{y_i \in Y_I, y_c \in Y_C} \{f(x_i,x_c,y_i,y_c) \quad \text{s.t.} \quad g(x_i,x_c,y_i,y_c) \le 0\}. \tag{3}$$

Remark 2. Observe that by reformulation (2) and particularly by the third constraint, i.e.,

$$f(x_i,x_c,y_i,y_c) \le w(x_i,x_c),$$

a restriction of problem (3), i.e., an upper bound on $w(x_i, x_c)$, yields a relaxation of (2) and consequently, a relaxation of (1). Similarly, a relaxation of (3) gives a restriction of (2) (Mitsos and Barton, 2006; Mitsos et al., 2008).

Part of our proposed method, Branch-and-Sandwich, is based on the property of Remark 2. For instance, in the original development of the Branch-and-Sandwich algorithm (Kleniati and Adjiman, 2014a,b), where only continuous variables were considered, we computed a constant upper bound on $w(x_c)$ for all the x_c values over the domain under consideration. Then, using this upper bound on $w(x_c)$ in formulation (2), we derived a relaxation of the overall problem. This relaxation played the role of our proposed lower bounding problem. In order to compute a valid upper bound on $w(x_c)$ for all the x_c values, we employed a semi-infinite formulation for the proposed inner upper bounding problem which we tackled via its tractable KKT relaxation. To achieve this, in the original development of the algorithm, a regularity condition was imposed for all the x_c values. Then, it was possible to employ the inner KKT conditions where necessary, e.g., in the inner upper bounding problem and in the overall lower bounding problem.

In the present article, we present a generalization of the Branch-and-Sandwich algorithm to the mixed-integer nonlinear case that removes the need to employ the inner KKT conditions; hence, the regularity assumption of the original algorithm is lifted. This is achieved with the introduction of a robust counterpart approach for the inner upper bounding problem. Nevertheless, when there are continuous variables in the inner problem and when regularity holds for this problem, the inner KKT conditions with respect to the continuous variables can be derived (Mitsos, 2010) and added to the outer problem in order to obtain tighter relaxations (Kleniati and Adjiman, 2014a).

The paper is organized as follows. In Section 2, the reader is introduced to the challenges one faces when tackling general (mixed-integer) nonlinear bilevel problems. The proposed branching scheme is presented in Section 3.1, and the bounding scheme in Section 3.2, together with rules to prune non-promising nodes (fathoming). The Branch-and-Sandwich algorithm is formally stated in Section 4. This exposition is followed by its detailed application to an illustrative example in Section 4.2. In Section 4.3, a proof of finite convergence to an ε-optimal solution is reported. Preliminary numerical results are presented in Section 5 and our conclusions in Section 6.

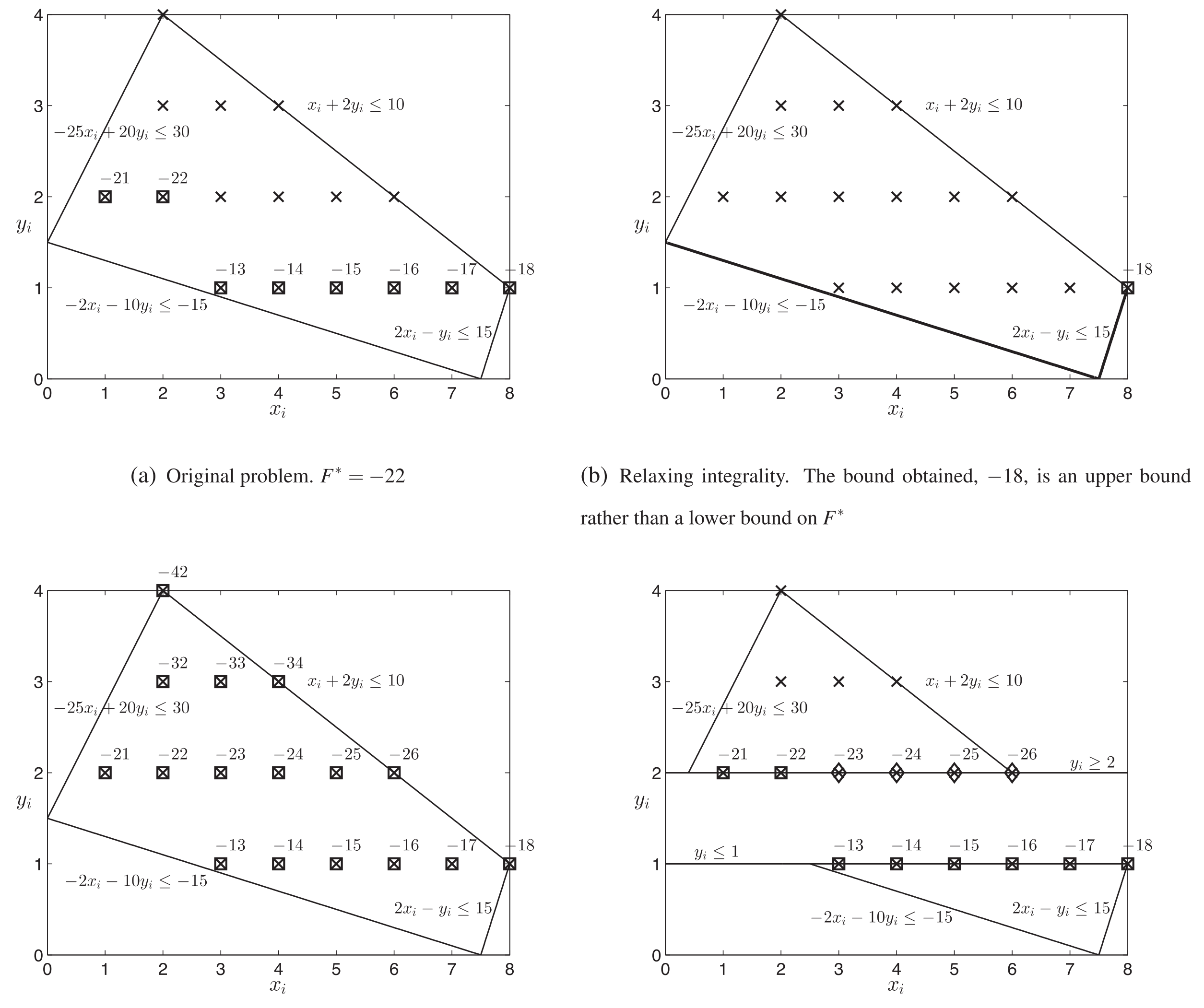
2. Challenges

Bilevel problems are very hard problems to solve: from the complexity point of view they are *NP*-hard even in the linear case (Deng, 1998). To make matters worse, in the discrete case, additional difficulties may arise due to the integrality conditions which make the inner problem nonconvex, regardless of the form of the functions. Observe that, in such a case, the (nonconvex) inner problem is an *NP*-hard problem embedded within an *NP*-hard problem.

Let us visualize some of the challenges encountered due to the integrality conditions and the resulting nonconvexity of the inner problem by using the well-known discrete linear bilevel example below, which provides a useful illustration (Gümüş and Floudas, 2005).

Example 1 *(Example 1 in Moore and Bard (1990)).*

$$\begin{aligned} \min_{x_i} \quad & -x_i - 10y_i \\ \text{s.t.} \quad & \min_{y_i} y_i, \\ & \text{s.t.} \qquad -25x_i + 20y_i \le 30, \\ & \qquad\qquad x_i + 2y_i \le 10, \\ & \qquad\qquad 2x_i - y_i \le 15, \\ & \qquad\qquad -2x_i - 10y_i \le -15, \\ & \qquad\qquad x_i \in [0,8] \cap \mathbb{Z}, \\ & \qquad\qquad y_i \in [0,4] \cap \mathbb{Z}. \end{aligned} \tag{4}$$



(a) Original problem. $F^* = -22$

(b) Relaxing integrality. The bound obtained, -18, is an upper bound rather than a lower bound on F^*

(c) Well-known relaxation. Constructing a convergent lower bound is challenging, making it difficult to improve on the lower bound of $-42 < F^*$

(d) Partitioning the inner space. Global solutions of the inner problems over subdomains of Y, e.g., -26 for $2 \leq y_i \leq 4$, do not always yield valid upper bounds on F^*

Fig. 1. Challenges illustrated using Example 1. Crosses denote feasible points of the original inner problem. Thick solid lines and boxed crosses denote bilevel feasible points for the problems considered in each figure. The number next to each symbol is the corresponding outer objective value. Diamond shaped symbols in (d) denote inner global optima over a subdomain of Y; those points are bilevel infeasible.

We use F^* to denote the outer objective value at the global solution, $\underline{F}$ and $\overline{F}$ to denote lower and upper bounds on F^*. Example 1 has a unique optimal solution at $(x_i^*, y_i^*) = (2, 2)$ with optimal objective value $F^* = -22$ as one can verify in Fig. 1(a). This example has been used extensively in the literature to illustrate that relaxing the integrality requirement of the inner problem does not lead to a valid lower bound on the overall objective value (Dempe, 2002). This phenomenon is demonstrated in Fig. 1(b), where the reaction set of the relaxed inner problem is highlighted and the overall optimal objective value is -18, which is an upper bound on F^*, rather than a lower bound. This result can in fact be explained by the observation of Remark 2, i.e., a relaxation of the inner problem yields a restriction of reformulation (2).

The next challenge arises from the difficulty to construct a convergent lower bounding problem (Mitsos and Barton, 2006). In Fig. 1(c), the feasible region of the well-known relaxed (overall) problem, which is obtained by dropping the constraint that variable y_i is the optimal solution of the inner problem, is plotted. The resulting problem is:

$$\begin{aligned} \min_{x_i} \quad & -x_i - 10y_i \\ \text{s.t.} \quad & -25x_i + 20y_i \leq 30, \\ & x_i + 2y_i \leq 10, \\ & 2x_i - y_i \leq 15, \\ & -2x_i - 10y_i \leq -15, \\ & x_i \in [0, 8] \cap \mathbb{Z}, \\ & y_i \in [0, 4] \cap \mathbb{Z}. \end{aligned} \tag{5}$$

The optimal value of (5), equal to -42, is a valid lower bound very far from F^* that cannot be improved unless more constraints are added. In particular, relaxation (5) is non convergent (Mitsos and Barton, 2006). We see in Section 3.2 how Branch-and-Sandwich adds one simple constraint/cut in order to construct a convergent relaxation. The final challenge, demonstrated in Fig. 1(d), concerns the effect of partitioning the inner space. If branching on y_i were allowed, the hierarchical nature of the bilevel problem would be violated because then, for certain x_i values, not all values in Y would have been considered. Notice in Fig. 1(a), for $x_i = \{3, \ldots, 6\}$, the unique optimal solution of the inner problem is $y_i = 1$. However, if branching on y_i were allowed, as depicted in Fig. 1(d), then over the Y subdomain where $y \geq 2$, the unique optimal solution of the inner problem would be $y_i = 2$. Accepting points $(x_i, 2)$ for $x_i = \{3, \ldots, 6\}$ as bilevel *feasible*, based on these points being global optimal solutions of the inner problem over a subdomain, would lead to the selection of bilevel *infeasible* points, i.e., infeasible in (4). Consequently, the corresponding objective values are invalid upper bounds on the outer objective value, namely $\{-23, \ldots, -26\}$ are not valid upper bounds on F^*. One strategy that has been proposed to overcome this challenge is to ensure that the (nonconvex) inner program is always solved over the whole Y domain (Mitsos et al., 2008; Tsoukalas, 2009). However, in Branch-and-Sandwich this challenge is addressed in a different way. Namely, the successively refined inner subdomains, where overall inner global optima may lie, corresponding to successively refined outer subdomains, are kept under consideration. Thus, although branching with respect to the inner and the outer variables is allowed without distinction, an inner suboptimal point is never computed. The details are provided in Section 3.1 and a thorough analysis can be found in Kleniati and Adjiman (2014a).

3. Branching and bounding schemes

The original Branch-and-Sandwich algorithm is a deterministic global optimization algorithm that tackles general bilevel programs with continuous decision variables, twice continuously differentiable functions and a nonconvex inner problem satisfying regularity (Kleniati and Adjiman, 2011, 2014a, 2014b). The Branch-and-Sandwich algorithm was proved to be ε-convergent in Theorem 6 of Kleniati and Adjiman (2014b) based on exhaustiveness and the general convergence theory (Horst and Tuy, 1996). This means that the algorithm is guaranteed to converge in finite time to a point (x^*, y^*) which is feasible with respect to the inner and outer problem constraints and which is such that $f(x^*, y^*) \leq w(x^*) + \varepsilon_f$ and $F(x^*, y^*) \leq F^* + \varepsilon_F$, where ε_f and ε_F are arbitrary positive constants, $w(x^*)$ is the optimal objective value of the inner problem at x^* and F^* is the value of the outer objective at the global solution, as per Definition 2 of Kleniati and Adjiman (2014a). The algorithm was tested successfully on 34 benchmark examples (Kleniati and Adjiman, 2014b). Its main components are:

i. partition of both the inner and the outer spaces without distinction,
ii. consideration of all inner (successively refined) subdomains, where (inner) global optima may lie for (successively refined) outer subdomains,
iii. solution of convex relaxations of the inner problem over refined inner subdomains,
iv. generation of guaranteed upper and lower bounds on the inner optimal value function and on the outer objective value for a given subdomain,[2] and
v. a novel tree management with auxiliary lists of nodes as well as inner and outer fathoming rules such that the exploration of *two* decision spaces using a *single* branch-and-bound tree is achieved.

[2] Two of the proposed bounding problems are nonconvex. The proposed bounding problems do not grow in size from node to node and are always obtained from the corresponding problems at the parent node.

For the mixed-integer case in the present paper, the construction of two sets of upper and lower bounding problems, approximating the inner and the outer problems, is again combined with the new tree management with auxiliary lists of nodes and (inner/outer) fathoming rules such that the two decision spaces, inner and outer, are explored simultaneously. Before describing the branching and bounding schemes, let us first define below the meaning of a node.

Definition 1 *(Node).* A node k represents (sub)domain

$$(X^{(k)} \times Y^{(k)}) \subseteq (X \times Y).$$

The root node is the node with $k = 1$ and corresponds to the whole domain $X \times Y$. Note also that each node k has an attribute $l^{(k)}$ that denotes its level (depth) in the branch-and-bound tree. The root node has level zero, i.e., $l^{(1)} = 0$.

3.1. Branching scheme

The design of our branching scheme is such that it allows branching on the outer and inner variables without distinction but at the same time considering all inner (successively refined) subdomains, where (inner) global optima lie for (successively refined) outer subdomains. This approach and its theoretical implications are fully developed in our earlier work, where the original Branch-and-Sandwich was introduced (Kleniati and Adjiman, 2014a, 2014b) and are summarized here for completeness. Notice that it is not affected by the presence of integrality conditions, so we simplify our notation for the outer (inner) host set from $X_I \times X_C$ ($Y_I \times Y_C$) to X (Y). Furthermore, in the examples below, let a one-dimensional host set X (Y) be represented by an interval, independent of whether or not it is continuous. For instance, if $Y = \{1, 2, 3\}$, then in our presentation we have $Y = [1, 3]$, where $Y \subset \mathbb{Z}$ is implied.

The first idea of the proposed branching scheme entails the use of more than one list to represent the list of *active nodes*. By active nodes we refer to the nodes that require further branching. In the context of our solution method, we have two types of active nodes, *open* and *inner-open* (or *outer-fathomed*) nodes. By *open* nodes we refer to nodes that can be explored further with respect to the overall problem that we intend to solve, i.e., the bilevel problem including the outer and the inner problems. These nodes belong to list $\mathcal{L}$, which corresponds to the classical list of active nodes in the context of a classical branch-and-bound method. By *inner-open* (or *outer-fathomed*) nodes we denote the nodes that are known (based on fathoming rules, cf., Section 3.2.1) not to contain the global solution of the bilevel problem. These nodes may not be ready to be discarded fully from the branch-and-bound tree, because they may still contain a globally optimal solution for the inner problem for a subdomain of X which has not yet been fathomed. These nodes are stored in list $\mathcal{L}_{\text{In}}$, which is the list of *inner-open* nodes, i.e., nodes that should be explored further with respect to the inner problem only. We also refer to $\mathcal{L}_{\text{In}}$ as the list of outer-fathomed nodes, implying that the nodes it contains are not included in $\mathcal{L}$. When fathoming is applied (cf., Section 3.2.1), there are also instances in which some active nodes need not be explored further with respect to either space (outer or inner) and can be deleted from any list to which they belong; this is referred to as *full fathoming* of a node and corresponds to the classical fathoming in a branch-and-bound method.

Table 1
Independent lists appearing in each partitioning example shown in Fig. 2.

	$\mathcal{L}^1$	$\mathcal{L}^2$	$\mathcal{L}^3$
Fig. 2(a)	$\{\mathcal{L}_1^1\}$	–	–
Fig. 2(b)	$\{\mathcal{L}_1^1, \mathcal{L}_2^1\}$	–	–
Fig. 2(c)	$\{\mathcal{L}_1^1\}$	$\{\mathcal{L}_1^2\}$	–
Fig. 2(d)	$\{\mathcal{L}_1^1\}$	$\{\mathcal{L}_1^2\}$	$\{\mathcal{L}_1^3\}$

Remark 3. Lists $\mathcal{L}$ and $\mathcal{L}_{\text{In}}$ are disjoint, i.e.,: $\mathcal{L} \cap \mathcal{L}_{\text{In}} = \emptyset$. Their union contains all the nodes of the branch-and-bound tree that have not yet been fully fathomed, i.e., all active nodes.

The second idea of the proposed branching scheme is the use of a number of auxiliary lists that link the active nodes in $\mathcal{L} \cup \mathcal{L}_{\text{In}}$ appropriately such that the hierarchical structure of the bilevel problem is maintained during branching. In what follows, let P be a finite index set and $\{\mathcal{X}_p \subseteq X : p \in P\}$ be a *partition* of X:

$$X = \bigcup_{p=1}^{|P|} \mathcal{X}_p \quad \text{and} \quad \mathcal{X}_p \cap \mathcal{X}_q = \partial\mathcal{X}_p \cap \partial\mathcal{X}_q \quad \text{for all} p, q \in P, p \neq q, \tag{6}$$

where $\partial\mathcal{X}_p$ denotes the (relative) boundary of $\mathcal{X}_p$ (Horst and Tuy, 1996, Def. IV.1). In the Branch-and-Sandwich framework, in addition to lists $\mathcal{L}$ and $\mathcal{L}_{\text{In}}$, we assign one auxiliary list $\mathcal{L}^p$ to each member $\mathcal{X}_p$ of the X partition.[3] Each list $\mathcal{L}^p$ consists of a collection of *sublists*:

$$\mathcal{L}^p = \{\mathcal{L}_1^p, \ldots, \mathcal{L}_{|S^p|}^p\}, \tag{7}$$

where S^p, $p \in P$, are finite sets of indices. The lists $\mathcal{L}^p$, $p \in P$, are pairwise disjoint and, for this reason, are referred to as *independent*:

$$\mathcal{L}^q \cap \mathcal{L}^p = \emptyset \quad \text{for all} \quad p, q \in P, \quad p \neq q. \tag{8}$$

In order to generate list $\mathcal{L}^p$, corresponding to set $\mathcal{X}_p$, $p \in P$, we first generate sublist(s)

$$\mathcal{L}_s^p := \{k \in \mathcal{L} \cup \mathcal{L}_{\text{In}} : \text{ri}(X^{(k)}) \cap \text{ri}(\mathcal{X}_p) \neq \emptyset\}, \quad s \in S^p, \tag{9}$$

such that

$$\text{ri}(X^{(i)}) \cap \text{ri}(X^{(j)}) \neq \emptyset \quad \text{for all} \quad i, j \in \mathcal{L}_s^p, \quad i \neq j, \quad s \in S^p, \tag{10}$$

$$\text{ri}(Y^{(i)}) \cap \text{ri}(Y^{(j)}) = \emptyset \quad \text{for all} \quad i, j \in \mathcal{L}_s^p, \quad i \neq j, \quad s \in S^p, \tag{11}$$

where ri(·) denotes the (relative) interior of a set (Boyd and Vandenberghe, 2004). In fact, observe that each sublist $\mathcal{L}_s^p$, $s \in S^p$, corresponds to a Y partition.

Remark 4. To recapitulate, each independent list $\mathcal{L}^p$, $p \in P$, covers all x values in the member set $\mathcal{X}_p$ of the X partition, and the "whole" Y as a collection of $|S^p|$ Y partitions:[4]

$$\mathcal{Y}_p = \bigcup_{s \in S^p} \bigcup_{k \in \mathcal{L}_s^p} Y^{(k)}. \tag{12}$$

In other words, in the Branch-and-Sandwich framework, the set $\{\mathcal{X}_p : p \in P\}$ cannot be any X partition, but has to satisfy the requirement that for all $x \in \mathcal{X}_p$ the "whole" Y is maintained. This requirement is met via the management of the independent lists (7) and their sublists (9).

Example 1 continued Consider the four partitioning examples of a purely integer two-dimensional space $X \times Y$ in Fig. 2. The corresponding independent lists and their sublists are identified in Tables 1 and 2, respectively. Notice that for a member set $\mathcal{X}_p$, $p \in P$,

[3] Overall, we have lists $\mathcal{L}$, $\mathcal{L}_{\text{In}}$ and $\{\mathcal{L}^1, \ldots, \mathcal{L}^{|P|}\}$.

[4] We use quotes to refer to the "whole" inner space because some Y subdomains are eliminated at some point due to fathoming (cf., Section 3.2.1). As a result, we may have $\mathcal{Y}_p \subset Y$, rather than $\mathcal{Y}_p = Y$, for some $p \in P$.

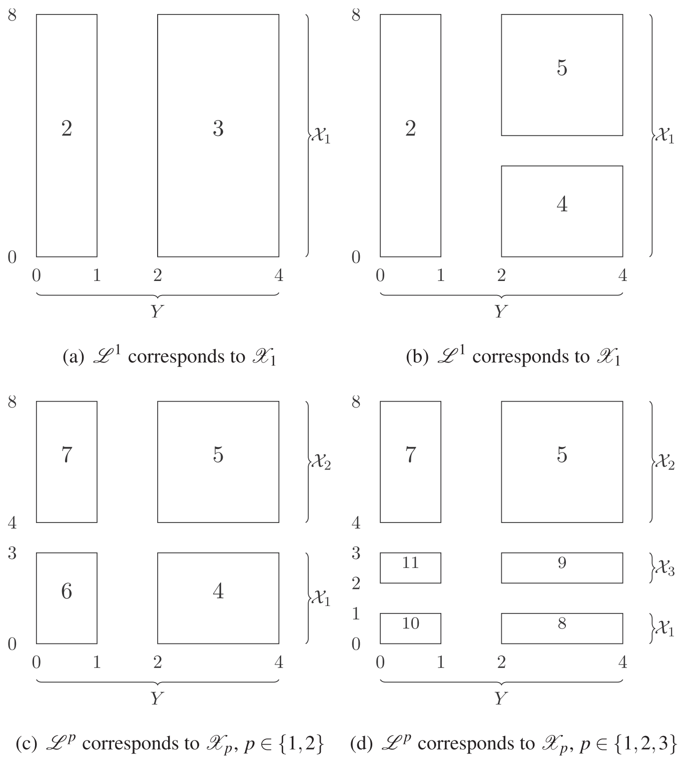

(a) $\mathscr{L}^1$ corresponds to $\mathscr{X}_1$
(b) $\mathscr{L}^1$ corresponds to $\mathscr{X}_1$
(c) $\mathscr{L}^p$ corresponds to $\mathscr{X}_p$, $p \in \{1,2\}$
(d) $\mathscr{L}^p$ corresponds to $\mathscr{X}_p$, $p \in \{1,2,3\}$
Y
$\mathcal{X}_1$
$\mathcal{X}_2$
$\mathcal{X}_3$

Fig. 2. Example of partitions for the host sets in Example 1, i.e., $X = [0, 8] \subset \mathbb{Z}$ and $Y = [0, 4] \subset \mathbb{Z}$. Numbers inside the rectangles denote the node numbers. The numbers at the bottom and to the left of each subfigure denote y and x values, respectively. All (sub)lists are shown in Tables 1 and 2. Briefly, (a) $X = \mathcal{X}_1$: one independent list with one sublist; (b) $X = \mathcal{X}_1$: one independent list with two sublists; (c) $X = \mathcal{X}_1 \cup \mathcal{X}_2$: two independent lists with one sublist each; (d) $X = \mathcal{X}_1 \cup \mathcal{X}_2 \cup \mathcal{X}_3$: three independent lists with one sublist each.

more than one sublist may satisfy (9)–(11), i.e., $|S^p| \geq 1$, and a given node can appear in more than one sublist. For instance, in Fig. 2(b), $\mathcal{X}_1$ has two sublists, $\mathcal{L}_1^1$ and $\mathcal{L}_2^1$, and node 2 belongs to both sublists. This is necessary to ensure that each sublist represents a Y partition.

Finally, let us consider list $\mathcal{L}^p$, $p \in P$, and branch on some of its nodes. If, after the partitioning of nodes, there exist index sets I and J such that:

$$\begin{aligned} & I \cap J = \emptyset, \quad I \cup J = S^p, \\ & \{\mathcal{L}_i^p : i \in I\} \cap \{\mathcal{L}_j^p : j \in J\} = \emptyset, \end{aligned} \tag{IC}$$

list $\mathcal{L}^p$ can be replaced by two (new) independent lists $\mathcal{L}^{p_1}$ and $\mathcal{L}^{p_2}$ corresponding to refined subdomains $\mathcal{X}_{p_1}$ and $\mathcal{X}_{p_2}$, respectively, that form a partition of $\mathcal{X}_p$. The new independent lists are:

$$\mathcal{L}^{p_1} := \{\mathcal{L}_i^p\}, \quad i \in I \subset S, \tag{13}$$

$$\mathcal{L}^{p_2} := \{\mathcal{L}_j^p\}, \quad j \in J \subset S, \tag{14}$$

where $p_1 := p$ and $p_2 := |P| + 1$. We refer to condition (IC) as the *independence condition* because it implies that sets $\{\mathcal{L}_i^p : i \in I\}$ and $\{\mathcal{L}_j^p :$

Table 2
Sublists appearing in each partitioning example shown in Fig. 2. The X subdomain covered by each set $\mathcal{X}_p$, $p \in \{1, 2, 3\}$, varies with each example. The relevant subdomain is marked next to each diagram.

	$\mathcal{X}_1$	$\mathcal{X}_2$	$\mathcal{X}_3$
Fig. 2(a)	$\mathcal{L}_1^1 = \{2, 3\}$	–	–
Fig. 2(b)	$\mathcal{L}_1^1 = \{2, 4\}$, $\mathcal{L}_2^1 = \{2, 5\}$	–	–
Fig. 2(c)	$\mathcal{L}_1^1 = \{4, 6\}$	$\mathcal{L}_1^2 = \{5, 7\}$	–
Fig. 2(d)	$\mathcal{L}_1^1 = \{8, 10\}$	$\mathcal{L}_1^2 = \{5, 7\}$	$\mathcal{L}_1^3 = \{9, 11\}$

$j \in J\}$ cover no overlapping X subdomains, i.e., they have become independent. The dimension of P is increased:

$$|P| := |P| + 1. \tag{15}$$

For example, list $\mathcal{L}^1$ in Fig. 2(b) is replaced by lists $\mathcal{L}^1$ and $\mathcal{L}^2$ in Fig. 2(c), following branching on node 2, Fig. 2(b), in the x dimension.

Remark 5. A node k not yet fully fathomed must belong to an independent list $\mathcal{L}^p$ for some $p \in P$, i.e., there must exist at least one sublist $\mathcal{L}_s^p \in \mathcal{L}^p$ such that $k \in \mathcal{L}_s^p$. For simplicity, we use the shorthand notation $k \in \mathcal{L}^p$. Combining this with Remark 3, we have

$$\exists p \in P : k \in (\mathcal{L} \cup \mathcal{L}_{\text{In}}) \cap \mathcal{L}_p$$

for any node k in the branch-and-bound tree that has not yet been fully fathomed.

To end the overview of our tree management approach, note that every independent list $\mathcal{L}^p$, $p \in P$, is assigned an attribute that we call *best inner upper bound*. Its definition is presented in the next section (cf., Definition 2), where we introduce our bounding scheme at any node k in the branch-and-bound tree.

3.2. Bounding scheme

We present the formulation of the subproblems used in obtaining bounds on the inner and outer problems by considering a node $k \in \mathcal{L}^p$ in the branch-and-bound tree.

Inner bounding scheme. The inner bounding scheme is based on finding valid lower and upper bounds on the inner optimal value function $w(x_i, x_c)$ for all values of the outer variable vector (x_i, x_c). In view of this, the inner lower bounding problem is stated as:

$$\begin{aligned} f^{\mathrm{L},(k)} = \min_{x_i,x_c,y_i,y_c} \quad & f(x_i, x_c, y_i, y_c) \\ \text{s.t.} \quad & g(x_i, x_c, y_i, y_c) \le 0, \\ & x_i \in X_I^{(k)}, x_c \in X_C^{(k)}, y_i \in Y_I^{(k)}, y_c \in Y_C^{(k)}. \end{aligned} \tag{ILB}$$

Problem (ILB) is a classical (nonconvex) mixed-integer nonlinear program (MINLP) with the minimization taking place over the combined outer and inner variable vector (x_i, x_c, y_i, y_c); thus, (ILB) can be solved with available global optimization techniques for general MINLPs, e.g., Adjiman et al. (2000), Tawarmalani and Sahinidis (2004), and Belotti (2009). One may also wish to obtain a valid, but looser, lower bound $\underline{f}^{(k)} \le f^{\mathrm{L},(k)}$ by solving the simpler convex MINLP relaxation:

$$\begin{aligned} \underline{f}^{(k)} = \min_{x_i,x_c,y_i,y_c} \quad & \breve{f}^{(k)}(x_i, x_c, y_i, y_c) \\ \text{s.t.} \quad & \breve{g}^{(k)}(x_i, x_c, y_i, y_c) \le 0, \\ & x_i \in X_I^{(k)}, x_c \in X_C^{(k)}, \\ & y_i \in Y_I^{(k)}, y_c \in Y_C^{(k)}. \end{aligned} \tag{RILB}$$

where $\breve{f}^{(k)}(x_i, x_c, y_i, y_c)$ and $\breve{g}^{(k)}(x_i, x_c, y_i, y_c)$ represent convex underestimators of functions $f(x_i, x_c, y_i, y_c)$ and $g(x_i, x_c, y_i, y_c)$, respectively, over the current domain at node k, with the integrality condition relaxed if necessary. The underestimators can be constructed using standard techniques appropriate to the functional forms in the formulation, e.g., Adjiman et al. (2000), Floudas (2000), and Tawarmalani and Sahinidis (2002).

Next, if k is an active node, i.e., $k \in (\mathcal{L} \cup \mathcal{L}_{\text{In}}) \cap \mathcal{L}^p$, we solve the inner upper bounding problem over the current domain to compute a valid upper bound on node k. To do so, one has to account for all values of the outer variables and take the worst-case scenario with respect to those. Thus, the inner upper bounding problem is given via the so-called *robust counterpart* of problem (3), which is parametric in (x_i, x_c):

$$\begin{aligned} f^{\mathrm{U},(k)} = \min_{y_i,y_c,t} \quad & t \\ \text{s.t.} \quad & f(x_i, x_c, y_i, y_c) \le t \quad \forall (x_i, x_c) \in X_I^{(k)} \times X_C^{(k)}, \\ & g(x_i, x_c, y_i, y_c) \le 0 \quad \forall (x_i, x_c) \in X_I^{(k)} \times X_C^{(k)}, \\ & y_i \in Y_I^{(k)}, \quad y_c \in Y_C^{(k)}. \end{aligned} \tag{IUB}$$

Problem (IUB) is a standard mixed-integer semi-infinite problem that can be tackled by overestimating the functions in the semi-infinite constraints. This can be done by using interval arithmetic extended to mixed-integer problems (Apt and Zoeteweij, 2007; Berger and Granvilliers, 2009) in the framework of the upper bounding procedure of Bhattacharjee et al. (2005), namely:

$$\begin{aligned} \bar{f}^{(k)} = \min_{y_i,y_c,t} \quad & t \\ \text{s.t.} \quad & \bar{f}(X_I^{(k)}, X_C^{(k)}, y_i, y_c) \le t, \\ & \bar{g}(X_I^{(k)}, X_C^{(k)}, y_i, y_c) \le 0, \\ & y_i \in Y_I^{(k)}, y_c \in Y_C^{(k)}, \end{aligned} \tag{RIUB}$$

where

$$\bar{f}(X_I^{(k)}, X_C^{(k)}, y_i, y_c) \ge f(x_i, x_c, y_i, y_c) \quad \forall (x_i, x_c) \in X_I^{(k)} \times X_C^{(k)};$$
$$\bar{g}(X_I^{(k)}, X_C^{(k)}, y_i, y_c) \ge g(x_i, x_c, y_i, y_c) \quad \forall (x_i, x_c) \in X_I^{(k)} \times X_C^{(k)}.$$

Remark 6. Observe that in the original Branch-and-Sandwich algorithm (Kleniati and Adjiman, 2014a), a max–min formulation was adopted to formulate the auxiliary inner upper bounding problem. In the present article, such a formulation is not desirable because it would lead to a challenging generalized semi-infinite program in which the (infinite) index set would be dependent on the mixed-integer outer variable vector (x_i, x_c). Furthermore, to the best of our knowledge, there is no algorithm to solve such problems. On the other hand, the worst-case scenario formulation (IUB) employed here for the mixed-integer case can also be applied to the continuous case, thereby lifting the assumption of regularity.

Remark 7. Note that the inner lower bound is used for the selection of a candidate node (cf., Section 3.2.2) and the inner upper bound is used to construct the outer lower bounding problem as in problem (LB) introduced later in this Section. Both bounds are used for fathoming when their values indicate that the current node does not contain the global solution of the inner problem (cf., Section 3.2.1).

Having computed the inner upper bounds for specific nodes, we update the best inner upper bound of list $\mathcal{L}^p$, $f^{\mathrm{UB},p}$, if necessary, based on the definition below.

Definition 2 *(Best inner upper bound).* The best inner upper bound is identified in a manner consistent with (IUB), i.e., it is the lowest value over the y variables but the largest value over the x variables. In particular, recalling (7), each list $\mathcal{L}^p$, $p \in P$, has:

$$f^{\mathrm{UB},p} = \max\{\min_{j \in \mathcal{L}_1^p}\{\bar{f}^{(j)}\}, \ldots, \min_{j \in \mathcal{L}_{|S|}^p}\{\bar{f}^{(j)}\}\}. \tag{16}$$

Outer bounding scheme. If k is a node in list L, i.e., if $k \in \mathcal{L} \cap \mathcal{L}^p$, the outer lower bounding problem is employed:

$$\begin{aligned} \underline{F}^{(k)} = \min_{x_i, x_c, y_i, y_c} \quad & F(x_i, x_c, y_i, y_c) \\ \text{s.t.} \quad & G(x_i, x_c, y_i, y_c) \leq 0, \\ & g(x_i, x_c, y_i, y_c) \leq 0, \\ & f(x_i, x_c, y_i, y_c) \leq f^{\text{UB},p}, \\ & x_i \in X_I^{(k)}, \quad x_c \in X_C^{(k)}, \\ & y_i \in Y_I^{(k)}, \quad y_c \in Y_C^{(k)}. \end{aligned} \tag{LB}$$

To tighten problem (LB), we can add the inner KKT conditions with respect to the continuous inner variables when regularity of the inner problem is satisfied for all the x_c and x_i values (e.g., see Mitsos, 2010; Kleniati and Adjiman, 2014a). Any feasible solution in (1) is feasible in the proposed relaxation (LB).

Let the solution of the outer lower bounding problem (LB) be $(\bar{x}_i, \bar{x}_c)$. Then, for the outer upper bounding procedure and for $k \in \mathcal{L} \cap \mathcal{L}^p$, we first fix (x_i, x_c) to $(\bar{x}_i, \bar{x}_c)$ and look for a node $k' \in (\mathcal{L} \cup \mathcal{L}_{\text{In}}) \cap \mathcal{L}^p$ such that:

$$k' := \underset{j \in (\mathcal{L} \cup \mathcal{L}_{\text{In}}) \cap \mathcal{L}^p}{\arg\min} \; w^{(j)}(\bar{x}_i, \bar{x}_c), \tag{MinISP}$$

where

$$w^{(j)}(\bar{x}_i, \bar{x}_c) := \min_{y_i, y_c} \{ f(\bar{x}_i, \bar{x}_c, y_i, y_c) \quad \text{s.t.} \quad g(\bar{x}_i, \bar{x}_c, y_i, y_c) \leq 0, \quad y_i \in Y_I^{(j)}, y_c \in Y_C^{(j)} \}. \tag{ISP}$$

Remark 8. The outer lower bounding problem (LB) contains only one cut based on the tightest constant upper bound for the inner objective value. This bound is valid over the whole Y for the domain $X_I^{(k)} \times X_C^{(k)}$ considered.

Finally, for $(x_i, x_c) = (\bar{x}_i, \bar{x}_c)$, we solve the following outer upper bounding problem over node k' (Mitsos et al., 2008; Mitsos, 2010; Kleniati and Adjiman, 2014a):

$$\begin{aligned} \overline{F}^{(k\prime)} = \min_{y_i, y_c} \quad & F(\bar{x}_i, \bar{x}_c, y_i, y_c) \\ \text{s.t.} \quad & G(\bar{x}_i, \bar{x}_c, y_i, y_c) \leq 0, \\ & g(\bar{x}_i, \bar{x}_c, y_i, y_c) \leq 0, \\ & f(\bar{x}_i, \bar{x}_c, y_i, y_c) \leq w^{(k')}(\bar{x}_i, \bar{x}_c) + \varepsilon_f, \\ & y_i \in Y_I^{(k')}, \quad y_c \in Y_C^{(k')}. \end{aligned} \tag{UB}$$

where ε_f is a given inner objective tolerance. Any feasible solution in the upper bounding problem (UB) is feasible in the original bilevel problem (1). Finally, we use $\overline{F}^{(k')}$ to update the best upper bound F^{UB} whenever $\overline{F}^{(k')} < F^{\text{UB}}$, yielding a new *incumbent objective value*.

3.2.1. Node fathoming rules

The fathoming rules, stated below for completeness, are the same as those introduced in the original Branch-and-Sandwich algorithm (Kleniati and Adjiman, 2014a,b).

Definition 3 *(Inner fathoming rules)*. Consider a node $k \in \mathcal{L}^p$. If

1. $\underline{f}^{(k)} = \infty$ or
2. $\underline{f}^{(k)} > f^{\text{UB},p}$

then fully fathom k, i.e., delete it from $\mathcal{L}$ (or $\mathcal{L}_{\text{In}}$) and $\mathcal{L}^p$.

Definition 4 *(Outer fathoming rules)*. Given outer objective tolerance ε_F, consider a node $k \in \mathcal{L} \cap \mathcal{L}^p$. If

1. $\underline{F}^{(k)} = \infty$ or
2. $\underline{F}^{(k)} \geq F^{\text{UB}} - \varepsilon_F$

then *outer fathom* k, i.e., move it from $\mathcal{L}$ to $\mathcal{L}_{\text{In}}$. Hence, after outer fathoming, $k \in \mathcal{L}_{\text{In}} \cap \mathcal{L}^p$.

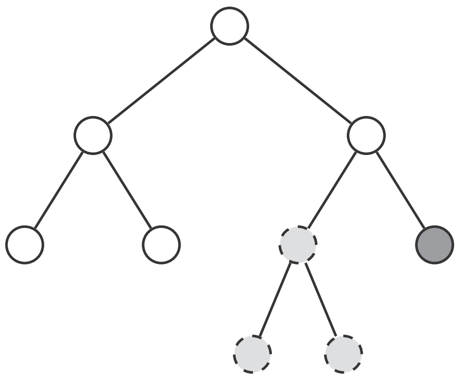


Fig. 3. Example of a branch-and-bound tree in the Branch-and-Sandwich algorithm. The clear nodes are open nodes in list $\mathcal{L}$, the light gray nodes are outer-fathomed nodes in list $\mathcal{L}_{\text{In}}$ and the dark gray node is fully fathomed.

Moreover, if a sublist contains outer-fathomed nodes only, i.e., it no longer contains any nodes in $\mathcal{L}$ which are open from the perspective of the overall problem, then it can be deleted. This may lead to full fathoming of the corresponding nodes as long as they do not appear in other sublists of the same independent list. The rules are summarized below.

Definition 5 *(List-deletion fathoming rules)*. Consider a sublist $\mathcal{L}_i^p \in \mathcal{L}^p$, $i \in S^p$.

1. If $\mathcal{L}_i^p \cap \mathcal{L} = \emptyset$ and $\mathcal{L}_i^p \cap \mathcal{L}_j^p = \emptyset$ for all $j \neq i \in S^p$, then fully fathom all nodes $k \in \mathcal{L}_i^p$, i.e., delete them from $\mathcal{L}_{\text{In}}$ and $\mathcal{L}^p$. Delete also sublist $\mathcal{L}_i^p$ and decrease $|S^p|$.
2. If $\mathcal{L}_i^p \cap \mathcal{L} = \emptyset$ and $\mathcal{L}_i^p \cap \mathcal{L}_j^p \neq \emptyset$ for some $j \neq i \in S^p$, then delete sublist $\mathcal{L}_i^p$ and decrease $|S^p|$.
3. If $|S^p| = 0$, delete list $\mathcal{L}^p$ and decrease $|P|$.

As a result of our fathoming rules, the branch-and-bound tree in the Branch-and-Sandwich algorithm contains three types of nodes:

1. *Open nodes:* those in $\mathcal{L} \cap \mathcal{L}^p$ for some $p \in P$. We continue exploration of these nodes with respect to both the inner and outer problems. Open nodes are shown as clear nodes in the tree of Fig. 3.
2. *Outer-fathomed nodes:* those in $\mathcal{L}_{\text{In}} \cap \mathcal{L}^p$ for some $p \in P$. We continue exploration of these nodes with respect to the inner problem only. Outer-fathomed (or inner-open) nodes are those with a dashed line and light-gray color in Fig. 3.
3. *Fathomed nodes:* deleted from all the lists. No further exploration of these nodes is required. The dark-gray nodes of Fig. 3 are (fully) fathomed nodes.

Remark 9. Recall that nodes in $(\mathcal{L} \cup \mathcal{L}_{\text{In}})$ are all active nodes, i.e., nodes at which we continue branching (see also Section 3.1). However, at nodes in $\mathcal{L}_{\text{In}}$ we no longer need to apply our outer bounding scheme, i.e., we only compute inner bounds.

Remark 10. Any active node, either in $\mathcal{L}$ or in $\mathcal{L}_{\text{In}}$, can be fully fathomed when one of the inner fathoming rules applies. Any active node in $\mathcal{L}$ may first be outer-fathomed (i.e., moved to $\mathcal{L}_{\text{In}}$) and then be fully fathomed at a later iteration.

3.2.2. Selection operation

The selection operation is that introduced in the original Branch-and-Sandwich algorithm (Kleniati and Adjiman, 2014a,b) and it is stated here for completeness. The selection operation of the algorithm includes three steps: (i) a classical selection rule, e.g., lowest (outer) lower bound, is first applied to the open nodes in list $\mathcal{L}$; (ii) then, the list $\mathcal{L}^p$ corresponding to this node is identified; (iii) finally, the node in that list, i.e., $k \in \mathcal{L} \cap \mathcal{L}^p$, with the lowest level is chosen; if several nodes are at the same level in the tree, the node corresponding to the lowest inner lower bound is chosen:

$$k := \arg\min_i\{\underline{f}^{(i)} \mid i := \arg\min_{j\in\mathcal{L}^p}\{l^{(j)}\}\}. \tag{ISR}$$

Furthermore, we branch on outer-fathomed nodes too, i.e., nodes that belong to list $\mathcal{L}_{\text{In}}$. In particular, for the list $\mathcal{L}^p$ already identified, we check whether $\mathcal{L}_{\text{In}} \cap \mathcal{L}^p \neq \emptyset$, in which case a node $k_{\text{In}} \in \mathcal{L}_{\text{In}} \cap \mathcal{L}^p$ is selected using (ISR). This node is explored further with respect to the inner problem only, since nodes in $\mathcal{L}_{\text{In}}$ are kept to provide information about the inner global optimal solutions. Branching on an outer-fathomed node can take place based on outer or inner variables. To recapitulate, the selection operation of the proposed algorithm is stated below.

Definition 6 *(Selection operation of the Branch-and-Sandwich algorithm).* The selection rule is:

(i) find a node in $\mathcal{L}$ with lowest overall lower bound: $k^{\text{LB}} = \arg\min_{j\in\mathcal{L}}\{\underline{F}^{(j)}\}$;
(ii) find the corresponding $\mathcal{X}_p$ subdomain, $p \in P$, such that $k^{\text{LB}} \in \mathcal{L}^p$;
(iii) select a node $k \in \mathcal{L} \cap \mathcal{L}^p$ and a node $k_{\text{In}} \in \mathcal{L}_{\text{In}} \cap \mathcal{L}^p$, if non empty, using (ISR).

4. Algorithm statement and convergence properties

A statement of the algorithm is given in this section. This is followed by an illustration of the algorithm on Example 1 and by a proof of convergence.

4.1. Algorithm statement

Given outer and inner objective tolerances ε_F, ε_f, respectively, and a feasibility tolerance ε, the Branch-and-Sandwich algorithm for mixed-integer nonlinear bilevel problems is stated here, and summarized in Fig. 4.

Algorithm 1. Branch-and-Sandwich

Step 0: Initialization Initialize lists: $\mathcal{L} := \mathcal{L}_{\text{In}} := \emptyset$. Set the incumbent $(x^{\text{UB}}, y^{\text{UB}}) := \emptyset$ and $F^{\text{UB}} := \infty$, the iteration counter: Iter := 0, and the node counter: $n_{\text{node}} := 1$ corresponding to the whole domain $X \times Y$.

Step 1: Inner and outer bounds for root node

Step 1.1: Solve the relaxed inner problem (RILB) to compute $\underline{f}^{(1)}$. If infeasible go to Step 2.

Step 1.2: Solve the restricted inner problem (RIUB).

Step 1.3: Solve the lower bounding problem (LB) globally to obtain $\underline{F}^{(1)}$. If infeasible, go to Step 2. Otherwise, if a feasible solution $(x^{(1)}, y^{(1)})$ is computed, add node to the universal list $\mathcal{L} := \{1\}$ with properties $(\underline{f}^{(1)}, \overline{f}^{(1)}, \underline{F}^{(1)}, x^{(1)}, l^{(1)})$, where $l^{(1)} := 0$. Initialize the partition of X, i.e., $p := 1$ and $\mathcal{X}_1 := X$, and generate the first independent list $\mathcal{L}^1 := \{1\}$. Set the best inner upper bound for $\mathcal{X}_1$: $f^{\text{UB},1} := \overline{f}^{(1)}$.

Step 1.4: Set $\bar{x} := x^{(1)}$ and compute $w(\bar{x})$ using (3). Then, solve (UB) locally to obtain $\overline{F}^{(1)}$. If a feasible solution (x_f, y_f) is obtained, update the incumbent $(x^{\text{UB}}, y^{\text{UB}}) = (x_f, y_f)$ and $F^{\text{UB}} = \overline{F}^{(1)}$.

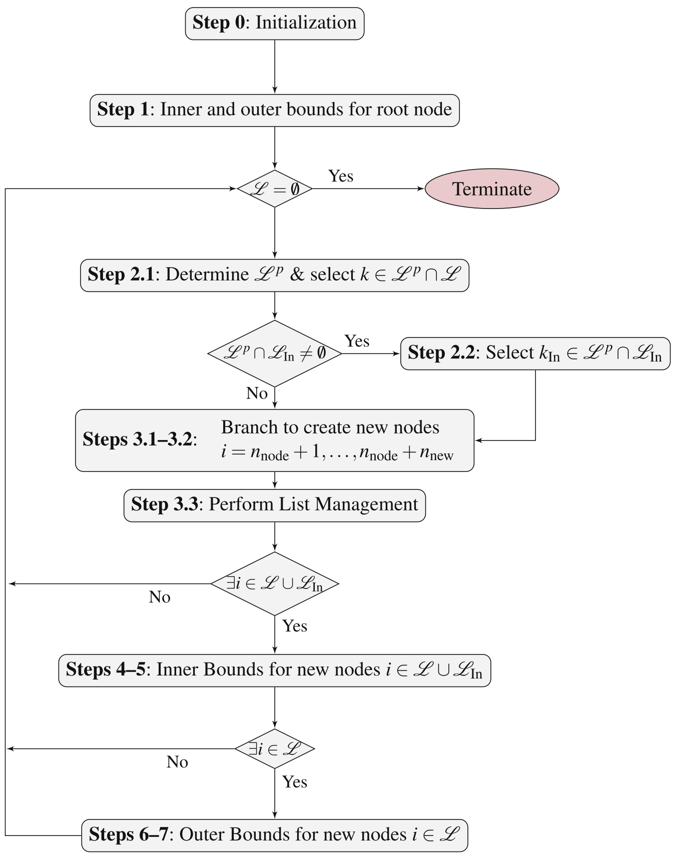

Step 0: Initialization
Step 1: Inner and outer bounds for root node
$\mathcal{L} = \emptyset$
Yes
Terminate
Step 2.1: Determine $\mathcal{L}^p$ & select $k \in \mathcal{L}^p \cap \mathcal{L}$
$\mathcal{L}^p \cap \mathcal{L}_{\text{In}} \neq \emptyset$
Yes
Step 2.2: Select $k_{\text{In}} \in \mathcal{L}^p \cap \mathcal{L}_{\text{In}}$
No
Steps 3.1–3.2: Branch to create new nodes $i = n_{\text{node}}+1, \ldots, n_{\text{node}}+n_{\text{new}}$
Step 3.3: Perform List Management
$\exists i \in \mathcal{L} \cup \mathcal{L}_{\text{In}}$
No
Yes
Steps 4–5: Inner Bounds for new nodes $i \in \mathcal{L} \cup \mathcal{L}_{\text{In}}$
$\exists i \in \mathcal{L}$
No
Yes
Steps 6–7: Outer Bounds for new nodes $i \in \mathcal{L}$

Fig. 4. An outline flowchart for the Branch-and-Sandwich algorithm.

Step 2: Node(s) selection If $\mathcal{L} = \emptyset$, terminate and report the incumbent solution and value. Otherwise increase the iteration counter, Iter = Iter + 1, and:

Step 2.1: Select a best candidate list $\mathcal{L}^p$ and a best candidate node $k \in \mathcal{L}^p \cap \mathcal{L}$; remove k from $\mathcal{L}$.

Step 2.2: If $\mathcal{L}^p \cap \mathcal{L}_{\text{In}} \neq \emptyset$, select a best candidate node $k_{\text{In}} \in \mathcal{L}^p \cap \mathcal{L}_{\text{In}}$; remove k_{In} from $\mathcal{L}_{\text{In}}$.

Step 3: Branching

Step 3.1: Branch on x or y variables at node k to create two new nodes, i.e., $n_{\text{node}}+1$ and $n_{\text{node}}+2$. Set $n_{\text{new}} := 2$ and initialize node properties.

Step 3.2: If a node k_{In} is selected, branch at k_{In} to create two new (outer-fathomed) nodes, i.e., $n_{\text{node}}+3$ and $n_{\text{node}}+4$. Set $n_{\text{new}} := 4$ and initialize node properties.

Step 3.3: List management: For $i = n_{\text{node}}+1, \ldots, n_{\text{node}}+n_{\text{new}}$, find the corresponding subdomain $\mathcal{X}_{p_i}$ such that $i \in \mathcal{L}^{p_i}$ and set/update f^{UB,p_i}. Apply the inner-value-dominance fathoming rule (cf., Definition 3).

Step 4: Inner lower bound If there is no $i \in \{n_{\text{node}}+1, \ldots, n_{\text{node}}+n_{\text{new}}\}$, such that $i \in \mathcal{L} \cup \mathcal{L}_{\text{In}}$, apply the list deletion rules (cf., Definition 5) and go to Step 2. Otherwise, for $i \in \{n_{\text{node}}+1, \ldots, n_{\text{node}}+n_{\text{new}}\} \cap (\mathcal{L} \cup \mathcal{L}_{\text{In}})$, solve the auxiliary relaxed inner problem (RILB) to compute $\underline{f}^{(i)}$. If feasible and $\underline{f}^{(i)} \le f^{\text{UB},p_i}$, then:

- if $i \in \{n_{\text{node}}+1, n_{\text{node}}+2\}$, add node i to the list $\mathcal{L}$ with properties $(\underline{f}^{(i)}, \overline{f}^{(i)}, \underline{F}^{(i)}, x^{(i)}, l^{(i)})$;

- else if $n_{\text{new}} = 4$ and $i \in \{n_{\text{node}}+3, n_{\text{node}}+4\}$, add node i to the list $\mathcal{L}_{\text{In}}$ with properties $(\underline{f}^{(i)}, \bar{f}^{(i)}, l^{(i)})$.

Otherwise, remove i from $\mathcal{L}^{p_i}$. Apply the list deletion rules (cf., Definition 5).

Step 5: Inner upper bound If there is no $i \in \{n_{\text{node}}+1, \ldots, n_{\text{node}}+n_{\text{new}}\}$, such that $i \in \mathcal{L} \cup \mathcal{L}_{\text{In}}$, go to Step 2. Otherwise, for $i \in \{n_{\text{node}}+1, \ldots, n_{\text{node}}+n_{\text{new}}\} \cap (\mathcal{L} \cup \mathcal{L}_{\text{In}})$, solve the auxiliary restricted inner problem (RIUB) to compute $\bar{f}^{(i)}$. Update $\bar{f}^{(i)}$ in $\mathcal{L}$ or in $\mathcal{L}_{\text{In}}$ and then update f^{UB,p_i} using (16) (cf., Definition 2). Apply the inner-value-dominance fathoming rule (cf., Definition 3) and, if necessary, the list deletion fathoming rules (cf., Definition 5).

Step 6: Outer lower bound If there is no $i \in \{n_{\text{node}}+1, n_{\text{node}}+2\}$, such that $i \in \mathcal{L}$, go to Step 2. Otherwise, for $i \in \{n_{\text{node}}+1, n_{\text{node}}+2\} \cap \mathcal{L}$, solve the lower bounding problem (LB) globally to obtain $\underline{F}^{(i)}$. If a feasible solution $(x^{(i)}, y^{(i)})$ is obtained with $\underline{F}^{(i)} \le F^{\text{UB}} + \varepsilon_F$, update $\underline{F}^{(i)}$ and $x^{(i)}$ in $\mathcal{L}$. If $\underline{F}^{(i)} \ge F^{\text{UB}} - \epsilon_F$, move i from $\mathcal{L}$ to the list $\mathcal{L}_{\text{In}}$ with properties $(\underline{f}^{(i)}, \bar{f}^{(i)}, l^{(i)})$ and apply the list deletion rules (cf., Definition 5).

Step 7: Outer upper bound If there is no $i \in \{n_{\text{node}}+1, n_{\text{node}}+2\}$, such that $i \in \mathcal{L}$, go to Step 2. Otherwise, for $i \in \{n_{\text{node}}+1, n_{\text{node}}+2\} \cap \mathcal{L}$, do:

Step 7.1: Set $\bar{x} := x^{(i)}$ and, using (ISP), compute $w^{(j)}(\bar{x})$ for all $j \in \mathcal{L}^{p_i}$, such that $\bar{x} \in X^{(j)}$. Set i' based on (Min-ISP).

Step 7.2: Solve (UB) (locally) to obtain $\bar{F}^{(i')}$. If a feasible solution $(x_f^{(i')}, y_f^{(i')})$ is obtained with $\bar{F}^{(i')} < F^{\text{UB}}$ update the incumbent:

$(x^{\text{UB}}, y^{\text{UB}}) = (x_f^{(i')}, y_f^{(i')})$ and $F^{\text{UB}} = \bar{F}^{(i')}$.

Move from the list $\mathcal{L}$ to the list $\mathcal{L}_{\text{In}}$ all nodes j such that $\underline{F}^{(j)} \ge F^{\text{UB}} - \varepsilon_F$ and apply the list deletion fathoming rules (cf., Definition 5). Increase the node counter, i.e., $n_{\text{node}} = n_{\text{node}} + n_{\text{new}}$, and go to Step 2.

Remark 11. In Step 3.1, the node properties are initialized as follows:

$$\underline{f}^{(n_{\text{node}}+1)} := \underline{f}^{(n_{\text{node}}+2)} := \underline{f}^{(k)};$$
$$\bar{f}^{(n_{\text{node}}+1)} := \bar{f}^{(n_{\text{node}}+2)} := \bar{f}^{(k)};$$
$$\underline{F}^{(n_{\text{node}}+1)} := \underline{F}^{(n_{\text{node}}+2)} := \underline{F}^{(k)};$$
$$x^{(n_{\text{node}}+1)} := x^{(n_{\text{node}}+2)} := x^{(k)};$$
$$l^{(n_{\text{node}}+1)} := l^{(n_{\text{node}}+2)} := l^{(k)} + 1.$$

where $x^{(k)}$ is inherited only if it is in the domain of the child node.

Similarly, in Step 3.2, the node properties are initialized as:

$$\underline{f}^{(n_{\text{node}}+3)} := \underline{f}^{(n_{\text{node}}+4)} := \underline{f}^{(k_{\text{In}})};$$
$$\bar{f}^{(n_{\text{node}}+3)} := \bar{f}^{(n_{\text{node}}+4)} := \bar{f}^{(k_{\text{In}})};$$
$$l^{(n_{\text{node}}+3)} := l^{(n_{\text{node}}+4)} := l^{(k_{\text{In}})} + 1.$$

Remark 12. In Steps 4–5, we start by testing whether the new nodes for inner and outer exploration, created in Step 3, still exist following the application of node fathoming in Step 3.3 and in Step 4. As further node fathoming may take place in Step 5 (resp. Step 6), we start Step 6 (resp. Step 7) by testing whether there still exist any new nodes for outer exploration.

Remark 13. In Step 7 it is not always necessary to compute $w^{(j)}(\bar{x})$ for each node j. If $w^{(j)}(x)$ has previously been calculated for $x = \bar{x}$ over the same Y subdomain, this information can be re-used.

4.2. Illustrative example revisited

Returning to Example 1, we present the branch-and-bound tree for its solution in Fig. 5 and provide in this section a step-by-step illustration of the progress of the algorithm in solving this problem. The inner upper bounding formulations (IUB) and (RIUB) are presented in Appendix A.

Step 0: Iter = 0. Set $\mathcal{L} = \mathcal{L}_{\text{In}} = \emptyset$, $(x^{\text{UB}}, y^{\text{UB}}) = \emptyset$, $F^{\text{UB}} = \infty$, $n_{\text{node}} = 1$.

Step 1: **Step 1.1:** Compute the inner lower bound for node 1, $\underline{f}^{(1)} = 1$.

Step 1.2: Compute the inner upper bound for node 1, $\bar{f}^{(1)} = 4$.

Step 1.3: The outer lower bounding problem yields $\underline{F}^{(1)} = -42$, with solution $(x^{(1)}, y^{(1)}) = (2, 4)$. Since the outer lower bounding problem is feasible, add node 1 to the list $\mathcal{L}$ with all the computed information:

$\mathcal{L} = \{1: \ 1 \ 4 \ -42 \ 2 \ 0\}$,

where the last field is the level of node 1: $l^{(1)} = 0$. Also, set $p = 1$ and

$\mathcal{X}_1 = [0, 8] \cap \mathbb{Z}$, $\mathcal{L}^1 = \{1\}$, $f^{\text{UB},1} = \bar{f}^{(1)} = 4$.

Step 1.4: Set $\bar{x} := x^{(1)} = 2$ and compute $w(\bar{x}) = 2$. Then, obtain $\bar{F}^{(1)} = -22$, with solution (2, 2). Update the incumbent:

$(x^{\text{UB}}, y^{\text{UB}}) = (2, 2)$ and $F^{\text{UB}} = -22$.

Step 2: Iter = 1 (since $\mathcal{L} \neq \emptyset$). Select node $1 \in \mathcal{L} \cap \mathcal{L}^1$ and remove it from $\mathcal{L}$. As a result, at this point $\mathcal{L} = \emptyset$ and $\mathcal{L}^1 = \{1\}$.

Step 3: **Step 3.1:** Branch on $y = 1$ and create nodes:

$2 := \{(x, y) \in \mathbb{Z}^2 \mid 0 \le x \le 8, \ 0 \le y \le 1\}$,
$3 := \{(x, y) \in \mathbb{Z}^2 \mid 0 \le x \le 8, \ 2 \le y \le 4\}$.

Set $n_{\text{new}} = 2$.

Step 3.2: $\mathcal{L}_{\text{In}} = \emptyset$ so this step does not apply.

Step 3.3: Set up a single independent list, $\mathcal{L}^1 = \{2, 3\}$ corresponding to $\mathcal{X}_1 = [0, 8] \cap \mathbb{Z}$. There is one sublist; set $f^{\text{UB},1} = 4$.

Step 4: Compute $\underline{f}^{(2)} = 1$ and $\underline{f}^{(3)} = 2$ and add both nodes to $\mathcal{L}$:

$$\mathcal{L} = \left\{ \begin{array}{llllll} 2: & 1 & 4 & -42 & 2 & 1 \\ 3: & 2 & 4 & -42 & 2 & 1 \end{array} \right\}.$$

The first and last properties, i.e., $\underline{f}^{(i)}$ and $l^{(i)}$, $i = 1, 2$, where $l^{(2)} = l^{(3)} = 1$, are set based on the new nodes 2 and 3; the other values are inherited from node 1.

Step 5: Compute $\bar{f}^{(2)} = 4$ and $\bar{f}^{(3)} = 4$. No updates of $\mathcal{L}$ or $f^{\text{UB},1}$ are needed.

Step 6: Compute $\underline{F}^{(2)} = -18$, with solution (8,1) and $\underline{F}^{(3)} = -42$ with solution (2,2). $\underline{F}^{(2)} > F^{\text{UB}} - \varepsilon_F$, so move node 2 from $\mathcal{L}$ to $\mathcal{L}_{\text{In}}$ (node 2 is outer-fathomed). $\underline{F}^{(3)} \le F^{\text{UB}} - \varepsilon_F$ and information for node 3 in $\mathcal{L}$ is unchanged.

Step 7: For $i = 3$, we set $\bar{x} = x^{(3)} = 2$ and compute $\underline{w}^{(2)}(\bar{x}) = \infty$ and $\underline{w}^{(3)}(\bar{x}) = 2$. The latter value is the lowest, so we compute $\bar{F}^{(3)} = -22$ at (2, 2), but no update of the incumbent is required. Set $n_{\text{node}} = 3$ and go to Step 2. At this point, the two lists of active nodes are as follows:

$$\begin{array}{ll} \mathcal{L} = & \{ \ 3: \ 2 \ \ 4 \ \ -42 \ \ 2 \ \ 1 \ \}; \\ \mathcal{L}_{\text{In}} = & \{ \ 2: \ 1 \ \ 4 \ \ 1 \ \}. \end{array}$$

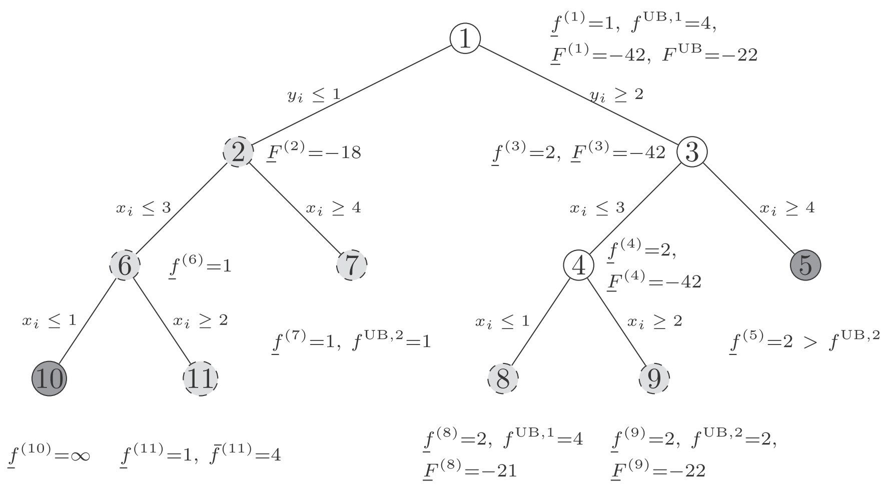

$\underline{f}^{(1)}=1,\ f^{\mathrm{UB},1}=4,$
$\underline{F}^{(1)}=-42,\ F^{\mathrm{UB}}=-22$
$y_i \le 1$
$y_i \ge 2$
$\underline{F}^{(2)}=-18$
$\underline{f}^{(3)}=2,\ \underline{F}^{(3)}=-42$
$x_i \le 3$
$x_i \ge 4$
$x_i \le 3$
$x_i \ge 4$
$\underline{f}^{(6)}=1$
$\underline{f}^{(4)}=2,$
$\underline{F}^{(4)}=-42$
$x_i \le 1$
$x_i \ge 2$
$\underline{f}^{(7)}=1,\ f^{\mathrm{UB},2}=1$
$x_i \le 1$
$x_i \ge 2$
$\underline{f}^{(5)}=2\ >\ f^{\mathrm{UB},2}$
$\underline{f}^{(10)}=\infty$
$\underline{f}^{(11)}=1,\ \bar{f}^{(11)}=4$
$\underline{f}^{(8)}=2,\ f^{\mathrm{UB},1}=4$
$\underline{F}^{(8)}=-21$
$\underline{f}^{(9)}=2,\ f^{\mathrm{UB},2}=2,$
$\underline{F}^{(9)}=-22$

Fig. 5. Branch-and-bound solution tree for Example 1.

Step 2: **Iter** = 2. Apply the selection rule of Definition 6 and choose node $3 \in \mathcal{L}$, which then points to the domain $\mathcal{X}_1$ via list $\mathcal{L}^1$, since $3 \in \mathcal{L} \cap \mathcal{L}^1$. As the only node in $\mathcal{L} \cap \mathcal{L}^1$, node 3 is selected and then removed from $\mathcal{L}$. As a result, at this point, $\mathcal{L} = \emptyset$ and $\mathcal{L}^1 = \{2, 3\}$. Furthermore, since node $2 \in \mathcal{L}_{\mathrm{In}}$, node 2 is selected from the list of outer-fathomed nodes, resulting in $\mathcal{L}_{\mathrm{In}} = \emptyset$.

Step 3: **Step 3.1:** At node 3, branch on $x = 3$, creating nodes 4 and 5:

$$4 := \{(x, y) \in \mathbb{Z}^2 \mid 0 \le x \le 3,\quad 2 \le y \le 4\},$$
$$5 := \{(x, y) \in \mathbb{Z}^2 \mid 4 \le x \le 8,\quad 2 \le y \le 4\}.$$

Set $n_{\mathrm{new}} = 2$.

Step 3.2: At node 2, branch on $x = 3$, creating nodes 6 and 7:

$$6 := \{(x, y) \in \mathbb{Z}^2 \mid 0 \le x \le 3,\quad 0 \le y \le 1\},$$
$$7 := \{(x, y) \in \mathbb{Z}^2 \mid 4 \le x \le 8,\quad 0 \le y \le 1\}.$$

Set $n_{\mathrm{new}} = 4$.

Step 3.3: Set up two independent lists: $\mathcal{L}^1 = \{4, 6\}$, corresponding to $\mathcal{X}_1 = [0, 3] \cap \mathbb{Z}$, and $\mathcal{L}^2 = \{5, 7\}$, corresponding to $\mathcal{X}_2 = [4, 8] \cap \mathbb{Z}$. There are two sublists (one per independent list). Set $f^{\mathrm{UB},1} = f^{\mathrm{UB},2} = 4$.

Step 4: Compute $\underline{f}^{(4)} = 2, \underline{f}^{(5)} = 2, \underline{f}^{(6)} = 1, \underline{f}^{(7)} = 1$. Nodes 4 and 5 are added to $\mathcal{L}$:

$$\mathcal{L} = \left\{ \begin{array}{cccccc} 4: & 2 & 4 & -42 & 2 & 2 \\ 5: & 2 & 4 & -42 & - & 2 \end{array} \right\}.$$

Nodes 6 and 7 are added to $\mathcal{L}_{\mathrm{In}}$:

$$\mathcal{L}_{\mathrm{In}} = \left\{ \begin{array}{cccc} 6: & 1 & 4 & 2 \\ 7: & 1 & 4 & 2 \end{array} \right\}.$$

Step 5: Compute $\bar{f}^{(4)} = 4$, $\bar{f}^{(5)} = 3$, $\bar{f}^{(6)} = 4$ and $\bar{f}^{(7)} = 1$. Update $f^{\mathrm{UB},1} = \min\{\bar{f}^{(4)}, \bar{f}^{(6)}\} = 4$ and $f^{\mathrm{UB},2} = \min\{\bar{f}^{(5)}, \bar{f}^{(7)}\} = 1$. Apply the inner-value-dominance fathoming rule and find that $\underline{f}^{(5)} > f^{\mathrm{UB},2}$. Node 5 can be fully fathomed: it is removed from list $\mathcal{L}$ and list $\mathcal{L}^2$. Since $\mathcal{L}^2 = \{7\}$ and $7 \in \mathcal{L}_{\mathrm{In}}$, node 7 can be fully fathomed. It is removed from $\mathcal{L}_{\mathrm{In}}$ and list $\mathcal{L}^2$ is deleted. At this point, $\mathcal{L} = \{4\}$, $\mathcal{L}_{\mathrm{In}} = \{6\}$ and $\mathcal{L}^1 = \{4, 6\}$.

Step 6: Compute $\underline{F}^{(4)} = -42$.

Step 7: For node 4, we set $\bar{x} = x^{(4)} = 2$ and compute $\underline{w}^{(4)}(\bar{x}) = 2$ and $\underline{w}^{(6)}(\bar{x}) = \infty$. The former value is the lowest, so we compute $\bar{F}^{(4)} = -22$ and no update of the incumbent is needed. Set $n_{\mathrm{node}} = 7$.

Step 2: **Iter** = 3. Node 4 is selected and removed from $\mathcal{L}$. Further, node 6 is selected and removed from $\mathcal{L}_{\mathrm{In}}$. Both lists $\mathcal{L}$ and $\mathcal{L}_{\mathrm{In}}$ are now empty.

Step 3: **Step 3.1:** At node 4, branch on $x = 1$, creating nodes 8 and 9:

$$8 := \{(x, y) \in \mathbb{Z}^2 \mid 0 \le x \le 1,\quad 2 \le y \le 4\},$$
$$9 := \{(x, y) \in \mathbb{Z}^2 \mid 2 \le x \le 3,\quad 2 \le y \le 4\}.$$

Set $n_{\mathrm{new}} = 2$.

Step 3.2: At node 6, branch on $x = 1$, creating nodes 10 and 11:

$$10 := \{(x, y) \in \mathbb{Z}^2 \mid 0 \le x \le 1,\quad 0 \le y \le 1\},$$
$$11 := \{(x, y) \in \mathbb{Z}^2 \mid 2 \le x \le 3,\quad 0 \le y \le 1\}.$$

Set $n_{\mathrm{new}} = 4$.

Step 3.3: Set up two independent lists $\mathcal{L}^1 = \{8, 10\}$, corresponding to $\mathcal{X}_1 = [0, 1] \cap \mathbb{Z}$, and $\mathcal{L}^2 = \{9, 11\}$, corresponding to $\mathcal{X}_2 = [2, 3] \cap \mathbb{Z}$. There are two sublists (one per independent list). Set $f^{\mathrm{UB},1} = 4$ and $f^{\mathrm{UB},2} = 4$.

Step 4: Compute $\underline{f}^{(8)} = 2, \underline{f}^{(9)} = 2, \underline{f}^{(10)} = \infty$, and $\underline{f}^{(11)} = 1$. Node 10 is fully fathomed and removed from list $\mathcal{L}^1$. Nodes 8 and 9 are added to $\mathcal{L}$:

$$\mathcal{L} = \left\{ \begin{array}{cccccc} 8: & 2 & 4 & -42 & - & 3 \\ 9: & 2 & 4 & -42 & 2 & 3 \end{array} \right\}.$$

Node 11 is added to $\mathcal{L}_{\mathrm{In}}$:

$$\mathcal{L}_{\mathrm{In}} = \left\{ \begin{array}{cccc} 11: & 1 & 4 & 3 \end{array} \right\}.$$

Step 5: Compute $\bar{f}^{(8)} = 4, \bar{f}^{(9)} = 2$ and $\bar{f}^{(11)} = 4$. Then, $f^{\mathrm{UB},1} = \bar{f}^{(8)} = 4$ and $f^{\mathrm{UB},2} = \min\{\bar{f}^{(9)}, \bar{f}^{(11)}\} = 2$. No fathoming is possible.

Step 6: Compute $\underline{F}^{(8)} = -21$ and $\underline{F}^{(9)} = -22$. Nodes 8 and 9 are outer fathomed ($\underline{F}^{(8)}, \underline{F}^{(9)} \ge F^{\mathrm{UB}} - \varepsilon_F$). At this point both independent lists $\mathcal{L}^1$ and $\mathcal{L}^2$ contain only outer-fathomed nodes and are discarded. This implies that the nodes they contain (node 8 from $\mathcal{L}^1$ and nodes 9 and 11 from $\mathcal{L}^2$) are fully fathomed. As a result, $\mathcal{L} = \emptyset$ and $\mathcal{L}_{\mathrm{In}} = \emptyset$.

Step 7: Return to Step 2 since there is no node i created at this iteration that belongs to $\mathcal{L}$ anymore.

Step 2: $\mathcal{L} = \emptyset$; terminate with $F^{\mathrm{UB}} = -22$ at $(2, 2)$.

Table 3
Test problems considered and their statistics. $\# x_i$ (resp. $\# x_c$) denotes the number of integer (resp. continuous) outer variables, $\# y_i$ (resp. $\# y_c$) the number of integer (resp. continuous) inner variables, $\# G$ the number of outer constraints and $\# g$ the number of inner constraints.

No.	Example (source)	Outer var.		Inner var.		Outer con.	Inner con.
		$\# x_i$	$\# x_c$	$\# y_i$	$\# y_c$	$\# G$	$\# g$
1	Example 1 (Moore and Bard, 1990)	1	0	1	0	0	4
2	Example 2 (Moore and Bard, 1990)	1	0	1	0	0	3
3	Eq. 3 (Edmunds and Bard, 1992)	0	1	1	0	0	3
4	Example 4 (Sahin et al., 1998)	0	2	2	0	0	1
5	Eq. 8.11 (Dempe, 2002)	0	2	2	0	2	3
6	am_1_0_0_1_01 (Mitsos, 2010)	0	1	1	0	1	5
7	am_1_1_1_0_01 (Mitsos, 2010)	1	1	0	1	1	1
8	am_1_1_1_1_01 (Mitsos, 2010)	1	1	1	1	1	1
9	am_1_1_1_1_02 (Mitsos, 2010)	1	1	1	1	1	1
10	am_3_3_3_3_01 (Mitsos, 2010)	3	3	3	3	4	3

In this example, inner fathoming, which arises from the use of the inner variable y in branching, allows two nodes to be fathomed, namely nodes 5 and 10, corresponding to just over 40% of the solution space. Furthermore, inner-variable branching makes it possible to show at level 1 of the branch-and-bound tree that the global solution must be such that $y \geq 2$. As a result, node 7, corresponding to points such that $(x, y) \in [4, 8] \times [0, 1]$, can be discarded from the tree as soon as node 5 is fathomed. This results from the fact that between them, nodes 5 and 7 cover the whole Y for $x \in [4, 8]$. This leads the elimination of a further 20% of the solution space. The extent to which this behavior is observed in other problems is discussed in Section 5.

4.3. *Proof of convergence*

We adapt the proof of finite ε-convergence of the standard Branch-and-Sandwich algorithm presented in Kleniati and Adjiman (2014b) to the generalized algorithm. The proof is based on the general convergence theory of branch-and-bound methods originally found in Horst (1988), and Horst and Tuy (1996). In particular, the subdivision process of the Branch-and-Sandwich algorithm was proved to be *exhaustive* (cf., Horst and Tuy (1996, Prop. IV.2.)) in Kleniati and Adjiman (2014b, Theorem 7) and the bounding scheme and the selection operation of the Branch-and-Sandwich algorithm were proved to be *consistent* (cf., Horst and Tuy (1996, Def. IV.4.)) and *bound improving* (cf., Horst and Tuy (1996, Definition IV.4)), respectively, in Kleniati and Adjiman (2014b, Theorems 4 and 5). Theorem 7 in Kleniati and Adjiman (2014b) is not affected by the addition of n_1 outer integer variables and m_1 inner integer variables as these may at worst add a finite number ($2^{n_1+m_1}$) of nodes to the branch-and-bound tree, based on the assumption, without loss of generality, that all integer variables are binary. As far as Theorems 4 and 5 in Kleniati and Adjiman (2014b) are concerned, they are derived based on Theorem 1 in the same paper and on Lemma 1 in Kleniati and Adjiman (2014a), which show that the best inner upper bound $f^{\mathrm{UB},p}$, $p \in P$, is, respectively, non-increasing and convergent to $w(\bar{x})$ as $\mathcal{X}_p$ converges to a singleton $\bar{x}$ in X. The reader is reminded that in the present paper, we propose a different inner upper bounding problem from the one presented in Kleniati and Adjiman (2014a). In particular, we have replaced the max−min formulation proposed in the original paper by the robust counterpart of the inner problem. Lemma 1 in Kleniati and Adjiman (2014a), showing that $f^{\mathrm{UB},p}$, $p \in P$, is non-increasing, applies to both formulations; hence, its proof need not be repeated.[5] On the other hand, it remains to be shown that Theorem 1 in Kleniati and Adjiman (2014b) is also valid for the inner upper bounding problem proposed in the present paper, i.e., that $f^{\mathrm{UB},p}$, $p \in P$, computed using problem (RIUB) is convergent. This is done in Theorem 1 below. Some required preliminary theoretical results are first stated.

Lemma 1 (Lemma 2 in Kleniati and Adjiman (2014a)). *Use $\mathcal{Y}_p \subseteq Y$ given in (12) to represent the Y (sub)domain maintained inside the independent list $\mathcal{L}^p$, $p \in P$. Then, for any $x \in \mathcal{X}_p$, the following holds:*

$$w(x) = \min_y \{f(x, y) \quad \text{s.t.} \quad g(x, y) \leq 0, \quad y \in \mathcal{Y}_p\}. \tag{17}$$

Definition 7 *(Horst (1988, Def. 2.4.))*. Let $\{k_q\}$, $q = 0, 1, \ldots$, be an infinite decreasing nested sequence of nodes such that any k_{q+1} is a child of k_q via a certain subdivision process. If the subdivision process is exhaustive, then

$$\lim_{q\to\infty} \delta(k_q) = 0, \tag{18}$$

or

$$\lim_{q\to\infty} k_q = \bigcap_q k_q = \{(\bar{x}, \bar{y})\}, \quad (\bar{x}, \bar{y}) \in X \times Y. \tag{19}$$

Remark 14. In view of Definition 7 and Kleniati and Adjiman (2014b, Theorem 7), any infinite decreasing sequence of successively refined nodes $\{k_q\}$ in the Branch-and-Sandwich method converges to a singleton as in (19). For the sequence of associated lists $\mathcal{L}^{p_q}$ such that $k_q \in \mathcal{L}^{p_q}$, we also have that $\mathcal{X}_{p_q} \subset X$ converges to a singleton in X in view of the exhaustive X subdivision process (bisection) alone, i.e.,:

$$\lim_{q\to\infty} \mathcal{X}_{p_q} = \{\bar{x}\}, \quad \bar{x} \in X. \tag{20}$$

Theorem 1. *Let $\{k_q\}$ be an infinite decreasing nested sequence of refined nodes. For the sequence of associated lists $\mathcal{L}^{p_q}$ such that $k_q \in \mathcal{L}^{p_q}$, we have that $\mathcal{X}_{p_q} \subset X$ converges to a singleton in X as in (20). Then, for the associated sequence of best inner upper bounds (16) (cf., Definition 2) we have:*

$$\lim_{q\to\infty} f^{\mathrm{UB},p_q} = w(\bar{x}).$$

Proof. In the limit, i.e., as q goes to infinity, the independent list $\mathcal{L}^{p_q}$ covers a single x value, as opposed to an X subdomain. This implies that no matter how much of the Y subdomain it may still cover, it can include one sublist only, i.e., the sublist corresponding to that x value. Thus, $\mathcal{L}^{p_q} = \{\mathcal{L}_1^{p_q}\}$. Then, based on Eq. (16), the best inner upper bound of $\mathcal{L}^{p_q}$ is:

$$\lim_{q\to\infty} f^{\mathrm{UB},p_q} = \min_{j \in \mathcal{L}_1^{p_q}} \{\bar{f}^{(j)}\}, \tag{21}$$

[5] Branching on x and branching on y have the same impact on the objective value of the inner upper bounding problem as in the proof of Lemma 1 in Kleniati and Adjiman (2014a), independent of which inner upper bounding problem we choose to employ.

Table 4
Numerical results with $\varepsilon_f = 10^{-5}$ and $\varepsilon_F = 10^{-3}$. N_{opt} is the number of the outer upper bounding problems solved until the optimal solution was computed for the first time. #UBD (#LBD) is the number of outer upper (lower) bounding problems solved and #Nodes the number of nodes required for convergence. Est. CPU is the approximate CPU time in s.

No.	Optimal value (F^*)	Branch-and-Sandwich optimal value (F^{UB})	N_{opt}	#UBD	#LBD	#Nodes	Est. CPU
1	−22	−22	1	3	5	11	0.055
2	5	5	5	5	7	13	0.065
3	$\frac{4}{9}$	0.444	1	1	1	1	0.090
4	−400	−400	1	1	1	1	0.160
5	−10.4	−10.4	1	1	2	3	0.150
6	−1	−1	1	1	1	1	0.060
7	0.5	0.5	2	3	11	11	1.670
8	−1	−1	4	7	12	13	1.320
9	0.209	0.209	1	1	1	1	0.290
10	−2.5	−2.5	1	1	1	1	0.130

where each $\bar{f}^{(j)}$ is obtained by solving problem (RIUB), given as:

$$\bar{f}^{(j)} = \min_{y,t}\{t \quad \text{s.t.} \quad f(X^{(j)}, y) \le t, \quad g(X^{(j)}, y) \le 0, \quad y \in Y^{(j)}\}. \tag{22}$$

By assumption, all $X^{(j)}, j \in \mathcal{L}_1^{p_q}$, have converged to $\bar{x}$; hence, the actual problem solved is

$$\bar{f}^{(j)} = \min_{y,t}\{t \quad \text{s.t.} f(\bar{x}, y) \le t, \quad g(\bar{x}, y) \le 0, \quad y \in Y^{(j)}\}, \tag{23}$$

which is equivalent to

$$\bar{f}^{(j)} = \min_{y \in Y^{(j)}} \max_{x \in \bar{x}}\{f(x, y) \quad \text{s.t.} g(x, y) \le 0\}. \tag{24}$$

Combining (21) with (24), we have that:

$$\lim_{q\to\infty} f^{\mathrm{UB},p_q} = \min_{j\in\mathcal{L}_1^{p_q}}\{\bar{f}^{(j)}\} = \min_{j\in\mathcal{L}_1^{p_q}} \min_{y\in Y^{(j)}}\{f(\bar{x}, y) \quad \text{s.t.} \quad g(\bar{x}, y) \le 0\}.$$

Finally, by using $\mathcal{Y}_{p_q} = \bigcup_{j\in\mathcal{L}_1^{p_q}} Y^{(j)}$, we can conclude that:

$$\lim_{q\to\infty} f^{\mathrm{UB},p_q} = \min_{y\in\mathcal{Y}_{p_q}}\{f(\bar{x}, y) \quad \text{s.t.} \quad g(\bar{x}, y) \le 0\} = w(\bar{x}),$$

where the last equality holds by Lemma 1. □

Theorem 2. *Given the standard assumptions made, i.e., continuity of all the functions involved, twice differentiability of their continuous relaxations and compactness of the host sets, the Branch-and-Sandwich algorithm for mixed-integer nonlinear bilevel problems terminates in a finite number of steps at an ε-optimal solution of problem* (1).

Proof. The finiteness of the integer host sets X_I and Y_I implies that the branch-and-bound tree relevant to branching on the integer variables is finite. Thus, without loss of generality, if we start by considering the integer variables only, the algorithm must visit a finite number of nodes. After this point onwards, convergence of the algorithm follows from the finite ε-convergence of the standard Branch-and-Sandwich algorithm proved in Kleniati and Adjiman (2014b). □

5. Computational experience

Branch-and-Sandwich for mixed-integer nonlinear bilevel problems was successfully applied to 10 test cases previously proposed in other works (Moore and Bard, 1990; Edmunds and Bard, 1992; Sahin et al., 1998; Dempe, 2002; Mitsos, 2010). They include between 2 and 12 variables and between 1 and 7 constraints. An overview of the problem statistics for each example is provided in Table 3. Convergence tolerances of $\varepsilon_f = 10^{-5}$ and $\varepsilon_F = 10^{-3}$ were used for the inner and outer problems, respectively. The global solutions of the inner lower bounding problem (RILB), the inner upper bounding problem (RIUB), the outer lower bounding problem (LB), the inner subproblem (ISP) and the outer upper bounding problem (UB) were found by employing the BARON solver (Tawarmalani and Sahinidis, 2005; Sahinidis, 2013).

The performance of the Branch-and-Sandwich algorithm for each test case is summarized in Table 4. The values of the objective function at the global solution, as reported in the original source, and as calculated by Branch-and-Sandwich are listed and it can be seen that the proposed algorithm identifies the solution in all cases. The numbers of outer upper bounding problems (#UBD) and outer lower bounding problems (#LBD) solved and the number of nodes generated to achieve convergence are also reported. In all cases, the number of nodes in the Branch-and-Sandwich tree is modest, with a maximum of 13 nodes reached for problems 2 and 8. In five instances, the problem is solved at the root node. We note that in problem 2, branching on x_i at the root node yields a 40% elimination of the solution space due to inner fathoming. Although the inner lower bound is not used to derive bounds on the outer problem, its computation is thus useful to accelerate convergence. In problem 8, branching on y_i at the root node results in a 50% elimination of the solution space again due to inner fathoming. Here, the ability to improve the inner lower bound by branching on y_i proves useful. However, further experience will be needed to determine whether this behavior is observed in larger problems.

For each example, an estimated CPU time is reported. This is based on the total time spent solving the various optimization subproblems, but does not include any overhead for formulating the subproblems or managing the branch-and-bound tree or the required lists, due to the fact that an implementation of the algorithm is under development. The CPU times are small. For these small problems, performance depends mostly on the number of nodes explored rather than the size of the problem. This is due to the fact that in small problems, the time required for the solution of the subproblems is fairly constant, so that the ability to fathom nodes early, and hence to solve fewer subproblems, is paramount.

6. Conclusions

In this work, the recently proposed Branch-and-Sandwich algorithm (Kleniati and Adjiman, 2014a, 2014b) has been extended to broader classes of bilevel problems by lifting the assumption of regularity of the inner problem, allowing the presence, in the inner problem, of equality constraints that depend on the outer variables and explicitly handling integer as well as continuous variables. This deterministic global optimization algorithm can thus be applied to a very broad class of mixed-integer nonlinear bilevel problems. The algorithm is uniquely characterized by the three following features: (i) it encompasses implicitly two branch-and-bound trees; (ii) it introduces simple bounding problems, always obtained from the bounding problems of the parent node;

(iii) it allows branch-and-bound with respect to outer (x) and inner (y) variables, but at the same time it keeps track of the partitioning of Y for successively refined subdomains of X. To complement the exposition of the algorithm, a proof of ϵ-convergence has been put forward in this work, extending results derived for the original version of the algorithm (Kleniati and Adjiman, 2014b). The algorithm has been applied successfully to 10 mixed-integer bilevel problems taken from the literature, with encouraging results both in terms of the number of branch-and-bound nodes explored and in terms of the estimated computational time. This promising performance is linked to the tightness of the inner upper bounds $f^{\mathrm{UB},p}$, which derives from the use of branching on the inner variables. Based on these promising results, more extensive computational experience is under investigation to explore the full potential of the Branch-and-Sandwich algorithm.

Acknowledgements

This paper is dedicated to Professor Ignacio Grossmann, in recognition of his inspirational contributions to process systems engineering. We gratefully acknowledge funding from the Leverhulme Trust through the Philip Leverhulme Prize and from the EPSRC through a Leadership Fellowship [EP/J003840/1].

Appendix A. Inner upper bound formulation for the illustrative example

In this section, we present in detail the inner upper bounding formulations (IUB) and (RIUB) that can be derived for Example 1. Although the formulations can be simplified due to the small size of the example, we present the general problem formulation for completeness. We first write the worst-case formulation of the inner problem.

$$f^{\mathrm{U},(k)} = \min_{Y_i \in y^{(k)}} \max_{x_i \in X^{(k)}} \{y_i \quad \text{s.t.} \quad -25x_i + 20y_i \le 30,$$

$$x_i + 2y_i \le 10, \quad 2x_i - y_i \le 15, \; -2x_i - 10y_i \le -15\},$$

which is reformulated equivalently as follows:

$$\begin{aligned} f^{\mathrm{U},(k)} = \quad & \min_{y_i} \quad y_i \\ & \text{s.t.} \quad -25x_i + 20y_i \le 30 \quad \forall x_i \in X^{(k)}, \\ & \qquad x_i + 2y_i \le 10 \quad \forall x_i \in [x^{\mathrm{L},(k)}, x^{\mathrm{U},(k)}] \cap \mathbb{Z}, \\ & \qquad 2x_i - y_i \le 15 \quad \forall x_i \in [x^{\mathrm{L},(k)}, x^{\mathrm{U},(k)}] \cap \mathbb{Z}, \\ & \qquad -2x_i - 10y_i \le -15 \quad \forall x_i \in [x^{\mathrm{L},(k)}, x^{\mathrm{U},(k)}] \cap \mathbb{Z}, \\ & \qquad y_i \in y^{(k)}. \end{aligned} \tag{IUB-Ex.1}$$

Observe that there is no need to introduce a variable t, as is required in the general inner upper bounding formulation (IUB), because the inner objective of Example 1 is x-independent.

To solve (IUB-Ex. 1) in this particular small-size integer case, one can reformulate it equivalently as a tractable problem with $4(x^{\mathrm{U},(k)} - x^{\mathrm{L},(k)} + 1)$ constraints. Nevertheless, following the more general case, the proposed formulation (RIUB) is:

$$\bar{f}^{(k)} = \min_{y_i \in [y^{\mathrm{L},(k)}, y^{\mathrm{U},(k)}] \cap \mathbb{Z}} \{y_i \quad \text{s.t.} \quad 20y_i \le 30 + 25x^{\mathrm{L},(k)},$$

$$2y_i \le 10 - x^{\mathrm{U},(k)}, \quad -y_i \le 15 - 2x^{\mathrm{U},(k)},$$

$$-10y_i \le -15 + 2x^{\mathrm{L},(k)}\}. \tag{RIUB-Ex.1}$$

After solving (RIUB-Ex.1), the values that are obtained are $\bar{f}^{(7)} = 1$ and $\bar{f}^{(9)} = 2$ and $\bar{f}^{(k)} = \infty$ for $k = 1, 2, 3, 4, 5, 6, 8, 11$. In reality, the worst (largest) value that the inner upper bound of this example can take is 4 and this latter value is reported in Section 5 for $k = 1, 2, 3, 4, 5, 6, 8, 11$. Note that this choice does not change the outcome of the algorithm. For example, $\underline{F}^{(1)} = -42$ for both $\bar{f}^{(1)} = \infty$ and $\bar{f}^{(1)} = 4$.

References

Adjiman CS, Androulakis IP, Floudas CA. Global optimization of mixed-integer nonlinear problems. AIChE J 2000;46:1769–97.
Apt KR, Zoeteweij P. An analysis of arithmetic constraints on integer intervals. Constraints 2007;12:429–68.
Bard JF. Practical bilevel optimization volume 30 of nonconvex optimization and its applications. Dordrecht: Kluwer Academic Publishers; 1998.
Belotti P. COUENNE: a user's manual. Clemson University Department of Mathematical Sciences; 2009.
Berger N, Granvilliers L. Some interval approximation techniques for MINLP. Technical Report CCSd/HAL: e-articles server (based on gBUS). In: Proceedings of the Eighth Symposium on Abstraction, Reformulation, and Approximation; 2009.
Bhattacharjee B, Green WH Jr, Barton PI. Interval methods for semi-infinite programs. Comput Optim Appl 2005;30:63–93.
Boyd S, Vandenberghe L. Convex optimization. Cambridge: Cambridge University Press; 2004.
Clark PA, Westerberg AW. Bilevel programming for steady state chemical process design I. Comput Chem Eng 1990;14:87–97.
Dempe S. Foundations of bilevel programming. Volume 61 of Nonconvex Optim. Appl. Dordrecht: Kluwer Academic Publishers; 2002.
Dempe S, Kalashnikov V. A mixed-discrete bilevel programming problem. In: Operations research proceedings 2003. Berlin: Springer; 2004. p. 284–91.
Dempe S, Kalashnikov V, Ríos-Mercado RZ. Discrete bilevel programming: application to a natural gas cash-out problem. Eur J Oper Res 2005;166:469–88.
Dempe S, Zemkoho AB. The generalized Mangasarian–Fromowitz constraint qualification and optimality conditions for bilevel programs. J Optim Theory Appl 2011;148:46–68.
Deng X. Complexity issues in bilevel linear programming. In: Multilevel optimization: algorithms and applications. Volume 20 of Nonconvex optimization and its applications. Dordrecht: Kluwer Academic Publishers; 1998. p. 149–64.
Edmunds TA, Bard JF. An algorithm for the mixed-integer nonlinear bilevel programming problem. Ann Oper Res 1992;34:149–62.
Faísca NP, Dua V, Rustem B, Saraiva PM, Pistikopoulos EN. Parametric global optimisation for bilevel programming. J Global Optim 2007;38:609–23.
Fanghänel D, Dempe S. Bilevel programming with discrete lower level problems. Optimization 2009;58:1029–47.
Floudas CA. Deterministic global optimization. Volume 37 of Nonconvex optimization and its applications. Dordrecht: Kluwer Academic Publishers; 2000.
Floudas CA, Gounaris CE. A review of recent advances in global optimization. J Glob Optim 2009;45:3–38.
Floudas CA, Gümüş ZH, Ierapetritou MG. Global optimization in design under uncertainty: feasibility test and flexibility index problems. Ind Eng Chem Res 2001;40:4267–82.
Grossmann IE, Biegler LT. Part II. Future perspective on optimization. Comput Chem Eng 2004;28:1193–218.
Gümüş ZH, Floudas CA. Global optimization of mixed-integer bilevel programming problems. Comput Manag Sci 2005;2:181–212.
Horst R. Deterministic global optimization with partition sets whose feasibility is not known: application to concave minimization, reverse convex constraints, DC-programming, and Lipschitzian optimization. J Optim Theory Appl 1988;58:11–37.
Horst R, Tuy H. Global optimization. Deterministic approaches. 3rd ed. Berlin: Springer-Verlag; 1996.
Kamath RS, Biegler LT, Grossmann IE. An equation-oriented approach for handling thermodynamics based on cubic equation of state in process optimization. Comput Chem Eng 2010;34:2085–96.
Kamath RS, Biegler LT, Grossmann IE. Modeling multistream heat exchangers with and without phase changes for simultaneous optimization and heat integration. AIChE J 2012;58:190–204.
Kleniati P-M, Adjiman CS. Branch-and-Sandwich: an algorithm for optimistic bilevel programming problems. In: Proceedings of the 21th European symposium on computer-aided process engineering. Elsevier volume 29 of computer-aided chemical engineering; 2011. p. 602–6.
Kleniati P-M, Adjiman CS. Branch-and-Sandwich: a deterministic global optimization algorithm for optimistic bilevel programming problems. Part I: theoretical development. J Global Optim 2014a., http://dx.doi.org/10.1007/s10898-013-0121-7.
Kleniati P-M, Adjiman CS. Branch-and-Sandwich: a deterministic global optimization algorithm for optimistic bilevel programming problems. Part II: convergence analysis and numerical results. J Global Optim 2014b., http://dx.doi.org/10.1007/s10898-013-0120-8.
Lemonidis P. Global optimization algorithms for semi-infinite and generalized semi-infinite programs [Ph.D. thesis]. Massachusetts Institute of Technology; 2008, chapter 9.
Loridan P, Morgan J. Weak via strong Stackelberg problem: new results. J Global Optim 1996;8:263–87.
Mitsos A. Global solution of nonlinear mixed-integer bilevel programs. J Global Optim 2010;47:557–82.

Mitsos A, Barton PI. Issues in the Development of Global Optimization Algorithms for Bilevel Programs with a Nonconvex Inner Program. Technical Report. Massachusetts Institute of Technology; 2006.

Mitsos A, Bollas GM, Barton PI. Bilevel optimization formulation for parameter estimation in liquid–liquid phase equilibrium problems. Chem Eng Sci 2009;64:548–59.

Mitsos A, Lemonidis P, Barton PI. Global solution of bilevel programs with a nonconvex inner program. J Global Optim 2008;42:475–513.

Moore JT, Bard JF. The mixed integer linear bilevel programming problem. Oper Res 1990;38:911–21.

Ryu J-H, Dua V, Pistikopoulos EN. A bilevel programming framework for enterprise-wide process networks under uncertainty. Comput Chem Eng 2004;28:1121–9.

Sahin KH, Ciric AR. A dual temperature simulated annealing approach for solving bilevel programming problems. Comput Chem Eng 1998;23:11–25.

Sahinidis NV. BARON 12.1.0: global optimization of mixed-integer nonlinear programs, user's manual; 2013.

Tawarmalani M, Sahinidis NV. Convexification and global optimization in continuous and mixed-integer nonlinear programming. Volume 65 of Nonconvex optimization and its applications. Dordrecht: Kluwer Academic Publishers; 2002.

Tawarmalani M, Sahinidis NV. Global optimization of mixed-integer nonlinear programs: a theoretical and computational study. Math Program 2004;99: 563–91.

Tawarmalani M, Sahinidis NV. A polyhedral branch-and-cut approach to global optimization. Math Program 2005;103:225–49.

Tsoukalas A. Global optimization algorithms for multi-level and generalized semi-infinite problems [Ph.D. thesis]. Imperial College London; 2009.

Vicente L, Savard G, Júdice J. Discrete linear bilevel programming problem. J Optim Theory Appl 1996;89:597–614.

Wiesemann W, Tsoukalas A, Kleniati P-M, Rustem B. Pessimistic bi-level optimization. SIAM J Optim 2013;23:353–80.